Design of Reservoir Computing Systems for the Recognition of Noise Corrupted Speech and Handwriting

Ontwerp van 'Reservoir Computing'-systemen
voor het herkennen van spraak en handschrift
in de aanwezigheid van achtergrondruis

Azarakhsh Jalalvand

Promotoren: prof. dr. ir. J.-P. Martens, prof. dr. ir. K. Demuynck
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. R. Van de Walle
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2014 - 2015

UNIVERSITEIT
GENT

ISBN 978-90-8578-767-9
NUR 984, 965
Wettelijk depot: D/2015/10.500/11

Examination Board

Acknowledgments

I cannot express enough the joy of successfully completing an important chapter of my life that contains so many characters. Some persons have been with me for a long time and some appeared in this chapter, but all of them had very important roles to help me with finishing this work. This PhD dissertation is the result of five years of life, research, collaboration, and many helps. Therefore, the least I could do is to thank everybody who was involved in any way in this part of my life.

First of all, my promoter, ***Prof. Jean-Pierre Martens****, who opened the door and let me in, not only in the speech lab for the doctoral research, but also in a new life that I always dreamed for. This, solely, is enough for me to show him my sincere gratitude. I never forget his first phone call for an interview in February 2009 that changed my life. During the research, his patience in guiding me was astonishing. Like in any other part of life, there have been tough times in my research, but I was always confident that he would be with me, support me, and help me to find and follow the correct road. In my opinion, he is the supervisor that any researcher wishes to have. My wish came true! He was always available to keep his critical eye on my work to guide and motivate me, and to revise my scientific manuscripts with his Parker pen. He was also available for social activities such as drink, dinner, New Year's events, etc. Jean-Pierre, thank you for giving me this opportunity, thank you for trusting me, and thank you for supporting me till the end!*

I also would like to thank my co-promoter, ***Prof. Kris Demuynck****, who shared his extraordinary knowledge with me and helped me when I was writing papers. We had plenty of brainstormings on both technical and practical issues. But his knowledge is not limited to the field of research. I also learned a lot from him during our chats at lunch in De Brug, drink, etc. and I am always impressed by his extensive general information.*

I would also like to thank the jury members for taking their time to critically read my thesis and for providing valuable comments which further improved this book.

During my research as a part of the ORGANIC project, I had a very close collaboration with many other researchers, especially with the Reservoir Lab in Ghent University. Particularly, I would like to thank ***Prof. Benjamin Schrauwen*** *and* ***Dr. David Verstraeten*** *for supporting me and for sharing their valuable knowledge on Reservoir Computing with me.*

An important parameter for enjoying a job is having pleasant colleagues and I was lucky because I worked among the best colleagues I could ask for. My special thanks go to my friend and my colleague, ***Fabian Triefenbach****. We had a very close daily collaboration and plenty of meetings together over the past 5 years. The most important things I learned from him were "friendly collaboration" and "efficiency at work". With the presence of* ***Jean-Pierre****,* ***Kris****,* ***Fabian****,* ***Bert****,* ***Brecht****,* ***Catherine****,* ***Dries****,* ***Johan****,* ***Line****,* ***Matthias****,* ***Michael*** *and* ***Stefaan****, the Speech Lab was a warm place to work and to live. I enjoyed lunch and coffee breaks as well as the drinks we usually had. I appreciate your effort for making me familiar with the new environment, new culture, and also for taking time to familiarize me with the more than one hundred types of Belgian beer, though it still needs another full time PhD period!*

Since June 2013, I joined the Multimedia Lab in Ghent University, where I work with many superb colleagues. Dear ***Prof. Rik Van de Walle****,* ***Prof. Erik Mannens****,* ***Prof. Peter Lambert*** *and* ***Prof. Wesley De Neve****, thank you for trusting me and offering me such a wonderful opportunity to join the lab and thank you for giving me enough time to work on this dissertation. Special thanks to my colleagues and friends* ***Abhineshwar****,* ***Baptist****,* ***Fréderic****,* ***Olivier****,* ***Steven****, and* ***Viktor****. Furthermore, warm regards go to the UGent staff, especially* ***Ellen****,* ***Laura****,* ***Muriel****,* ***Marnix****,* ***Jeroen****,* ***Ronny*** *and* ***Michiel*** *for all their administrative work and technical support which are necessary to perform our research.*

I would also like to express my sincere thanks to the promoter and co-promoter of my master's degree, ***Prof. Ahmad Akbari*** *and* ***Prof. Babak Nasersharif*** *who taught me the fundamentals of speech processing and sparked my interest in pattern recognition. Furthermore, I acknowledge my math teacher in high school,* ***Mr. Hooshang Rokhforooz****, who is always in my mind, because he first discovered and developed my interest in mathematics and guided me to follow this way.*

Apart from the academic world, I would like to thank my whole family and friends who always support me in my endeavors. I have a great life with all of you by my side. Particularly, my sincere gratitude goes to my parents who worked so hard to provide me with a comfortable life and the

opportunity to continue my study and follow my dreams. I wish my father was still alive to celebrate this achievement together.

Last but not least, I would like to thank ***Poona****, my lovely wife, my best friend, and above all, the joy of my life. Dear Poona, your love, companionship and support since the bachelor studies are the main reasons of all my achievements. In the sands of time, I see our footprints side by side in all the happy and difficult moments. Being together makes every impossible possible for us!*

Azarakhsh Jalalvand
Gent, 27 February 2015

Table of Contents

Acronyms

AFE	**A**dvanced **F**ront-**E**nd
AI	**A**rtificial **I**ntelligence
AM	**A**coustic **M**odel
ANN	**A**rtificial **N**eural **N**etworks
ASR	**A**utomatic **S**peech **R**ecognition
ATM	**A**utomated **T**eller **M**achine
CDR	**C**ontinuous **D**igit **R**ecognition
CE	**C**ross-**E**ntropy
CMLLR	**C**onstrained **MLLR**
CNN	**C**onvolutional **N**eural **N**etworks
DAE	**D**enoising **A**uto**E**ncoder
DBM	**D**eep **B**oltzmann **M**achine
DBN	**D**eep **B**elief **N**etworks
DCN	**D**eep **C**onvex **N**etworks
DCT	**D**iscrete **C**osine **T**ransformation
DER	**D**igit **E**rror **R**ate
DFT	**D**iscrete **F**ourier **T**ransformation
DNN	**D**eep **N**eural **N**etworks
EB	**E**xemplar-**B**ased

EBP . . .	**E**rror **B**ack-**P**ropagation
ELM . . .	**E**xtreme **L**earning **M**achine
EM	**E**xpectation **M**aximization
ESN . . .	**E**cho **S**tate **N**etworks
ETSI . . .	**E**uropean **T**elecommunications **S**tandards **I**nstitute
FFNN . .	**F**eed-**F**orward **N**eural **N**etworks
FFT . . .	**F**ast **F**ourier **T**ransform
GMM . .	**G**aussian **M**ixture **M**odel
GPS . . .	**G**lobal **P**ositioning **S**ystem
HDR . . .	**H**andwritten **D**igit **R**ecognition
HEQ . . .	**H**istogram **EQ**ualization
HMI . . .	**H**uman-**M**achine **I**nterface
HMM . .	**H**idden **M**arkov **M**odel
HTK . . .	**H**idden Markov Model **T**ool**K**it
HWR . .	**H**and**W**ritten **R**ecognition
ITU . . .	**I**nternational **T**elecommunication **U**nion
JUD . . .	**J**oint **U**ncertainty **D**ecoding
LDA . . .	**L**inear **D**iscriminant **A**nalysis
LIN . . .	**L**eaky **I**ntegrator **N**euron
LM	**L**anguage **M**odel
LPC . . .	**L**inear **P**redictive **C**oding
LSM . . .	**L**iquid **S**tate **M**achine
LSTM . .	**L**ong **S**hort-**T**erm **M**emory
LVCSR .	**L**arge **V**ocabulary **C**ontinuous **S**peech **R**ecognition

MCE	**M**inimum **C**lassification **E**rror
MDT	**M**issing **D**ata **T**echniques
MFCCs	**M**el-**F**requency **C**epstral **C**oefficients
MIDA	**M**utual **I**nformation-based **D**iscriminant **A**nalysis
ML	**M**achine **L**earning
MLE	**M**aximum **L**ikelihood **E**stimation
MLLR	**M**aximum **L**ikelihood **L**inear **R**egression
MLP	**M**ulti-**L**ayer **P**erceptron
MMI	**M**aximum **M**utual **I**nformation
MNIST	**M**odified **N**ational **I**nstitute of **S**tandards and **T**echnology
MSE	**M**ean **S**quared **E**rror
MSVA	**M**FCC, **S**pectral Subtraction, Spectral Flooring And Mo**V**ing-**A**verage Smoothing
MVN	**M**ean & **V**ariance **N**ormalization
NF	**N**oise **F**raction
OCR	**O**ptical **C**haracter **R**ecognition
PAC	**P**hase **A**uto**C**orrelation
PER	**P**hone/Phoneme Recognition **E**rror **R**ate
PIN	**P**ersonal **I**dentification **N**umber
PLP	**P**erceptual **L**inear **P**rediction
RASTA	**R**el**A**tive **S**pec**T**r**A** Filtering of Log Domain Coefficients
RBM	**R**estricted **B**oltzmann **M**achine
RC	**R**eservoir **C**omputing
RCN	**R**eservoir **C**omputing **N**etworks
RNN	**R**ecurrent **N**eural **N**etworks

RTRL	**R**eal-**T**ime **R**ecurrent **L**earning
SER	**S**tate **E**rror **R**ate
SNR	**S**ignal-to-**N**oise **R**atio
SPLICE	**S**tereo-Based **P**iecewise **Li**near **C**ompensation for **E**nvironments
SPRAAK	**S**peech **P**rocessing, **R**ecognition & **A**utomatic **A**nnotation **K**it
SSDA	**S**tacked **S**parse DBN-Based **D**enoising **A**utoencoder
SVCSR	**S**mall **V**ocabulary **C**ontinuous **S**peech **R**ecognition
SVIWR	**S**mall **V**ocabulary **I**solated **W**ord **R**ecognition
SVM	**S**upport **V**ector **M**achine
TDNN	**T**ime-**D**elay **N**eural **N**etworks
TRAP	**T**empo**RA**l **P**attern
UD	**U**ncertainty **D**ecoding
VAD	**V**oice **A**ctivity **D**etection
VTLN	**V**ocal **T**ract **L**ength **N**ormalization
VTS	**V**ector **T**aylor **S**eries
WER	**W**ord **E**rror **R**ate
WTA	**W**inner **T**ake **A**ll

Nomenclature

Conventions:

- Sequences and matrices are specified with bold Roman letters. (e.g., $\mathbf{Y}$)
- Vectors are specified with uppercase Roman letters (e.g., Y).
- Components of vectors or sequences are specified with lowercase Roman letters (e.g., $u_{t,i}$).
- Scalars that are not components of vectors or sequences are specified by Greek letters (e.g., ρ) or capital italic letters (e.g., N^{res}).
- Probabilities, likelihoods and density functions are all, for simplicity, denoted with a capital $P(\cdot)$ (e.g., $P(\mathrm{X}|\mathrm{q})$).

$\mathbf{U}$, U, u .	input feature vector obtained with the front-end
$\mathbf{X}$, X, x .	static acoustic features
$\mathbf{R}$, R, r .	reservoir state
$\mathbf{D}$, D, d .	desired ANN output
$\mathbf{Y}$, Y, y .	estimated ANN output
$\mathbf{H}$, H, h .	hidden ANN state
$\mathbf{W}^{in}$ . . .	input weight matrix
$\mathbf{W}^{rec}$ . .	recurrent weight matrix
$\mathbf{W}^{out}$ . .	output weight matrix
$\mathbf{W}$	weight matrix/vector of an ANN/neuron
N^{frm} . .	number of training frames

N^{out}	number of outputs
N^{res}	number of reservoir neurons
N^{in}	number of acoustic inputs
N^{frm}	total number of trainable parameters
K^{in}	amount of input connections per neuron
K^{rec}	amount of recurrent connections per neuron
ρ	spectral radius
λ	leak rate
α_U	input scale factor
α_R	recurrent scale factor
V_a	variance of observation inputs distribution
V_b	variance of recurrent inputs distribution
V_{opt}	optimal variance of the reservoir neuron activation
τ	time constant of the reservoir implied by (ρ,λ)
τ_ρ	time constant implied by ρ
τ_{fr}	time shift between frames
τ_λ	time constant implied by λ
$\overline{\mathbb{E}[\|R\|]}$	mean reservoir states
T_{state}	average duration of a model state
P_o	insertion penalty
ϵ	regularization term
F_{in}	bandwidth of input dynamics
F_{in}^*	bandwidth of dynamics (in kHz)
F_{out}	bandwidth of output dynamics

$\lvert B(f)\rvert$ . .	mean power spectrum of inputs
$f_{res}(\cdot)$ . .	activation function of a reservoir neuron
$f_{out}(\cdot)$ . .	activation function of a readout neuron
$f_{map}(\cdot)$.	output mapping function
$\mathbf{q}$, q . . .	word state sequence & single word state
$\hat{\mathbf{q}}$, $\hat{\mathrm{q}}$ . . .	optimal word state sequence & single word state
$\mathbf{w}$, w . . .	word sequence & single word identity
$\hat{\mathbf{w}}$, $\hat{\mathrm{w}}$ . . .	optimal word sequence & single word identity
L	number of ANN layers
w_{ik}	ANN connection between neuron i and k
$\mathbf{I}$	identity matrix
∂z	partial derivative of z
Δz, Δ^2z .	first and second order differentials of z
$\mathbf{A}^{-1}$ . . .	pseudo inverse of a matrix $\mathbf{A}$
$\mathbf{A}^T$	transpose of a matrix $\mathbf{A}$
$[\mathrm{A}, \mathrm{B}]$ . .	concatenation of vectors A and B
$\{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$	sequence of vectors A, B and C

Samenvatting

In de natuur lijkt een flexibel en doordringend communicatiesysteem tussen individuen een voorwaarde te zijn voor complex sociaal gedrag en intelligentie. *Spraak* –de gevocaliseerde vorm van intermenselijke communicatie– speelt als dusdanig een centrale rol in de unieke mogelijkheden van de menselijke soort. Het toestaan van de mens om via spraak te communiceren met machines zorgt dus niet alleen voor een meer natuurlijke interactie tussen mens en machine, het kan ook helpen bij het begrijpen van een aantal belangrijke aspecten van intelligentie in het algemeen. Een van de eerste stappen die spraakgestuurde man-machine communicatie mogelijk maakt is de omzetting van het spraaksignaal in een reeks woorden, een proces dat gewoonlijk wordt aangeduid als *spraakherkenning*.

Een systeem dat ontworpen is voor het herkennen van menselijke spraak heet een "automatisch spraakherkenningssysteem", of afgekort, een ASR-systeem *[Automatic Speech Recognition]*. ASR-systemen worden al op grote schaal gebruikt in smartphones, GPS-navigatiesystemen, ondertitelingssystemen, enz. Het wordt ook gebruikt om mensen met een handicap te helpen beter te communiceren met hun omgeving. Maar misschien hebben velen onder ons al problemen ondervonden bij het gebruik van de ASR-technologie. Bijvoorbeeld, het bellen van de juiste persoon via "voice dialing" kan problematisch zijn in een omgeving met achtergrondgeluid (bijvoorbeeld in een auto of op een feestje). De gevoeligheden aan spreker, accent en emotie zijn gekende problemen. Verder verhoogt de onnauwkeurigheid dramatisch wanneer spontane spraak moet herkend worden in plaats van formele spraak.

Al decennia lang zijn Gaussian mengselmodellen (GMM) *[Gaussian Mixture Models]* erg populair, vooral in commerciële producten. GMMs gebruiken kansdichtheidsfuncties om te bepalen hoe groot de kans is dat een waargenomen spraakfragment overeenkomt met een specifieke linguïstische eenheid (bv, foneem of een woord). De parameters van de kansdichtheidsfuncties worden bepaald door middel van een trainingsprocedure die een grote hoeveelheid spraak nodig heeft waarvan de transcriptie in termen van die eenheden gekend is. GMMs zijn populair omdat ze op een efficiënte wijze te ontwerpen zijn (er bestaat veel software voor) en omdat ze tijdens het gebruik niet te veel berekeningen vereisen. Ondanks de enorme vooruitgang van de afgelopen jaren, zijn GMM-gebaseerde herkenners echter nog geen concurrent voor de menselijke luisteraar. Vooral de grote degradatie die optreedt wanneer men spraak dient te herkennen in omstandigheden die niet aanwezig waren in de trainingsvoorbeelden blijft een groot probleem, en dus, het verminderen van die gevoeligheid is nog steeds een uitdaging.

Veel van de reeds voorgestelde methodes focussen op het verwijderen of compenseren van de effecten van ruis op de akoestische elementen. Anderen streven ernaar om de akoestische modellen aan te passen aan de ruis of om ze in te zetten op een meer lawaai-robuuste manier. GMMs vervangen door alternatieve modellen, zoals neurale netwerken (ANNs) *[Artificial Neural Networks]* die geacht worden gelijkaardig te werken als een menselijk brein, wordt eveneens als een veelbelovende aanpak beschouwd.

ANNs bestaan uit artificiële neuronen die aan elkaar gekoppeld zijn door middel van gewogen verbindingen. De gewichten van deze verbindingen zijn zodanig getraind dat de outputs van het ANN de a posteriori kansen van de spraakeenheden benadert na waarneming van het akoestisch signaal.

Hoewel de toepassing van ANNs voor het bepalen van de kans van een spraakeenheid dateert uit de late jaren tachtig, was de techniek de laatste jaren een beetje uit het beeld verdwenen. Krachtiger hardware en software maken het nu echter mogelijk om ANNs bouwen met vele lagen en vele neuronen per laag, en daarvoor veel meer geavanceerde en complexe trainingsalgoritmes dan deze uit de jaren tachtig aan te wenden.

Deze nieuwe ANNs noemt men Deep Neural Networks (DNN). Deze DNNs hebben recent reeds geleid tot een betere LVCSR en ze zullen vermoedelijk op termijn ook tot meer lawaai robuuste LVCSR leiden. Een ander interessant ANN-type is de Recurrent Neuraal Netwerk (RNN) dat vertraagde feedback introduceert door het opnemen van extra recurrente verbindingen.

Deze vertraagde feedback stelt het RNN in staat om met lange termijn afhankelijkheden tussen akoestische waarnemingen op een efficiënte manier om te gaan (zoals Infinite Impulse Response filters doen in lineaire systeemtheorie). Ondanks de veelbelovende resultaten verkregen met recente ANN-gebaseerde spraakherkenners, is er nog aarzeling om ze op grote schaal te gebruiken, vooral omdat de trainingsprocedures meestal kritisch en zeer tijdrovend zijn.

In dit opzicht kunnen Reservoir Computing Networks (RCNs) een interessant alternatief bieden. Dergelijke netwerken zijn immers zeer eenvoudig te trainen. De belangrijkste doelstelling van mijn onderzoek was om de levensvatbaarheid van dit alternatief te bewijzen in de context van ruisrobuustheid.

In het kort gezegd is een RCN een speciaal type RNN, namelijk een waarin (1) de ingangsverbindingen en de recurrente verbindingen voor de niet-lineaire neuronen willekeurig bepaald zijn, (2) het uitgangssignaal wordt gegenereerd door een laag van lineaire neuronen en (3) alleen de gewichten van deze neuronen worden getraind. Het ongetrainde niet-lineaire gedeelte en de lineaire uitgangslaag worden respectievelijk het reservoir en de readout van het RCN genoemd. De gewichten van het reservoir worden eerst genomen uit een normale verdeling met een gemiddelde nul en een variantie één en ze waren daarna correct geschaald zodat een stabiel dynamisch systeem verkregen wordt dat een goed evenwicht nastreeft tussen de effecten die nieuwe en van vroegere inputs opleveren in de output van het RCN.

Het trainen van de uitvoerlaag van een RCN komt neer op het minimaliseren van de gemiddelde kwadratische fout (MSE) *[Mean Squared Error]* tussen de werkelijke en de gewenste output. Deze laatste wordt afgeleid uit de sequentie van spraakeenheden die op haar beurt wordt afgeleid uit de orthografische transcriptie van de spraak. De beoogde uitvoergewichten zijn de oplossing van een stelsel lineaire vergelijkingen. Aangezien het trainen van een conventioneel ANN gebaseerd is op een iteratieve gradiëntafdalingsaanpak en aangezien een dergelijke aanpak niet gegarandeerd een globale optimum bereikt, is het trainen van een RCN opvallend voordelig.

Dankzij de gemakkelijke en eenvoudige trainingsprocedure is het mogelijk om de grootte van het reservoir te vergroten (dwz. het aantal knooppunten te vergroten) tot op een niveau dat ver buiten het bereik van een conventioneel RNN ligt (reservoirs met 32K knooppunten zijn nog eenvoudig te trainen). Hierdoor kan het reservoir een rijke, hoog-dimensionale toestandsruimte creëren die geschikt is voor het onderscheiden van de spraakeenheden met behulp van een eenvoudig lineair regressiemodel.

Doordat de signaaldynamiek ook in die ruimte gecodeerd wordt, en doordat de spraak- en ruisdynamiek verschillend zijn, verwacht men dat het RCN tot op zekere hoogte in staat is de effecten van de spraak en deze van de ruis van elkaar te scheiden.

In dit werk worden RCNs voor de eerste keer (voor zover ik weet) toegepast voor het akoestisch modelleren van spraakeenheden met het oog op ruisrobuuste, continue spraakherkenning. Alhoewel het uiteindelijke doel is om ruisrobuuste LVCSR te realiseren, koos ik er voor om te beginnen met het bestuderen van de lawaairobuustheid van RCN-gebaseerde systemen in de context van continue spraakherkenning met een kleine woordenschat (SVCSR) *[Small Vocabulary Continuous Speech Recognition]*. Ik heb meer bepaald gefocust op continue cijferherkenning (CDR *[Continious Digit Recognition]*). CDR is immers een relevante taak op zich, en er bestaat een gekende dataset (Aurora-2) die speciaal ontworpen werd voor het bestuderen van ruisrobuuste cijferherkenning. Deze dataset bevat zowel ruisloze continue cijferreeksen als vele ruizige versies daarvan (inclusief verschillende soorten ruis met verschillende signaal-tot-ruisverhoudingen (SNR) *[Signal-to-Noise Ratio]*).

Aangezien de toepassing van RCN voor spraak(-herkenning) vrij nieuw was toen dit werk begon, dienden alle mogelijke aspecten van RCNs bestudeerd te worden. In het bijzonder, diende ik mezelf vertrouwd te maken met het opzetten van goede reservoirs. Aangezien alle reservoirgewichten willekeurig worden ingesteld bij de start van het trainingsproces en vanaf dat ogenblik ook vast blijven, is het initialiseren van het reservoir een cruciale stap. De initialisatie van de gewichten bepaalt het vermogen van het reservoir om de oorspronkelijke invoer te projecteren naar een interessante hoog-dimensionale ruimte waarin zowel de tijdelijke verbanden op korte en lange termijn gecodeerd worden.

Om een cijferherkenningssysteem te ontwikkelen, heb ik eerst veel experimenten naar de herkenning van geïsoleerde gesproken cijfers uitgevoerd en vervolgens experimenten naar de herkenning van continu gesproken cijfers. Op een gegeven ogenblik had ik systemen die getraind waren op ruisloze spraak en die state-of-the-art prestaties leverden op ruizige spraak. Ze leverden echter nog geen competitieve prestaties in ruisloze omstandigheden.

Daarom onderzocht ik meer complexe RCN-architecturen met meerdere lagen. Elke laag bestaat uit een basis RCN, bestaande uit een reservoir en een lineaire uitvoerlaag. Het eerste RCN wordt gestimuleerd door de akoestische ingang, het tweede door de uitgangen van de eerste laag, enzovoort. De netwerken worden een-voor-een getraind, wat inhoudt dat er geen gezamenlijke optimalisatie van het aaneengeschakelde netwerk wordt uitgevoerd. Het aaneenschakelen van RCNs leidde tot verbeterde spraakherkenning in zowel ruisloze als ruizige omstandigheden. Ik meen dat de belangrijkste redenen hiervoor de volgende zijn: (1) elk RCN induceert een nieuwe willekeurige projectie en reduceert bijgevolg de kans op het missen van interessante niet-lineaire projecties, (2) elk RCN kan fouten, waargenomen in de resultaten van de vorige laag, corrigeren, en (3) elk RCN voegt dynamisch modelleringscapaciteit toe.

Om de systemen verder te kunnen verbeteren, introduceerde ik ook bidirectionele reservoirs die het vermogen bezitten omzowel de linkse als de rechtse context van een akoestische waarneming in rekening te brengen. Hierdoor kunnen ze beter de co-articulaties modelleren die een gevolg zijn van de eerstvolgende klanken die de spreker nog zal uitspreken.

Een vaak gebruikte oplossing om de invloed van ruis op de prestaties te verminderen bestaat erin het spraakherkenningssysteem te trainen met voorbeelden die reeds verschillende types toegevoegde ruis op verschillende niveaus bevatten. Maar zelfs in dit scenario bestaat nog altijd de kans dat men tijdens de werking geconfronteerd wordt met een nieuwe toestand die tijdens het trainen niet voorkwam. Een praktische oplossing hiervoor bestaat erin het spraakherkenningssysteem met een beperkte inspanning aan te passen aan een nieuwe toestand. In dit verband, heb ik aangetoond dat het mogelijk is zonder manuele interventie een kleine module te trainen die in de gegeven omstandigheden de oorspronkelijke uitvoer van het RCN kan verbeteren.

Een algemene conclusie van alle bovengenoemde stappen was dat zowel de grootte en de architectuur van het reservoir op een belangrijke manier bijdragen tot het verkrijgen van nauwkeurige en ruisrobuuste systemen. Echter, hoe complexer de architectuur is, hoe langer het duurt om een reservoir met geschikte reservoirparameters te kunnen identificeren. Daarom was ik er van overtuigd dat er nood was aan een meer begrijpelijke, automatische procedure voor het bepalen van quasi-optimale waarden voor de reservoirparameters. Mijn onderzoek leidde tot een nieuwe analyse van RCN als een niet-lineair dynamisch systeem. Ik heb een aantal empirische verbanden tussen de de reservoirparameters en de dynamica van de ingang en de uitgang van het RCN kunnen afleiden, en deze verbanden hebben uiteindelijk geresulteerd in een recept voor het ontwerpen van reservoirs dat tegelijk zeer efficiënt, eenvoudige en begrijpelijk is.

Daar waar GMMs zeer goed presteren als ze in vergelijkbare omstandigheden getraind en gebruikt worden, leiden RCNs vooral tot superieure prestaties in niet-vergelijkbare omstandigheden. Daarom was het volgende doel van mijn onderzoek een manier vinden om de voordelen van beide benaderingen te combineren.

Ik onderzocht twee verschillende strategieën: een fusie van waarschijnlijkheden en een tandemaanpak. In de waarschijnlijkheidsfusie wordt de gewogen som van de waarschijnlijkheden die uit de GMM-gebaseerde en de RCN-gebaseerde akoestische modellen voortvloeien, aan de decoder aangeboden. In de tandemaanpak wordt een conventioneel GMM-systeem aangestuurd door de uitgangen van het RCN, of vice versa, krijgt een RCN-systeem de uitgangen van de GMMs als input. Mijn werk toonde aan dat de combinatie van deze twee systemen nuttig is, zolang de training- en de testcondities vergelijkbaar zijn, maar een lichte prestatieverlies induceert in situaties die sterk afwijken van de situaties die tijdens de training gezien werden.

In plaats van een conventionele ruisrobuuste front-end te gebruiken, zoals de ETSI geavanceerde front-end (AFE) *[Advanced Front-End]*, heb ik een eenvoudige front-end (pure MFCC *[Mel-frequency cepstral coefficients]*) getest en heb ik de features die door die front-end berekend werden via een RCN omgevormd naar features die veel beter op de 'ruisvrije' features lijken. Een systeem dat een dergelijk featuretransformatie beoogt wordt algemeen een Denoising Auto-encoder (DAE) genoemd.

Uit mijn werk blijkt dat de introductie van een DAE na de MFCC front-end wel de prestatie doe toenemen in het geval het akoestische model op ruisvrije spraak werd aangeleerd, maar dat ze in het geval van een multi-style training van het akoestisch model een degradatie van de prestatie veroorzaakt. Dit kan alleen maar betekenen dat de DAE informatie verwijdert die anders door de RCN-herkenner had kunnen gebruikt worden.

De laatste ASR die ik onderzocht heb is een RCN-gebaseerd akoestisch model dat bestaat uit 3-lage bi-directionele reservoirs met 8K neuronen per laag, en wordt aangestuurd door de zogenaamde AFEs. Wanneer deze systemen worden getraind op schone spraak, leiden ze tot een gemiddeld foutenpercentage (WER) *[word error rate]* over verschillende soorten ruis en geluidsniveaus van 9.0% op Aurora-2, in vergelijking met de 13.2% bereikt door een GMM-systeem. Wanneer ons systeem getraind wordt op zowel schone als ruisbevattende spraak, dan daalt zijn gemiddelde WER tot 5.8% in vergelijking met de 8.2% die met een GMM-gebaseerde ASR bereikbaar is.

De belangrijkste conclusies van mijn werk op spraakherkenning waren dat (1) RCN-gebaseerde systemen kunnen worden ontworpen volgens een eenvoudig recept, (2) vakkundig ontworpen RCN-gebaseerde systemen kunnen concurreren met conventionele systemen in gelijke omstandigheden en (3) RCN- gebaseerde systemen meer robuustheid bieden tegen verschillen tussen de leer- en de testomstandigheden.

De vraag bleef echter of deze conclusies ook standhouden in andere toepassingen, zoals b.v. beeldherkenning. Om dit te controleren heb ik beslist om een RCN-systeem voor de herkenning van geïsoleerde handgeschreven cijfers te ontwikkelen. Ik heb dit systeem getest op de internationaal bekende MNIST dataset. Elk sample van deze dataset is gecodeerd in 28 $\times$ 28 grijswaarde pixels. Daarnaast, heb ik ook gebruik gemaakt van een ruisbevattende versie van deze dataset, waarop andere auteurs reeds experimenten met Deep Belief Networks (DBNs) hebben uitgevoerd. Net als bij mijn onderzoek naar ruisrobuuste spraakherkenning, gebruikte ik een RCN op twee verschillende manieren, als classificator (herkenner) en als DAE (ruisfilter).

De belangrijkste conclusie van mijn onderzoek was dat de voorgestelde strategie voor het optimaliseren van de reservoirparameters, hoewel oorspronkelijk bedacht voor spraakherkenningstaken, ook toepasbaar is op handschriftherkenning. Aangezien het een DER *[Digit Error Rate]* van 0.8% bereikt, kan het concurreren met de state-of-the-art. Bij het testen op ruisbeelden (met vijf types ruis), levert het RCN-gebaseerd systeem een gemiddeld foutenpercentage van 37% op. Hetzelfde systeem, maar ditmaal getraind op zowel schone als ruizige beelden levert een DER van 1.5% op voor de schone en een van 3.5% voor de ruizige beelden op.

Door toevoeging van een RCN-gebaseerde DAE, getraind om vijf verschillende types ruis te kunnen verwijderen uit de beelden, werd een dramatische daling van de DER verkregen na training van het systeem op schone samples: de DER daalde van 37% tot minder dan 2.1%. De gerapporteerde DER voor DBN-systeem op dezelfde gegevensreeks was 2.3%.

Tot slot wens ik te benadrukken dat mijn werk laat zien dat RCN-gebaseerde akoestische modellering kan worden opgenomen in het statistisch raamwerk van een ruisrobuuste spraakherkenner voor doorlopende cijfers. De gepresenteerde modellen bereiken state-of-the-art prestaties in schone omstandigheden en verbeteren de state-of-the-art in ruizige omstandigheden. Mijn werk toont ook aan dat er een eenvoudige maar effectieve strategie bestaat voor het optimaliseren van de reservoirparameters op basis van eenvoudig te verzamelen kennis omtrent de ingang en de gewenste uitgang van het systeem. Deze strategie lijkt zowel te werken voor de herkenning van gesproken als van handgeschreven cijfers. Blijkbaar is het idee van het willekeurig vastleggen en het herhaaldelijk na elkaar aanwenden van pools van niet-lineaire neuronen een goede basis voor het modelleren van complexe temporele patronen (in spraakherkenning) of van complexe lokale verbanden tussen aangrenzende pixels (in beeldherkenning). Gezien het feit dat nog maar weinig onderzoek naar reservoirgebaseerde systemen voor complexe taken gebeurd is, kan men het gebied nog als jong en in ontwikkeling zijnde beschouwen. Bijgevolg is er nog voldoende ruimte voor het verkennen van nieuwe ideeën en voor het maken van verdere vooruitgang in dit domein.

Abstract

In nature, a flexible and pervasive communication system among individuals seems to be a prerequisite for complex social behavior and intelligence. As such, *Speech* –the vocalized form of inter-human communication– plays a central role in the unique capabilities of the human species. Allowing humans to communicate with machines via speech thus not only provides for a more natural man-machine interaction, it may also help in understanding some key aspects concerning intelligence in general. One of the first steps needed to allow speech-based man-machine communication is the conversion of the speech signal into a sequence of words, a process which is usually referred to as *speech recognition*.

A system that is designed to perform human speech recognition is called an "Automatic Speech Recognition System", or briefly, an ASR system. ASR systems are already widely used in many devices such as smartphones, GPS navigation systems, subtitling systems, etc. It is also utilized to help people with disabilities to better communicate with their environment. Widespread application of ASR-technology is however hampered by the fact that ASR is far less robust than human speech recognition. The current generation ASR systems cannot cope very well with noisy speech, with the large variability in human voices, nor with the variability in the speech due to accent or emotion. For instance, calling the correct person via 'voice dialer' can be problematic in an environment with background noise (e.g., inside a car or in a party).

Traditionally, Gaussian Mixture Models (GMMs) were used to relate the observed speech signal (represented by some features derived from the speech signal) with the underlying linguistic units such as phoneme or word. This procedure is called *Acoustic Modeling*.

The noise robustness of such systems could be improved by compensating the effect of noise on the acoustic features or by adjusting the GMMs so that they describe noisy instead of clean speech features. Adapting the acoustic models is also proved to be useful when coping with other sources of variability such as speaker variations or regional accents.

The more recent approaches, and the one also adopted in this work, consist of replacing the GMMs with alternative modeling techniques that mimic the operation of the human brain more closely, such as Artificial Neural Networks (ANNs).

ANNs are composed of artificial neurons that are linked to each other by means of weighted connections. The connection weights are trained such that the output neurons approximate the posterior probabilities of the linguistic units given the acoustic observations. Although the application of ANNs for speech unit probability estimation dates back to the late eighties, the technique was over-shadowed by GMMs until very recently. More powerful hardware and software make it possible now to construct ANNs with many layers and many neurons per layer, and to employ more sophisticated and complex training algorithms for the training of such networks. These so-called Deep Neural Networks (DNN) have led to better LVCSR and are believed to improve noise-robustness, as well. Another interesting ANN type is the Recurrent Neural Networks (RNNs) that add recurrent connections to the network. These recurrent connections introduce a delayed feedback which permits the RNN to handle long-term dependencies between acoustic observations in an efficient way (like Infinite Impulse Response filters do in linear system theory). Despite the promising results obtained with recent ANN-based speech recognition systems, there is still hesitation in using them on a large scale, mainly because the training procedures are usually difficult and very time consuming.

In this respect, Reservoir Computing Networks (RCNs) may offer an interesting alternative as the training of such systems is easy and robust. The main objective of my research was to prove the viability of this alternative in the context of noise robustness mainly in speech recognition and briefly in image recognition.

An RCN is a special type of RNN, namely, one in which (1) the input connections and the recurrent connections entering the non-linear neurons are set once to some random values, and (2) the outputs are generated by a layer of linear neurons and (3) only the weights of these neurons are trained. The untrained non-linear part and the linear output layer are called *reservoir* and *readout*, respectively. The weights of the reservoir are first drawn from a normalized Gaussian distribution and are then properly re-scaled to obtain a stable dynamical system that offers a good balance between the impact of new inputs and that of past inputs as attributed by the recurrent connections.

Training of the output layer of an RCN boils down to minimizing the Mean Squared Error (MSE) between the actual network output and the desired output that follows from the presumed correct speech unit sequence, retrievable from the orthographic transcription of the speech. The envisioned output weights emerge as the solution of a set of linear equations. Compared to the iterative gradient-descent based training of conventional ANNs and considering that such training is not guaranteed to reach the global optimum, training of RCNs is remarkably advantageous.

Thanks to the easy and straightforward training procedure, it is possible to enlarge the size of the reservoir (i.e., number of nodes) to a level that is far beyond the scope of a conventional RNN (e.g., reservoirs with 32K nodes are easy to deal with). Hence, the reservoir creates a rich and high-dimensional state space that is suitable for distinguishing the speech units by means of a simple linear regression model. As the signal dynamics are represented in that space as well, and as the speech and the noise dynamics are bound to be different, the RCN may also be a very tractable approach for modeling noisy speech.

In this work, RCNs were (to my best knowledge) applied for the first time to acoustic modeling for noise robust continuous speech recognition. The aim of the work was to investigate continuous speech recognition with RCNs, especially, their behavior in the presence of background noise in the environment. Since the focus was on noise robustness, I opted for a small vocabulary continuous speech recognition (SVCSR) task, more particularly continuous digit recognition (CDR) on the well-known Aurora-2 dataset. Aurora-2 contains clean samples of continuous digits as well as many noisy versions thereof (different noise types and signal to noise ratios (SNRs)).

Since speech is a complex signal which shows a lot of variability related to external factors such as noise, speaker, accent or emotion, the dimensionality of the intermediate reservoir space had to be order of magnitude higher than what was employed before for other tasks. Hence, some work was needed to design a good strategy on how to develop and train such large RCNs, efficiently. In a later stage, the empirical rules to optimize the design parameters were converted to a more comprehensible procedure that fixes the reservoir parameters to quasi-optimal values in an automatic way. This research led to a novel analysis of an RCN as a non-linear dynamical system. This, in combination with some sensible heuristics allowed me to distill a number of empirical relations between the reservoir parameters and the RCN input and output dynamics. The resulting recipe is very efficient and at the same time very simple and highly comprehensible.

Instead of handling the full complexity of the speech signal in a single layer (with a very high dimensional reservoir state), one could also distribute this complexity to multiple layers leading to multi-layer RCN architectures. Each layer consisted of a basic RCN, including a reservoir and a linear output layer. The first RCN was stimulated by the acoustic input, the second one by the output of the first one, and so on. The networks were trained one at the time and no joint optimization of the cascaded network was performed. Cascading RCNs led to recognition improvements both in clean and in noisy conditions. I argued that the main reasons for this success are the following: (1) each RCN induces a new random projection and consequently reduces the chance of missing interesting non-linear projections (2) each RCN can correct errors observed in the outputs of the previous RCN, and (3) each RCN adds dynamical modeling capacity to the overall dynamical model. To further improve the systems I also introduced bi-directional reservoirs that have the capacity to consider both the left and the right context of an acoustic observation. Due to this, they can better handle co-articulation effects that are induced by the anticipation of upcoming speech sounds which the speakers intend to articulate.

Although RCNs are expected to introduce some level of noise robustness by themselves, I also investigated several strategies either to better cope with noise or to find a better balance between noise robustness and clean speech accuracy.

A known remedy to reduce the effect of noise on the performance, is to train the recognizer with various samples comprising different added noise types at different levels. However, even in this scenario, there is always a chance to face a new condition that has not been seen during training. A practical solution is to adapt the recognizer to a new condition with a limited effort. In this respect, I have shown that the unsupervised training (requiring no human intervention) of a small module that adapts the original readouts to noise specific readouts can improve the performance.

Where GMMs are known to perform very well if they are trained and operated on similar conditions, my RCNs only lead to superior performance in mismatched conditions. Finding a way of combining the advantages of both approaches was the next purpose of my research. I investigated two different strategies: likelihood fusion and a tandem approach. In likelihood fusion, a weighted sum of the likelihoods emerging from the GMM-based and the RCN-based acoustic models is fed as the likelihood to the decoder. In a tandem approach, a conventional GMM system is fed with the outputs of the RCN, or vice versa, an RCN-based system gets the outputs of the GMMs. My work showed that combining these two systems is beneficial if the training and testing conditions are similar, be it at the cost of a slight loss in performance in the mismatched situations.

Instead of performing a conventional noise-robust front-end such as the ETSI advanced front-end (AFE), I have also tested a simple front-end (pure MFCCs) followed by an RCN that aims at retrieving clean features from the noisy features at its inputs. Such a system is generally called a Denoising Auto-Encoder (DAE). Again, the fact that reservoir computing is able to capture temporal information, makes it attractive for removing complex patterns induced by the noise. My work shows that in the case of clean speech training, inserting a DAE after the MFCC front-end does reduce the average WER, but in the multi-style training case it causes an increase of the average WER. This can only mean that the DAE removes information from the feature stream that is otherwise exploitable by the RCN-based recognizer.

The final ASR I investigated had an RCN-based acoustic model composed of 3-layer bi-directional reservoirs with 8K nodes per layer, they were supplied with AFE features and trained on clean speech. They lead to an average WER (word error rate) over different noise types and noise levels of 9.0% on Aurora-2, compared to the 13.2% achieved by a state-of-the-art GMM-based system. When that system is trained on both clean and noisy samples, its average WER goes down to 5.8% compared to the 8.2% of the GMM-based ASR.

The previously described design recipes are fairly general and hence should be applicable to other pattern recognition problems rather than speech. To check this, I considered isolated handwritten digit recognition as another task and I benchmarked on the well-known MNIST dataset. Each sample of this dataset is encoded in 28×28 gray-scale pixels. I verified the noise robustness of the RCN based system using a noisy version of the MNIST dataset together with some published results obtained with Deep Belief Networks (DBNs) on these noisy versions. Similar to my research on noise robust speech recognition, I utilized an RCN in two ways, as a classifier (recognizer) and as a DAE (denoiser).

By following the aforementioned instruction to design the RCN-based recognizer and achieving the promising error rate of 0.81% on clean dataset, the main conclusion of my research was that the proposed strategy for optimizing reservoir parameters remains valid for handwritten recognition as well. When tested on noisy images (five noise types), the RCN-based system yields an average error rate of 23.6% compared to 28.8% of the reference DBN. The RCN-based system trained on clean as well as noisy images yields a WER of 1.5% and 3.5% on clean and noisy samples, respectively.

Adding an RCN-based DAE which was trained to remove five different noise types from the images caused a dramatic drop of the WER of the clean trained classifier to only 2.1%. The reported WER for a DBN-based system on the same dataset was 2.3%.

In conclusion, my work shows that RCN-based acoustic modeling can be incorporated in the statistical framework of a noise robust continuous digit speech recognizer. The presented models reach state-of-the-art performance in clean conditions and advance the state-of-the-art for noisy situations. My work also shows that there exists a simple but effective strategy for optimizing the reservoir parameters on the basis of some easy-to-gather knowledge of the system input and output dynamics. This strategy seems to work for both speech and handwritten digit recognition. Apparently, the idea of randomly fixing and recurrently connecting a pool of non-linear neurons is a good basis for modeling complex temporal patterns (in speech recognition) or complex local relations between adjacent pixels (in image recognition). Given that research on reservoir based systems for such complex tasks is still young and underdeveloped, there is still enough room for exploring new ideas and for making new progress.

Thesis Format

This dissertation is written in a 'thesis by publications' format. Chapters 3 – 8 represent distinct pieces of the overall doctoral research that have been published as original research work in the proceedings of a peer-reviewed international conference or in a peer-reviewed international journal. The original publications are to some extent edited to ensure a coherent dissertation with consistent formatting. Each of these chapters is also expanded with a preface to provide a fluent transition between the chapters. This thesis format reveals the progress made during the research.

The main body of the work (Chapters 3 – 8) is surrounded by a comprehensive introduction and a conclusive discussion. The introduction (Chapters 1 and 2) focuses on the global objective of the dissertation, the underlying research background, and highlights the scope and main results of the individual publications and their interdependency. The conclusive discussion (Chapter 9) contains additional insight into collaborative research, the conclusions drawn from the conducted research, and directions for future work. Appendix A lists, in detail, the performances (on Aurora-2) of the main continuous digit recognition systems developed during this work.

A list of all publications the candidate was involved in during his doctoral research is given on page xxxvii.

Acknowledgments: The research leading to the results presented in this dissertation received funding from: (1) the European Community's Seventh Framework Program (FP7) project ”Self-organized recurrent neural learning of language processing” (ORGANIC) under grant agreement 231267 and (2) the Research Foundation Flanders (FWO) project ”Reservoir Computing for Auditive Pattern Recognition” (RECAP) under grant agreement G.0088.09N.

Statement of original authorship: The work contained in this thesis has not been previously submitted for a degree or diploma at any other higher education institutions. This thesis contains no material previously published or submitted for publication by another person except where due reference has been made.

Publications by Candidate

[1] A. Jalalvand, F. Triefenbach, D. Verstraeten, and J.-P. Martens, "Connected digit recognition by means of reservoir computing," *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1725–1728, 2011.

[2] A. Jalalvand, F. Triefenbach, and J.-P. Martens, "Continuous digit recognition in noise: Reservoirs can do an excellent job!," *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, p. ID:644, 2012.

[3] A. Jalalvand, F. Triefenbach, K. Demuynck, and J.-P. Martens, "Robust continuous digit recognition using reservoir computing," *Computer Speech and Language (CSL)*, vol. 30, no. 1, pp. 135 – 158, 2015.

[4] A. Jalalvand, K. Demuynck, and J.-P. Martens, "Noise robust continuous digit recognition with reservoir-based acoustic models," *Proceedings of the International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, p. ID:99, 2013.

[5] A. Jalalvand, K. Demuynck, and J.-P. Martens, "Feature enhancement with a reservoir-based denoising auto encoder," *Proceedings of the International Symposium on Signal Processing and Information Technology (ISSPIT)*, p. 6, 2013.

[6] A. Jalalvand, W. De Neve, R. Van de Walle, and J.-P. Martens, "Noise robust handwritten digit recognition with reservoir computing networks," *Journal of Machine Learning Research*, submitted, 2014.

[7] A. Jalalvand, G. Van Wallendael, and R. Van de Walle, "Reservoir computing networks for detection tasks on (motion) pictures," *IAPR Intl. Conference on Machine Vision Applications*, submitted, 2015.

[8] F. Triefenbach, A. Jalalvand, B. Schrauwen, and J.-P. Martens, "Phoneme recognition with large hierarchical reservoirs," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 2307–2315, 2010.

[9] F. Triefenbach, A. Jalalvand, K. Demuynck, and J.-P. Martens, "Acous-

tic modeling with hierarchical reservoirs," *IEEE Transaction on Audio, Speech and Language Processing*, vol. 21, pp. 2439–2450, November 2013.

[10] F. Triefenbach, A. Jalalvand, K. Demuynck, and J.-P. Martens, "Context-dependent modeling and speaker normalization applied to reservoir-based phone recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3342–3346, 2013.

Part I

Introduction

Preface

Conversation is one of the most attractive ways of communicating with other human beings. Through conversation we can express ourselves in an intuitive and effective way, whether this involves simple instructions or complex explanations. The history of human-machine communication via speech dates back to more than 1000 years ago, when Pope Sylvester II was supposed to have built a brazen head. This 'robotic' head would answer his questions with "yes" or "no". Ever since, the evolution has continued and these days we use various speech-based communication modules in many electronic devices such as TVs, smartphones, navigators, etc. However, those who frequently use this voice-based human-machine communication do not always find it sufficiently reliable, and definitely not as efficient as human-human communication.

The human-machine interaction via speech has some components and in this dissertation, I will focus on one of these components, namely Automatic Speech Recognition (ASR), and more precisely, the acoustic modeling component of a speech recognizer. The aim of the acoustic model is to handle the relation between the acoustics produced by the vocal tract and their linguistic interpretation. While there are many approaches to handle this relation, none of them seems to be satisfying enough. Each one has its own pro's and cons. For instance, some perform well in very constrained environments and some are computationally expensive. The aim of this research is to investigate Reservoir Computing Networks (RCN) as an alternative approach. It is argued that an RCN can more efficiently exploit the dynamic properties of speech than the conventional approaches. Furthermore, the RCN is believed to be more robust against the presence of background noise.

The first part of this thesis provides the necessary general background about ASR and the architecture that is conceived to achieve it. The background is reviewed in two chapters. Chapter 1 provides a general overview of ASR and addresses some of the encountered difficulties. It also provides a concrete outline of the rest of the thesis and it tells the main story line of my research. In Chapter 2, the actual research focus, namely acoustic modeling, is presented in more depth. I will review the most popular approaches for modeling the speech acoustics, namely Gaussian Mixture Models and various types of Artificial Neural Networks, and I will assess the most recent findings reported in literature. Moreover, I will introduce Reservoir Computing Networks as a special type of Neural Network and I will argue why they may, in time, compete or even surpass the current acoustic modeling approaches.

1
Problem, Motivation and Formulation

1.0 Human-Machine Interaction by Speech

Human-machine interaction (HMI) is a major research area in the field of Artificial Intelligence (AI) and robotics. Among the different ways of interacting with machines, communication through speech is one of the most appealing, but at the same time, one of the most challenging ones. In this respect, the aim is to develop a machine that can listen and respond to users by means of naturally sounding speech.

Despite the fact that we usually find it effortless to understand other people when they speak, especially when the conversation is in our mother tongue language, parsing the speech stream is an impressive perceptual masterpiece. Our ability to instantaneously decode speech, which is a highly complex acoustic signal, into a sequence of sounds and to group those sounds into words, sentences and semantic concepts is remarkable. Figure 1.1 shows the diagram of speech-based human-machine communication. The first task of the machine is to listen to the speech and to *recognize* the words that were spoken. In the literature, this procedure is called Speech-to-Text (S2T) conversion or *Automatic Speech Recognition* (ASR) [11]. The next step is to *understand* the meaning of what has been spoken [12]. Then the machine *generates* a reasonable response [13] as a sequence of words and sentences. Finally, these words are converted into a speech signal. The last step is usually called *speech synthesis* [14].

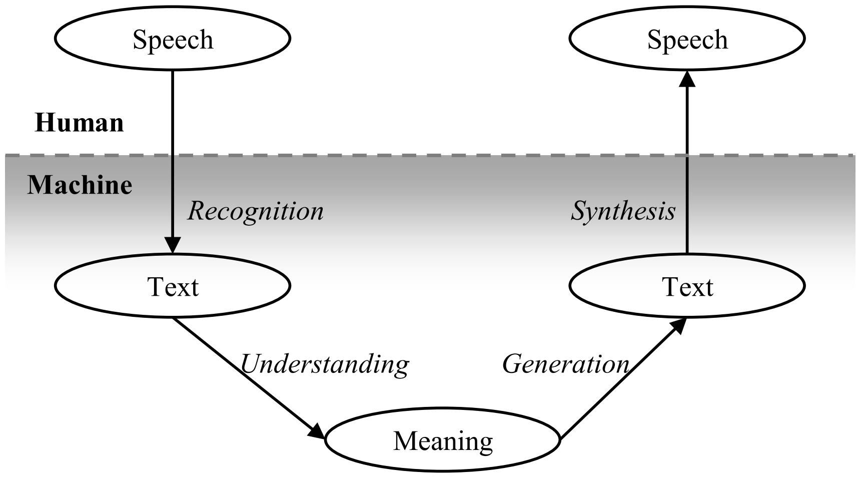


Figure 1.1: A standard diagram of speech-based human machine communication

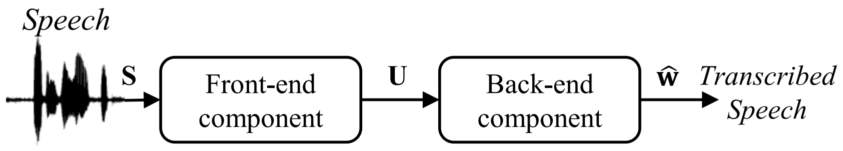


Figure 1.2: High-level diagram of a speech recognizer

The concern of my work is the first step, namely speech-to-text. This chapter offers a brief introduction to modern speech recognition systems and their limitations. Subsequently, I review some state-of-the-art remedies that have been devised to tackle those limitations.

1.1 Automatic Speech Recognition

From a general perspective, the human speech recognition system comprises two major blocks corresponding to the ear(s) and the auditory centers in the brain. The ears are responsible for capturing the acoustic signal and for extracting acoustic features from that signal. These acoustic features are processed in the brain with the aim of separating speech from non-speech parts, detecting the language and recognizing the spoken words.

Likewise, a standard ASR system, contains two major components: a front-end and a back-end. A diagram of the ASR components is shown in Figure 1.2.

The front-end component, like the ears, extracts acoustic features from the speech signal. At a given time, the features are collected in a vector U_t. They usually represent the short-time energy and spectrum of the signal [15]. One of the most well-known acoustic features used for speech recognition are the Mel Frequency Cepstral Coefficients [16]. Acoustic features vary at the rate the human articulators (tongue, lips, velum, etc.) can move, hence, they need only to be computed at a rate of one vector per 10 ms.

The back-end component (like the brain) *decodes* the acoustic feature stream and returns a sequence of words $\hat{\mathbf{w}}$ that fits best with the feature vectors. One can simply translate the task of the back-end to the following equation:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \, P(\mathbf{w}|\mathbf{U}), \tag{1.1}$$

where, $\mathbf{U} = [U_1, \cdots, U_t, \cdots, U_T]$.

Models of Speech

The complexity of the solution to the above equation strongly depends on the kind of speech the recognizer is expected to recognize. In this section, I briefly review the different types of ASR systems that have been devised in order to deal with different types of speech.

Suppose that we want to control a robot vacuum cleaner with speech. Such a system is supposed to respond to a small set of predefined commands such as “left”, “right”, “stop”, “turn”, etc. That means that a small vocabulary isolated word recognizer (SVIWR) can do the job. Such a recognizer encompasses a so-called acoustic model for each word. It represents the ensemble of acoustic feature streams that can be observed when that word is spoken. The solution to the basic equation is then obtained by examining which model best explains the acoustic feature stream.

Entering a PIN-code into an ATM machine, on the other hand, requires entering a fixed or variable length of digits sequence. To recognize such a sequence, one needs a small vocabulary continuous speech recognizer (SVCSR). In order to solve the basic equation now, one can still use word models like in the SVIWR, but one then needs to factorize the probability of a word sequence as a product of word-level probabilities in order to find the sequence best explaining the acoustic feature stream that needs to be recognized.

Entering an address into a navigation system or dictating a letter are examples in which the speech represents a continuous sequence of very diverse words. Recognizing such a sequence takes a large vocabulary continuous speech recognizer (LVCSR). Since the vocabulary can encompass

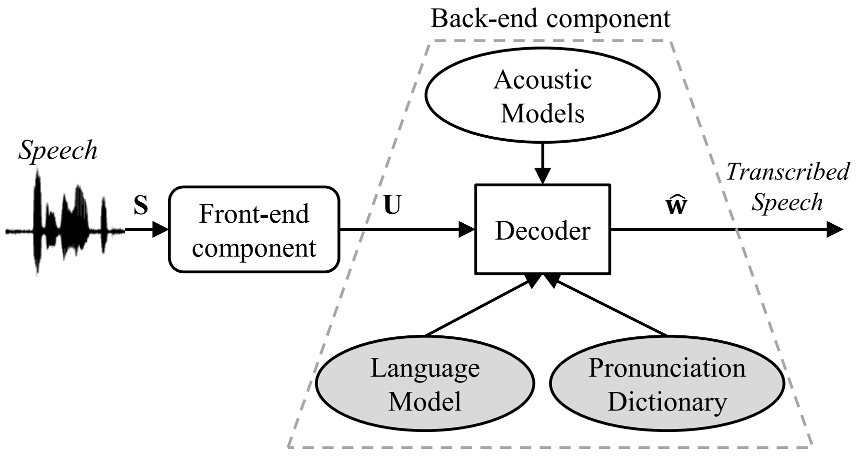


Figure 1.3: Diagram of an ASR. The gray components are usually used for LVCSR, only.

a few hundred thousand words in such a case, one can no longer construct separate acoustic models for each word.

The solution then is (1) to decompose words into smaller units that belong to a finite generative set of what we call basic phonetic units, such as syllables or phonemes, and (2) to construct acoustic models for these phonetic units. The idea is then to factorize the probability of a word sequence into a product of phonetic unit level probabilities and to search for the word sequence maximizing this product. In order to achieve that, one has to create a *Pronunciation Dictionary* comprising all the words of the vocabulary, together with their pronunciations, defined as their decomposition into basic phonetic units. For instance, the word “left” could be represented by its pronunciation “L-EH-F-T”. Given the large number of words and given the fact that each word of a sentence is, to some extend, predictable from the preceding words, an LVCSR also encompasses a so-called *Language Model* that assigns prior probabilities to word sequences and uses these probabilities to constrain the search space.

In what follows, the ensemble of unit acoustic models is usually denoted by the term *Acoustic Model*, abbreviated to AM. Figure 1.3 shows the diagram of an LVCSR in more detail.

While the type of features extracted in the front-end is usually independent of the language and the recognition mode (e.g., small or large vocabulary), some of the components in the back-end, such as the acoustic model and the language model are fine-tuned to the recognition task at hand. In practice, the acoustic model and the language model are stochastic models

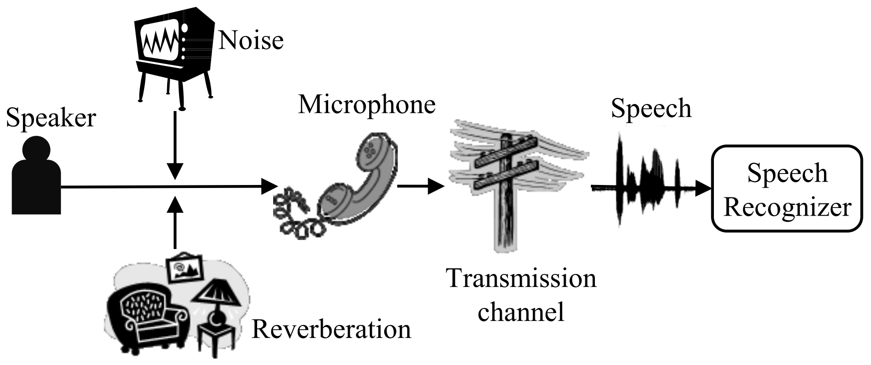


Figure 1.4: Distortions in adverse environment.

that are learned from training material (e.g., labeled speech corpora and text material respectively). This learning phase usually takes place long before the system is put into operation. This is again comparable to what happens in speech recognition by humans. The front-end (ear) extracts the same type of acoustic features regardless of the language and the environment. While, our brain uses task specific processes, that were learned before, to extract the various types of information encoded in an acoustic signal (detecting noise and speech, identifying the speaker and language, decoding words, etc.).

The trained stochastic models will only work well if they are designed according to well founded procedures, and if the training material was sufficiently large and well representing the situations the system is going to experience during operation. In this respect, my work is mainly about reducing, as much as possible, the performance degradation that is due to the mismatch between the circumstances during training and operation.

1.2 Noise Robustness in ASR

Speech signals carrying an identical linguistic message still show a large amount of variability. This variability is governed by different factors:

- *Variability due to the environment*: The acoustic environment, in which the speech is recorded, modifies the speech signals. Examples thereof are background noise, reverberation, microphone distortion, and transmission channel distortion. Figure 1.4 shows the typical sources of distortion in adverse environments.

  – *Reverberation*: In an enclosed environment such as a room, the acoustic speech wave is reflected by objects and surfaces. These (delayed) reflections are added to the signal captured by the microphone.

  – *Different microphones*: It is often the case that a recognizer is trained using speech that was captured with a high quality close-talking microphone, while in the real world, the applications are sometimes forced to work with cheaper microphones that have different frequency response characteristics.
  – *Transmission channels - telephone*: A great deal of applications need telephone speech recognition. However, due to the narrow bandwidth ($300 \sim 3400$ Hz) and non-linear distortion in the transmission channels, telephone speech is much more difficult to decode than full bandwidth speech.

- *Variability due to the speakers*: Since no two persons share identical vocal cords and vocal tract, they cannot produce the same acoustic signals. Typically, females sound different from males and children sound different from adults. Also, there is variability due to dialect and foreign accent. A word may also be uttered differently by the same speaker because of illness or emotion. It may be articulated differently depending on whether it appears in planned read speech or spontaneous conversational speech. Even speech produced in the presence of noise differs from speech produced in a quiet environment because the speaker modifies its speech to sound more clearly, a phenomenon called *Lombard effect* [17].

The variability caused by speakers is generally considered as *genuine*, whereas, the one that is caused by changes in the channel and the environment are referred to as *noise*.

The noise brings two problems: (1) noise adds to the variability and therefore, it may lead to more confusion between words, (2) there can be differences between the noise present during operation and the noise that was present during training of the AM. This causes a so-called mismatch between training and testing and forces the AM to make extrapolations.

Remedies

Problems arising from noise can be addressed at various stages of the recognition process. I will discuss some of the most frequently used ones in this section:

- *Robust feature extraction*: The idea here is to extract features that are affected as less as possible by the kind of noises that can be present during operation. For example, RASTA [18] feature extraction uses filters to remove those components of the signal which do not follow the dynamics of the speech. Therefore, the features are less affected by stationary noises than standard PLP [19] or MFCC features. Similarly, phase-autocorrelation (PAC) features [20] are able to enhance the peaks of the spectrum and were shown to be less affected by additive noise.

  Each of these feature representations tries to preserve as much as possible the characteristics of the (clean) speech in the features. However, a single feature representation might not perform well under all conditions.

  - *Problem*: The signal transformation used to extract robust features need to balance their need to reduce unwanted variability and the need to retain as much relevant information as possible. In practice, most robust features introduce some information loss, and hence, they do not describe clean speech as accurately as standard features.

- *Feature enhancement*: Feature enhancement [21–24] is a popular approach to reduce the effect of additive uncorrelated noise on the extracted features. In this approach, the effect of noise is removed from the features.

  - *Problem*: usually, feature enhancement is based on simplified assumptions (e.g., uncorrelated features or speech and noise uncorrelated) which may not entirely hold. Furthermore, reconstructing a speech signal which is completely suppressed by the noise is simply not possible.

- *Feature normalization*: Normalization techniques like cepstral mean and variance normalization (MVN) [25, 26] and histogram normalization [27,28] can help to reduce the mismatch between the statistics of the features collected in different circumstances.

  - *Problem*: These approaches normalize the marginal distribution (feature distributions irrespective of the linguistic content) based on long-term averages only. This requires a fair amount of speech, otherwise, the differences between the words (the linguistic content) can be normalized away. Furthermore, identical marginal distributions do not guarantee identical distributions.

- *Multi-style training*: Models can be made robust by training them on different conditions (various noise types and levels [29] and/or different speaking styles [30]).
  - *Problem*: not all the noise types might be present during training and the presence of noise can broaden the distributions, causing the recognition to be sub-optimal in good conditions.
- *Missing data approach*: In this approach, it is assumed that noise affects only some regions in the spectro-temporal plane and it is possible to identify such regions. Some features are then treated as unavailable or unreliable and are suggested to be replaced by more appropriate features. Two popular techniques to achieve this are marginalization and deriving the features from the neighboring features [31].
  - *Problem*: The approach is hindered by the fact that it is difficult to detect the reliable and unreliable parts with high accuracy in the presence of noise.
- *MLLR*: Maximum-Likelihood Linear Regression (MLLR) [32, 33] and its variant Constrained MLLR (CMLLR) [34] are adaptation techniques that try to adapt the parameters of the AM by means of an affine transform. Compared to the normalization techniques, MLLR adjusts the complete distribution and not only the marginal distributions.

  If the acoustic model is based on n Gaussian distributions (see Section 2.2) the parameters of the AM are the mean vectors $\mu_{1\cdots n}$ and the diagonal covariance matrices $\mathbf{\Sigma}_{1\cdots n}$ of the distributions. Therefore, in conventional MLLR, μ and $\mathbf{\Sigma}$ are updated according to the different transformations:

  $$\hat{\mu} = \mathbf{A}\mu + b$$

  $$\hat{\mathbf{\Sigma}} = \mathbf{H}\mathbf{\Sigma}\mathbf{H}^T$$

  where $\mathbf{A}$ and b are the transformation matrix and bias for the mean vector and $\mathbf{H}$ is the transformation matrix for the covariance matrix. In order to reduce the number of transformation parameters, one has introduced CMLLR. It assumes a shared transformation matrix $\mathbf{C}$ for both means and covariances:

$$\hat{\mu} = \mathbf{C}\mu + b$$
$$\hat{\mathbf{\Sigma}} = \mathbf{C}\mathbf{\Sigma}\mathbf{C}^T$$

  - *Problem*: Collecting enough reliable adaptation data is an issue [35]. Furthermore, if the acoustic conditions of the environment are not stationary (change over time) the transformation matrices must change, as well.

- *Uncertainty decoding*: An appealing approach to robust recognition, called uncertainty decoding (UD), is proposed in [36, 37]. In UD, there is a model for estimating the amount of uncertainty about the features. The uncertainty is calculated efficiently in the front-end, and passed to the back-end as a single, simple variance offset to the acoustic model components. This can provide an elegant compromise of a fast feature-based compensation scheme with promising accuracy.

  - *Problem*: In some very low signal-to-noise ratios where noise masks the speech, the uncertainty models are acoustically indistinguishable, which can cause spurious errors [36]. Furthermore, propagating the uncertainty requires several approximations in the feature extraction and acoustic modeling to make the mathematics tractable.

1.3 The Aim of This Work

Various techniques have been utilized to model the acoustics of the speech units. Gaussian Mixture Models (GMM) and different types of Artificial Neural Networks (ANN) are the dominant ones in state-of-the-art ASRs. The common property of these models is that their outputs at time t are based on short-term acoustic representations of the speech in an interval centered around t. However, it is argued that one needs a larger scope to separate the speech from the noise dynamics. In this respect, the main objective of my research was to investigate a new neural network type (new in the domain of speech), called reservoir computing networks (RCN), that are recursive in nature and thus expose a larger context. Therefore, I believe that they are inherently more robust than their competitors against the various sources of variability mentioned before.

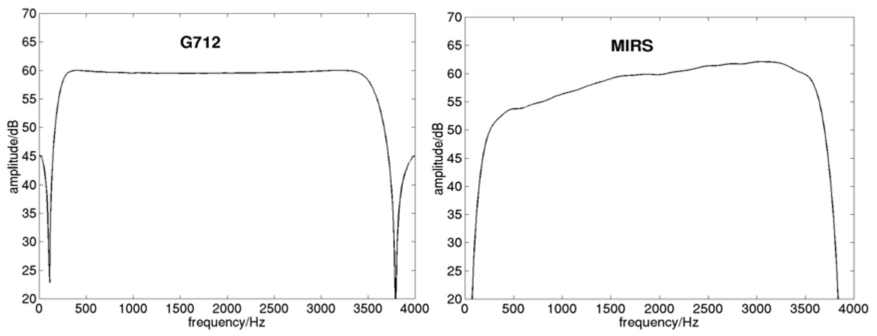


Figure 1.5: Frequency responses of G.712 and MIRS filter [29]

As I explained earlier, the recognition of speech involves a language model and a pronunciation dictionary in addition to the AM. However, noise sensitivity mainly originates from the acoustic features and the AM. Therefore, in order to concentrate on that aspect and to avoid the interference of the other components when measuring the noise sensitivity of the AM, I chose to study noise robustness in a SVCSR task, namely continuous digit recognition (CDR). Since in such a task any digit can follow any other digit, no language model is needed and due to the restriction of the dictionary no pronunciation dictionary is needed, either. More precisely, I will use the Aurora-2 spoken digit benchmark that is briefly introduced in the next section.

1.4 The Aurora-2 Framework

The Aurora-2 experimental framework [29] is designed specifically to bolster research on the noise robust continuous digit recognition. The Aurora-2 corpus consists of clean and noise corrupted digit sequences counting 1 to 7 digits per utterance. The original 20 kHz data have been downsampled to 8 kHz with an “ideal” low-pass filter extracting the spectrum between 0 and 4 kHz. These data are considered as “clean” data and distortions are artificially added to these clean data.

An additional filtering is applied to consider the realistic frequency characteristics of terminals and equipment in the telecommunication area. Two “standard” frequency characteristics are used which have been defined by ITU [38]. The abbreviations G.712 and MIRS have been introduced as reference to these filters. Their frequency responses are shown in Figure 1.5. The major difference is a flat curve of the G.712 characteristic in the range between 300 and 3400 Hz where the MIRS shows a rising characteristic

	Train & Test A	Test B	Test C
Noise types	subway babble car noise exhibition hall	restaurant street airport train station	subway street
Filter	G712	G712	MIRS

Table 1.1: Noise types and filters used in different Aurora-2 sets

with an attenuation of lower frequencies. MIRS can be seen as a frequency characteristic that simulates the behavior of a mobile telecommunication terminal.

Since there are two variants of ‘0’ in American English, namely *zero* and *oh*, the vocabulary is composed of 11 words.

The data is divided into a training part and a testing part. The framework supports two types of experiments: clean training experiments in which systems are developed on 8440 clean training utterances from 110 adults and multi-style training experiments in which systems are developed on 8440 noise corrupted versions of the same utterances. The corruption is randomly chosen out of four noise types and five signal-to-noise ratios (SNRs): ∞ (clean), 20, 15, 10 and 5 dB. The evaluation utterances come from speakers that are not present in the training data. The utterances are divided into three tests. Tests A and B each contain 28,028 utterances covering 4004 different digit sequences, 4 noise types and 7 SNRs (∞ (clean), 20, 15, 10, 5, 0, and -5 dB). The noise types occurring in Test B do not occur in the multi-style training data, while those of Test A do. Test C contains 14,014 utterances covering 2002 different digit sequences, 2 noise types (one matched and one mismatched) and 7 SNRs. The utterances of the training set as well as Test A and B were passed through a G712 filter, whereas those of Test C were passed through a MIRS filter (see Table 1.1).

1.5 Outline

In this thesis, the issue of robustness in the context of continuous digit recognition (CDR) and reservoir computing networks (RCNs) is investigated as an approach to improve the robustness of ASR. Many of the techniques already mentioned in this thesis (such as feature normalization and spectral subtraction) which help to make the acoustic feature representations more robust, can be used in conjunction with RCNs.

The remainder of this document reviews, in chronological order, my work on noise robust continuous digit recognition.

When I started my work, RCNs had already been applied for clean isolated digit recognition but the published studies only reported small experiments. Therefore, I decided to build an RCN-based CDR system considering the established knowledge about reservoirs and HMMs and to evaluate its robustness on the Aurora-2 benchmark. After some naive attempts which failed to achieve noise robustness with RCNs and standard acoustic features, I started to employ robust features, like many state-of-the-art systems do. With such features, I finally succeeded to conceive an RCN-based system that degrades more slowly with an increasing noise level than a conventional GMM-based system. That work was published as an *Interspeech 2011* conference paper and is handled in Chapter 3 of this thesis.

Although the created system outperforms GMMs in mismatched (noisy) conditions, it did not perform so well on clean speech. Improving the reservoir performance for high SNRs was the topic of my *Interspeech 2012* paper, which forms the basis for Chapter 4. In that work, I proposed to employ a multilayer RCN (where each layer is supplied with the outputs of the previous layer) and I investigated three unsupervised ways to adapt a trained RCN to unseen conditions. The adaptations are called unsupervised, because they do not require any manual labeling of the adaptation data.

The above investigations helped me to understand how an RCN should be optimized for speech recognition. In practice, I learned how to fix the reservoir parameters in order to achieve high performance. This resulted in a set of heuristics to find appropriate values for the hyper-parameters, defined as the parameters of the random processes that fix the reservoir parameters. However, even with these heuristics, there is still a whole range of hyper-parameters combinations to examine. This stimulated me to further investigate the relations between hyper-parameters and to translate them into a minimum set of principles/rules that should be employed to get good hyper-parameters. This resulted in a 'theory' that leads to a hyper-parameter set that is quasi-optimal, but derivable from easy-to-gather prior knowledge and statistical information gathered by small-scale experiments.

As this theory alleviates the time-consuming grid searches in the hyper-parameter space we were forced to conduct otherwise, it significantly sped up the experiments. In a short time, I could develop and evaluate several new architectures and establish that I can achieve state-of-the-art CDR accuracy in clean conditions and better robustness against noise than much more complicated systems. The results of this work are elaborated in an article that appears in the *Computer, Speech and Language* journal and constitutes Chapter 5 in this thesis.

My work shows that the performance of RCN-based systems degrades less quickly than conventional systems when the mismatch between training and test condition increases. On the other hand, the latter systems seem to be better equipped to learn the characteristics of a particular condition, leading to excellent performance in matched conditions. Therefore, I searched for sensible approaches to combine the advantages of both system types. Chapter 6 describes three techniques that attempt to accomplish this. This work was published and presented at the *ISPACS 2013* conference.

Instead of improving the acoustic model, one can also apply RCNs to enhance the features. To that end, I trained an RCN to act like a Denoising Auto-Encoder (DAE) that maps the noisy features to 'clean' copies. Although one can train a DAE for each condition, it is more appealing to build one DAE for all conditions and to use the outputs of that DAE as the inputs to the recognizer. The results of this work was presented at the *ISSPIT 2013* conference and are described in chapter 7 of this dissertation.

The main objective of my research was clearly to show the potential of RCN in robust speech recognition. However, I was also curious to find out whether the proposed strategy of designing RCN is, to a large extent, transferable to a completely different task. Chapter 8 of this thesis appertains to this curiosity. It describes how I developed an RCN-based system for handwritten digit recognition. Here too, I focused on the robustness of the system to the presence of noise in images. I performed experiments on the renowned MNIST dataset and showed that RCN is indeed a serious alternative to the state-of-the-art noise robust image recognizers. This work has been submitted to the *Journal of Machine Learning Research (JMLR).*

2 Acoustic Modeling

2.0 Introduction

As a result of extensive research, ASR has improved significantly over the last three decades. The progress in both hardware (e.g., more powerful computers) and software (better training algorithms) have been key reasons to this improvement. Besides this, ASR has become an attractive reference problem for testing novel methods developed in other research fields such as Machine Learning and Cognitive Science [39].

However, in spite of the achieved improvements, state-of-the-art ASRs are still far from being competitive with the incredible human speech recognition system. Partly, this is also the reason why there is hesitation in deploying these not so well-performing ASRs in human-machine interaction.

Considering the fact that most of the human speech processing is done in the brain, it is not surprising to see that most ASR research is devoted to the back-end, and particularly to the acoustic model (AM) it contains.

The aim of this chapter is to give the reader an overview of the most popular and powerful approaches that were adopted for acoustic modeling. Next, I introduce the Reservoir Computing Network (RCN) as an interesting new type of neural network and I argue why it might resolve some of the limitations of the current modeling approaches.

Obviously, a discussion of the AM is not complete if it is not also considering the role of the AM in the whole system and its interactions with the other parts of ASR. Therefore, I begin with some more detail about the

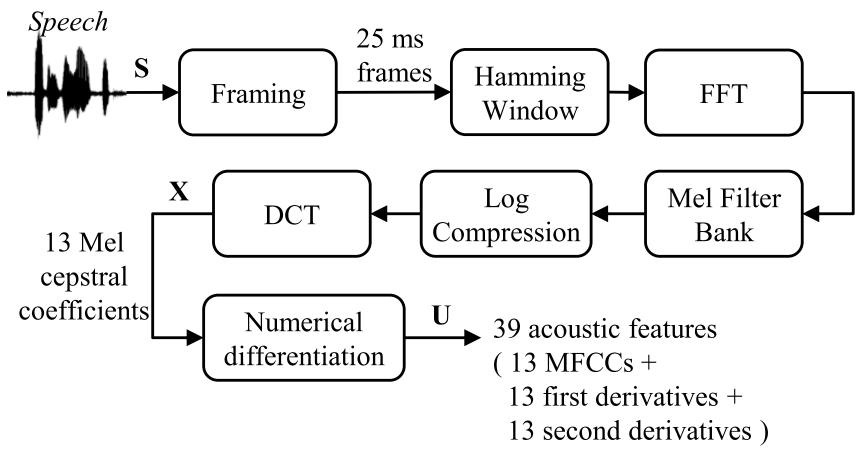

Speech
S
Framing
25 ms frames
Hamming Window
FFT
Mel Filter Bank
Log Compression
DCT
X
13 Mel cepstral coefficients
Numerical differentiation
U
39 acoustic features (13 MFCCs + 13 first derivatives + 13 second derivatives)

Figure 2.1: Extracting MFCC components

components of a speech recognition system.

2.1 Components of a Speech Recognition System

2.1.1 The Front-End

The aim of the front-end is to extract features that still contain all the information related to the linguistic content of the signal, while at the same time simplify the signal and suppress unwanted variability in the signal (e.g., background noise). A set of desirable features is supposed to offer (1) discrimination between the basic speech units, (2) invariance to unwanted sources of variability such as speaker identity, channel properties and the presence of background noise.

Although it is theoretically possible to recognize speech directly from a digitized waveform, almost all ASR systems perform some spectral transformation on the speech signal. This is because numerous physiological and psychological experiments have shown that the inner ear acts like a spectral analyzer [40].

The acoustic features most widely used in ASR are Mel-Frequency Cepstral Coefficients (MFCCs) and Perceptual Linear Prediction (PLP) coefficients. There are however plenty of other options, such as Linear Predictive Coding (LPC), Power Normalized Cepstrum Coefficients (PNCCs), etc. In the present work, I mainly use MFCCs as acoustical features. The process of extracting them consists of the following steps (see also Figure 2.1):

1. *Framing*: Speech is a time-varying signal and is (quasi) stationary for

a short time only. This calls for a subdivision of the signal into short blocks of samples, called frames. Typically, such a block is 20-35 ms long. The shift between adjacent blocks is typically 10 ms, meaning that there is overlap between consecutive frames.

2. *Windowing*: Framing in the time domain corresponds to multiplying the signal with a rectangular window. Since the subsequent discrete Fourier transformation (see Figure 2.1) also analyses the signal transition from the end of the frame back to the beginning of the frame (periodic extension of the signal) the cutting at the edges of the window introduces unwanted discontinuities which introduce artifacts. These artifacts can be suppressed by replacing the rectangular with a non-rectangular framing function that tapers off at the edges. The Hamming window [41] is a common choice for speech signal processing.

3. *Static Feature Computation*: In order to extract the acoustic features, the human ear makes a constant-Q analysis [42]. However, due to the varying number of samples needed for calculating the outputs at each frequency, implementation of this transform is tricky. Instead, many ASRs employ an approximation of such an analysis.

   A front-end method that produces MFCCs [43] proceeds as follows. First, it calculates a power spectrum of each frame by means of an FFT which can be interpreted as a filterbank with fixed bandwidth band-pass filters. Then a set of center frequencies, equidistantly spaced on a frequency scale that is close to a log-frequency scale, is chosen. Based on these center frequencies, triangular filter weights (presenting the filter gains of the constant-Q filters) are designed and a weighted sum of power contributions emerging from the FFT is computed for each center frequency. The outputs of this stage approximate the energies that would have emerged from a Mel filterbank, composed of band-pass filters with bandwidths that are proportional to the center frequencies of these filters. Such a filterbank much better explains the spectral analysis in the human ear than the fixed-band filterbank of an FFT. In a third step, the filterbank energies are compressed by taking their logarithm. This is also motivated by the fact that the perceived loudness of a sound is not proportional to its energy but to its log-energy. The final step computes the DCT of the log filterbank energies. The motivation for this is that adjacent filters of the filterbank are overlapping in frequency, and consequently, their output energies are quite correlated. The DCT decorrelates the filterbank energies, resulting in a compact set of decorrelated acoustic features. This decorrelation leads to a less expensive training procedure of the GMM-based acoustic models.

Notwithstanding their widespread use, MFCCs are sub-optimal. MFCCs give equal weight to both the spectral peaks and the spectral dips. The human perception, on the other hand, uses the fact that spectral peaks are less affected by noise, and hence focus more on spectral peaks.

4. *Dynamic Features*: Most of the acoustic models compute probabilities for a signal (converted to a sequence of frames by the feature extraction) as a product of frame-wise probabilities meaning that they assume independency between frames. Obviously, subsequent frames are correlated. In order to take this correlation into account as well, one can supplement the static features with dynamic features.

   Dynamic features are retrieved from the static features by taking first and second order time derivatives of each static feature [44,45]. The delta ($\Delta x_{t,l}$) and double delta ($\Delta^2 x_{t,l}$) cepstral coefficients are the first and second order time derivatives of the cepstral coefficients ($x_{t,l}$) respectively, and are obtained by,

$$\Delta x_{t,l} = \frac{\sum_{k=-i}^{k=i} k \cdot x_{t+k,l}}{\sum_{k=-i}^{k=i} |k|} \quad , \quad \Delta^2 x_{t,l} = \frac{\sum_{k=-i}^{k=i} k \cdot \Delta x_{t+k,l}}{\sum_{k=-i}^{k=i} |k|} \quad .$$

   where t is the frame index, l is the feature component index and i determines the context length. The latter is generally kept between 2 and 4 to have a context of 5 to 9 frames [45].

Altogether, the output of the front-end component at time t is an MFCC acoustic feature vector U_t containing the log-energy and 12 cepstral coefficients along with their first and second derivatives:

$$U_t = (\log E_t, X_t, \Delta \log E_t, \Delta X_t, \Delta^2 \log E_t, \Delta^2 X_t).$$

2.1.2 The Back-End

As explained in Chapter 1, the back-end of a continuous digit recognizer contains two main components, an acoustic model (AM) and a decoder.

2.1.2.1 Acoustic Models

Just as written words consist of a sequence of letters, spoken words consist of a sequence of basic sounds. In order to model this, a finite state linear automaton is created for each word, expressing the fact that the word can be split into a sequence of subsequent units, called states (e.g., three states for the beginning, the middle, and the end part of the word). Figure 2.2 shows

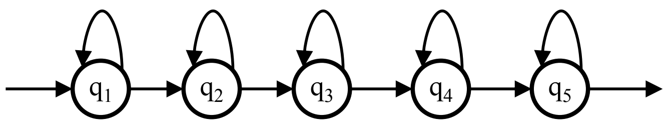


Figure 2.2: HMM automaton with five states for a word.

an example of an automaton for a word with five acoustic states. The active state of the automaton progresses through time. At each time step, a transition between states occurs and the visited state ‘emits’ an acoustic vector. The transitions between states are governed by so-called transition probabilities. The emission of an acoustic vector in a state is governed by a so-called emission distribution. The latter models how the feature vectors, observed in a particular state, are distributed. During recognition, this distribution is used to compute how likely an observed feature vector is for a given state. The emission distributions can be modeled in various ways, but the most prominent ones are: (1) a Gaussian Mixture Model (GMM) per state that computes the likelihoods of that state directly, and (2) an Artificial Neural Network (ANN) that computes the posterior probabilities of all the states followed by Bayes’ law that converts these state posteriors to state likelihoods. In any case, the models will only enable good recognition if the transition probabilities and the parameters of the emission model have been given appropriate values. These values emerge from an automatic training procedure that just needs orthographically transcribed speech utterances.

Due to the probabilistic nature of the transitions and the emissions, the automaton becomes a Hidden Markov Model (HMM). This automaton can so-to-speak generate acoustic vector sequences $\mathbf{U}$ of an arbitrary length not smaller than the number of states it contains. For a simple task such as SVIWR, the task of a word recognizer then boils down to identifying the model that is most likely to have generated the observed acoustic sequence $\mathbf{U}$. Since words can be surrounded by silences, the recognizer must also encompass a silence model, which can be a single state HMM. In practice, the recognizer searches for the most likely state sequence in a large HMM that looks like the one depicted on Figure 2.3 for the case of an isolated digit recognizer (digits 0 to 9).

In continuous digit recognition (CDR), the automaton has to be extended to accommodate multiple words in an utterance. This is accomplished by introducing an opportunity to transit from the last state of each

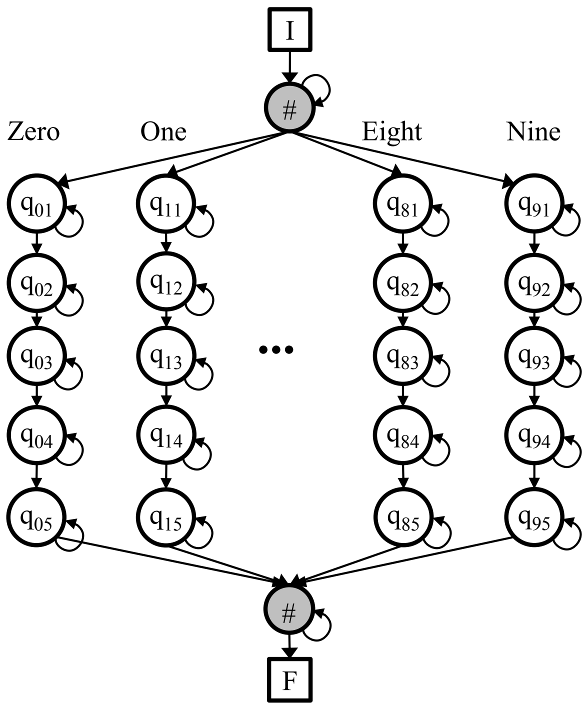


Figure 2.3: HMM automaton for isolated digit recognition with an initial state I and a final state F. q_{ds} denotes the state s of digit d and $\#$ is the state assigned to silence.

digit to the first state of each digit (including the current digit). Therefore, the HMM corresponding to such a task becomes like the one depicted in Figure 2.4. The recognizer has to find the best path (state sequence) through the automaton, and from that sequence it can retrieve the corresponding word sequence. The penalty factor, P_o, controls the balance of insertion and deletion errors.

2.1.2.2 Viterbi Decoder

Finding the best path through an HMM is achieved by means of a Viterbi decoder [46]. This decoder finds the most-likely sequence of acoustic states for an utterance of length T. If K is the total number of HMM states (i.e., sum of the number of states over all the words in the vocabulary), the decoder applies dynamic programming [47, 48] to find the most likely path though a trellis $\mathbf{M}_{I\times T}$ where $m_{i,t}$ represents the probability that the acoustic vectors $U_1...U_t$ are generated along a state sequence that ends in

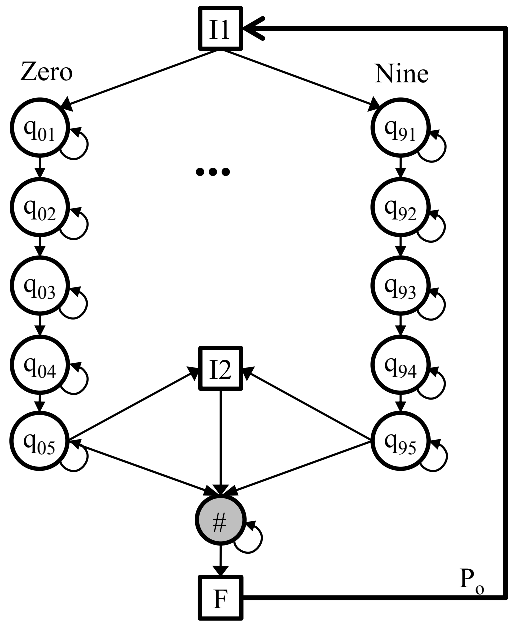


Figure 2.4: HMM automaton for CDR with two initial states $I1$ and $I2$ and a final state F.

$q_t = i$.

2.2 Main Approaches for Acoustic Modeling

The common denominator of most AM approaches is that they model the variability of the speech in state q by means of stochastic models which either estimate $P(U|q)$ or $P(q|U)$ where U represents one acoustic feature vector.

Before going into more detail, it is worth to mention that within the past thirty years, the dominant approach was to compute $P(U|q)$ by means of a Gaussian mixture model (GMM). In what follows, I briefly describe first how a GMM is used to estimate $P(U|q)$ and then I review the most relevant alternative approaches that were proposed in the hope to improve the AM.

2.2.1 Gaussian Mixture Models

A Gaussian Mixture Model (GMM) [48–50] models the probability density function $P(U|q)$ as

$$P(U|q) = \sum_{k=1}^{K} \mathrm{w}_{qk} \mathcal{N}_{qk}(U, \mu_{qk}, \Sigma_{qk}). \tag{2.1}$$

This is a weighted sum of K multi-variant Gaussian densities $\mathcal{N}_{qk}$. A GMM has the following trainable parameters: the mixture weights w_{qk}, the mixture means μ_{qk} and the mixture variances Σ_{qk}. One can either independently model each state by a mixture of state-specific Gaussians, or model all data by a pool of Gaussians and then select the mixture components from that pool. Figure 2.5 shows a mixture of three Gaussians for a

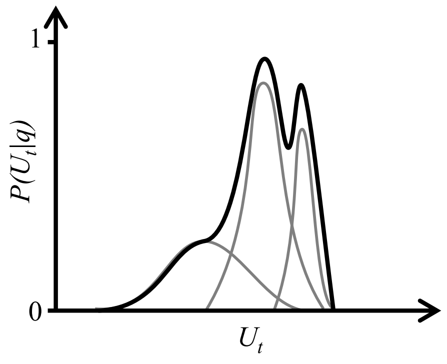


Figure 2.5: Example of a GMM density function for a simplified one-dimensional case.

one-dimensional density function.

The likelihood of the acoustic vector sequence $\mathbf{U}$ along a state sequence $\mathbf{q}$ is computed as

$$P(\mathbf{U}|\mathbf{q}) = \prod_{t=1}^{T} P(U_t|q_t) \tag{2.2}$$

Training of the models is achieved by means of a procedure such as Maximum Likelihood Estimation (MLE) [48, 51, 52] that maximizes the log likelihood sum of the training data using the iterative Expectation Maximization (EM) algorithm [53]. Instead of using such a generative method, one can also opt for a discriminative training [54–56] that maximizes the probability of the correct state sequence given the acoustic feature stream.

A possible technique for discriminative training of GMMs is Maximum Mutual Information (MMI) training [57].

GMMs are still popular, particularly in commercial products. There are also many toolkits implemented for training and evaluating them (e.g., HTK [58] and SPRAAK [59]) in a very efficient way. Furthermore, in practice, GMMs are shown to be much faster in both training and evaluation than competitors based on e.g., ANNs or support vector machines (SVMs)
.

2.2.2 Artificial Neural Networks

At the end of the 1980s, artificial neural networks were introduced for ASR [60–64]. An ANN is a network of interconnected neurons that are simple simulations of cells in the human brain. Each neuron is stimulated by the acoustic inputs and/or by other neurons and its output on its turn also stimulates a large number of other neurons

If X_t is the input to neuron i at time t and W_i the weights of the input connections, the neuron activation is computed as $a_{t,i} = W_i^T X_t + w_{i0}$ and the neuron output is obtained by applying a compressing non-linear function $f(\cdot)$ on this activation (see Figure 2.6 (a)). An example of such a

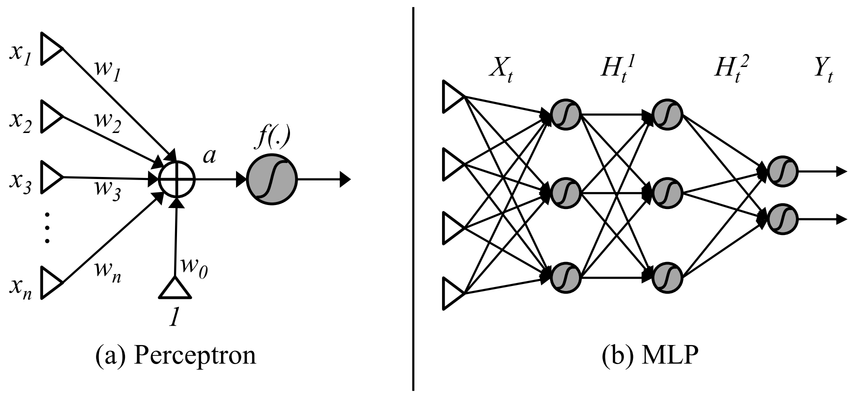


Figure 2.6: Schematic view of (a) a perceptron and (b) an MLP with two hidden layers, input X_t, hidden layer activations H_t and output Y_t at time t.

function is the sigmoid function:

$$f(a_{t,i}) = \frac{1}{1 + e^{-a_{t,i}}} \tag{2.3}$$

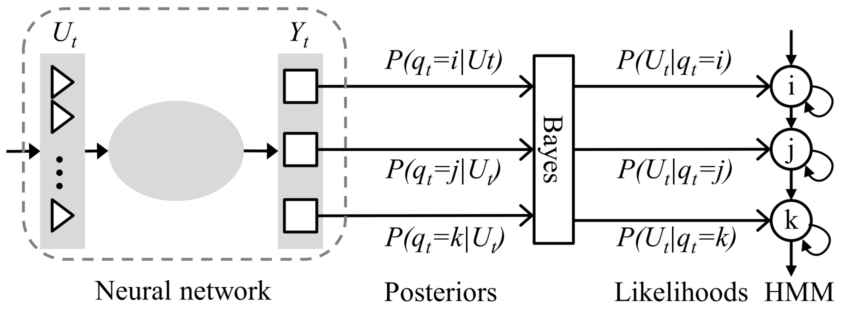


Figure 2.7: The basic architecture of a hybrid ANN-HMM system where the ANN is used to estimate the state posterior probabilities.

The aim of $f(a_{t,i})$ is to project the entire real axis on a bounded interval ($[0, 1]$ in the sigmoid case).

The neurons are usually organized in layers and the outputs of one layer stimulate the neurons of the next layer (see Figure 2.6(b)). The last layer is called the output layer because it produces the outputs of the ANN. The other layers are called hidden layers. The number of layers is often called the depth of the network.

In acoustic modeling, the output layer is designed to provide outputs that resemble the posterior probabilities $P(q_t|U_t)$ of the HMM states [65]. The objective is to attain that

$$y_{t,i} = P(q_t = i|U_t)$$

The resulting posterior probabilities can then be converted to scaled likelihoods using Bayes' law (see Figure 2.7) and $P(\mathbf{U}|\mathbf{q})$ can be computed as

$$P(\mathbf{U}|\mathbf{q}) = \prod_{t=1}^{T} P(U_t|q_t) = \prod_{t=1}^{T} \frac{P(q_t|U_t)P(U_t)}{P(q_t)}. \tag{2.4}$$

Since $P(U_t)$ is independent of the state hypothesis, it can be discarded in the search for the best state sequence [66].

Training an ANN-based acoustic model boils down to optimizing all the weights of the interconnections between neurons. The approaches used to train the network weights strongly depend on the network architecture. But as the ANN outputs are expected to represent posterior probabilities, the ANN must always be trained by means of a discriminative procedure.

2.2.2.1 Multi-Layer Perceptrons

If only forward connections are used, the ANN is called a feed-forward network (FFNN), and each speech frame U_t is processed independently of the previously processed frames. An FFNN with multiple hidden layers is referred to as a Multi Layer Perception (MLP) [67–69]. The weights of a simple Perceptron can be found with a supervised logistic regression training [70, 71] that minimizes a cost function (also called error function) such as the Mean Squared Error (MSE) or the Cross-Entropy (CE) [65] between the actual output y and the desired output d. The MSE and CE are computed as follows:

$$MSE = \frac{1}{2} \sum_{t=0}^{N_{frm}} \sum_{j=0}^{N_{out}} (\mathrm{y}_{tj} - \mathrm{d}_{tj})^2 , \tag{2.5}$$

$$CE = - \sum_{t=0}^{N_{frm}} \sum_{j=0}^{N_{out}} [\mathrm{d}_{tj} \ln(\mathrm{y}_{tj}) + (1 - \mathrm{d}_{tj}) \ln(1 - \mathrm{y}_{tj})] , \tag{2.6}$$

where t indexes the N^{frm} training examples and j indexes the N^{out} network outputs (i.e., total number of states). Logistic regression works with a gradient descent procedure that changes a weight by an amount that is proportional to the derivative of the error function to that weight. In the case of an MLP, a similar training concept, called Error Back-Propagation (EBP) [69] is adopted to optimize the weights of the hidden layers. A classical MLP is fully connected but sparse networks (SMLPs) [72] are possible as well. The term hierarchical MLP [73–75] is used to describe a cascade of two or more MLPs that are trained one after the other without a joint optimization. Each MLP of such a hierarchy may contain multiple hidden layers and the output of a preceding MLP is directly fed to the next one using a feature stacking approach in order to increase the context gradually from one MLP to the other.

2.2.2.2 Time-Delay Neural Networks

A direct way to take temporal dependencies into account is to stack a window of frames in the network input (instead of only one frame). As the major ANN-based alternative approaches, one can name time-delay neural networks (TDNN) and recurrent neural networks (RNN). Time-delay neural networks [76, 77] represent an effective attempt to train an MLP [78] for time-sequence processing, by converting the temporal sequence into a spatial sequence over corresponding units. An example of a TDNN is shown in Figure 2.8 (a).

At each time step t, the input to the hidden layer is a window of the current feature vector along with the past $n-1$ preceding vectors, U_t^{t-n}, and the input to the output layer is a combination of current hidden layer output along with those of the past $m-1$ preceding ones, H_t^{t-m}. The same extension can also be applied to subsequent layers, introducing a tapped-delay mechanism between hidden units (e.g., only the first block of units in the tapped line actually receives input from the previous layer), giving the ability to deal with more complicated time dependencies. The EBP algorithm can be used to train such a network as well.

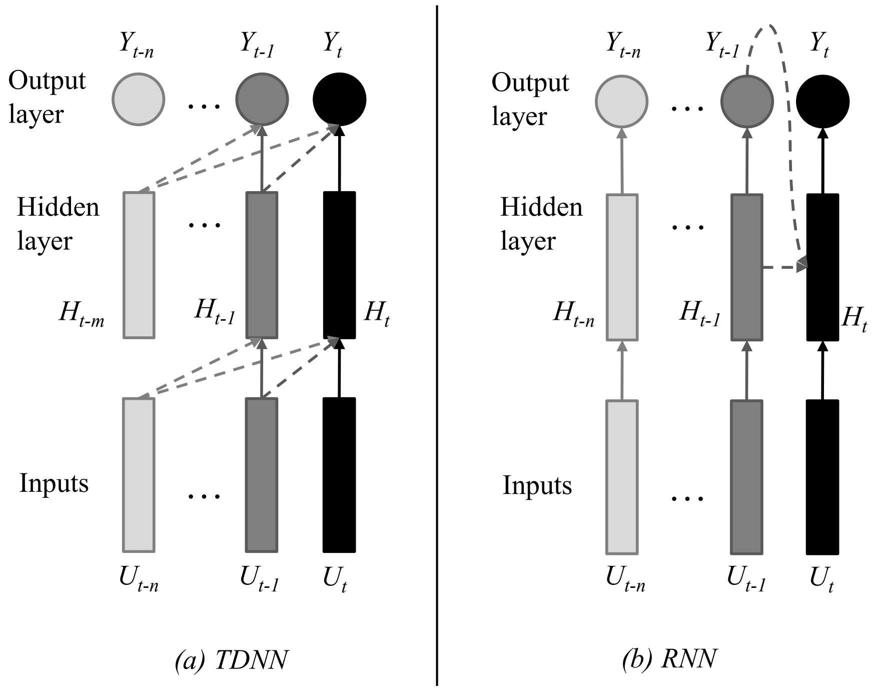


Figure 2.8: (a) Time-delay neural network. Input is fed into the rightmost set of input units (U_t) at time t and previous inputs are shifted to the left. A similar mechanism holds for the hidden layer. The output layer of the net integrates the activations of the hidden units over time. (b) Recurrent Neural Network. The neurons can model the dynamics over time by using the internal neuron activations from time step $t-1$ while processing an observation of time step t. The gray colors and the dashed arrows resemble, respectively, the situation of the system and the stream of data through time.

2.2.2.3 Recurrent Neural Networks

In Recurrent Neural Networks (RNNs) [79], the neurons are supplied with the input observations along with the past outputs of the neurons in the hidden and output layer (see Figure 2.8 (b)). Hence, RNNs provide dynamic modeling by using the neuron activations at time step $t - 1$ while processing an observation of time step t. The network actually computes $Y_t = P(q_t|U_t, H_{t-1}, Y_{t-1})$ and provides a fading memory of the entire past. Due to this connectivity, the networks can model sequential behavior and take dependencies between frames into account.

RNNs cannot be trained with simple EBP training and call for a more complex training procedure [80] such as Back-Propagation Trough Time (BPTT) [81] or Real-Time Recurrent Learning (RTRL) [82]. In the case of BPTT training the network is unfolded in time (given a certain depth) and EBP is performed through the resulting unfolded feed-forward network. RNNs are causal models that only encode the recent past in their internal memory. They do not access the right (future) context of an observation. This capability can be added by feeding the RNN with features that provide information about the future (stacked future frames and/or dynamic features computed for the present frame). Another approach is to perform bi-directional processing [83]. In such a case, two networks are used, one that processes the sequence from left-to-right and a second one that works from right-to-left. Joining the two networks provides a model that takes the past and future context of an observation into account.

2.2.2.4 Deep Neural Networks

A DNN is a feed-forward NN with more than one hidden layer. Each hidden neuron typically uses the logistic function to map its activation emerging from its inputs to a bounded output that stimulates the next layer.

The networks are trained so that each layer of neurons represents a different level of abstraction, obtained with a layer-wise training procedure. In the Deep Belief Network approach [84], a two-stage training procedure is used for fitting the DNNs. In the first stage, layers of feature detectors are initialized, one layer at a time, by fitting a stack of generative models, each of which has one layer of latent variables. These generative models are trained to be good at modeling the structure in the input data, which means, without using any information about the speech units (i.e., HMM states) that the acoustic model will need to discriminate. After this generative 'pre-training', the multiple layers of feature detectors can be used as a much better starting point for a discriminative 'fine-tuning' phase during which back-propagation through the DNN slightly adjusts the weights found in the pre-training to predict the target HMM states [64, 85].

Such a pre-training creates many hyper features (transformations of the input features). In practice, not all hyper features are useful for the discrimination, but a fair fraction of them will be far more useful than the raw inputs. The generative pre-training finds a region of the weight space that allows the discriminative fine-tuning to make rapid progress, and it also significantly reduces over-fitting [64, 86].

Optimizing the hidden layers in DNNs using gradient descent algorithms is not that straightforward. If the initial weight scales are not set cleverly, the back-propagated gradients will have very different magnitudes in different layers [87]. Furthermore, the strength of DNNs relies on the fact that they have many hidden layers and neurons. Consequently, there are many trainable parameters which make the DNNs capable of modeling very complex nonlinear relations between the inputs and outputs. Although this strength is crucial for acoustic modeling, one has to provide the training samples very carefully. Otherwise, the neural network may also learn some spurious patterns that are an accidental property of the particular examples in the training set, leading to over-fitting to the training data. A remedy can be early stopping the training procedure which also reduces the modeling power of the system. One can also collect a huge amount of training data [88] to suppress these accidental relationships without loosing the modeling power, but of course with the cost of increasing computational complexity.

2.2.2.5 Long Short-Term Memory Networks

A recent variant of the RNN approach is the Long Short-Term Memory (LSTM) network [89, 90]. It is a special RNN that is enriched with complex memory cells that can store certain activations for an arbitrary length of time. Hence, an LSTM can model sequences that are governed by long time dependencies over time lags of unknown size. The memory cells are controlled by gates that determine if an activation should be stored, accumulated or forgotten. The training of both the neurons weights and the memory cell gate weights is accomplished with BPTT or RTRL. Although it is not so clear whether long-time dependencies need to be modeled in the AM, some recent research by Google demonstrated good potential for LSTMs in LVCSR [91].

2.2.3 Combining Methods

Next to employing ANNs for class posterior probability estimation, ANNs can also be combined with GMMs in a single system. Examples of such approaches are the Tandem approach [92–94], and the model combination approach [95–97].

- *Tandem*: Normally, GMMs and ANNs are trained separately. In a Tandem approach [92–94], the ANN is used as a feature extractor whose outputs are supplied to a GMM-based speech recognizer. In principle, one could supply the phoneme state likelihoods of an ANN-based system to a GMM-based recognizer. Considering the typical 3-state model per phoneme, the feature vector to GMM has around 120 components, whereas in practice, GMMs favor feature vectors with around 40 to 80 components and with uncorrelated components. This can be obtained by training an ANN with a final hidden layer of 40 to 80 neurons, which then feeds into the output layer modeling the HMM state posteriors. A Tandem system then discards the ANN output layer and feeds the activation of the final hidden layer to the GMM instead.

- *Model combination*: Instead of putting ANN- and GMM-based systems in a sequential order, one can train the ANN and GMM to produce the state likelihoods, and then linearly combine these likelihood scores using the following equation:

$$log(P(q)) = \alpha log(P(q)_{GMM}) + (1-\alpha) log(P(q)_{ANN}) \quad (2.7)$$

  where $log(P(q))$ is the combined log-likelihood for state q, $P(q)_{GMM}$ is the likelihood given by the GMM, $P(q)_{ANN}$ is the average posterior given by the ANN, and α is the weight of combination. In section 6, this approach will be further examined in the context of noise robust ASR.

2.3 Reservoir Computing Networks

2.3.1 Motivation

Both GMM- and ANN-based speech recognizers have advantages and disadvantages. On the one hand, GMM-based systems can rely on a strong research and development background, and they are much faster in both training and evaluation. On the other hand, they have shown to be very sensitive to the mismatch between training and test conditions. Table 2.1 lists the performance of different GMM-based systems trained on clean and tested on clean as well as noisy data from the Aurora-2 benchmark. While

Method	Clean	0-20	-5dB
GMM-HMM (ETSI)	0.97	38.9	91.5
GMM-HMM (MVN)	0.84	19.7	82.2
GMM-HMM (AFE)	0.77	13.0	69.7
GMM-HMM (VTS)	0.40	(7.3)	-
GMM-HMM (MDT)	-	11.4	-
GMM-HMM (EB)	0.40	(5.6)	(45.8)
GMM-HMM (D-HMM)	0.51	36.0	90.0
GMM-HMM (UD)	-	10.3	-
DBN-GMM-HMM (Tandem)	1.26	21.0	74.6

Table 2.1: WERs (in %) obtained in clean speech training experiments on Tests A - C of Aurora-2. Results between brackets are obtained by employing a noise dictionary and are, therefore, biased towards the case of multi-style training.

human performance remains stable below 1% WER up to SNRs 5-10 dB, none of the listed GMM-based systems comes even close to that. The fact that artificial neural networks are supposed to imitate the behavior of human brain cells raises the hope that an ANN-based speech recognizer might be more robust against the presence of noise. However, at the time I started my research, there was no evidence for this. On the contrary, the discriminatively trained ANNs seemed to reduce the noise robustness. In spite of that, I did have reasons to believe that neural networks based on Reservoir Computing Networks (RCN) [98] have much more potential because an RCN can focus on the speech dynamics and thereby ignore the noise effects.

2.3.2 Theory

The basic principle of Reservoir Computing (RC) is that information can be retrieved from sequential inputs by means of a two-layer Recurrent Neural Network (RNN) with the following characteristics (see Figure 2.9). The first layer is a hidden layer composed of non-linear neurons which, at time t are driven by actual inputs U_t and delayed hidden layer outputs R_{t-1}. A key point is that the hidden neurons have randomly fixed input weights and recurrent weights and thus, that they do not follow from any training procedure. The second layer consists of linear neurons which are driven by the actual hidden layer outputs R_t and which have trainable coefficients. The recurrently connected hidden neurons can be imagined as a pool of interconnected computational neurons, excited by inputs. Such a pool of

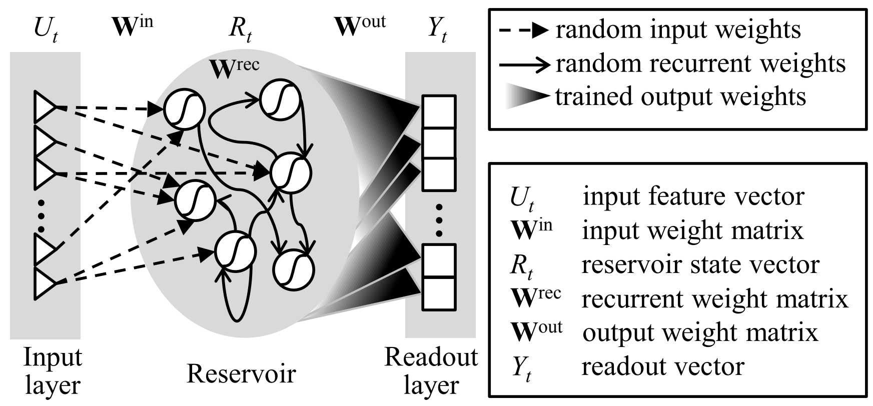


Figure 2.9: A basic RCN system consists of a reservoir and a read-out layer. The reservoir is composed of interconnected non-linear neurons with randomly set weights. The readout layer consists of linear neurons with trained weights.

neurons is called a *reservoir*. Together with the linear output neurons, it forms a *reservoir network*. The network outputs Y_t are usually called *readouts* [99] to differentiate them unambiguously from the reservoir outputs R_t and to indicate that they 'read out' the reservoir states.

The reservoir can perform a temporal analysis of the input stream. In order to be effective, it should have the so-called echo state property [99]. The latter states that, with time, the reservoir should forget the initial state it was in. This corresponds to the requirement that a linear filter should have an impulse response that fades out to be suitable for performing a meaningful short-term analysis of a non-stationary signal such as speech. As each reservoir output is the output of a non-linear filter with multiple inputs and as the filter coefficients are chosen randomly, a big enough reservoir will give rise to a large variety of filters. By training the readouts on speech, they will focus on the outputs of those filters that resonate to frequencies which are typical for the dynamics of the speech signal. This leads to the hypothesis that an RCN can filter-out noise-inflicted dynamics (modulations) whose frequencies are a-typical for speech.

Reservoir Network Equations

If there are N^{in} input features and N^{out} readouts and if the reservoir consists of N^{res} neurons, the reservoir network is governed by the following

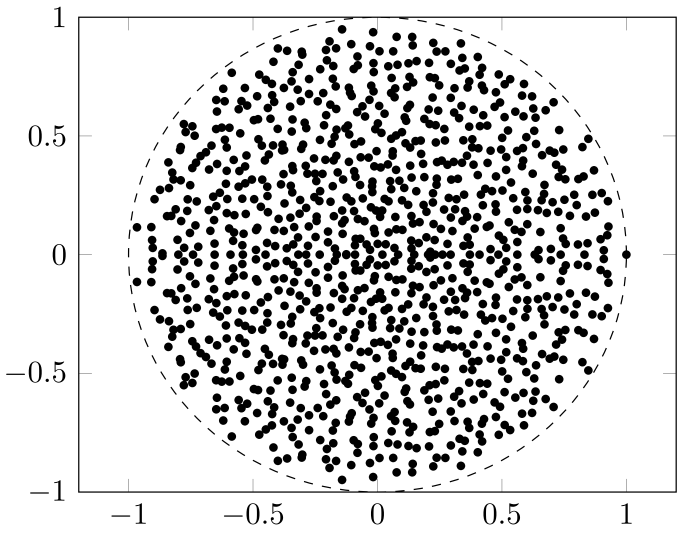


Figure 2.10: The eigenvalues of the recurrent weights matrix corresponding to a 1000-node reservoir and the spectral radius of 1. The latter is revealed by the fact that the matrix is divided by its largest absolute eigenvalue.

equations:

$$\begin{aligned} R_t &= f_{res}(\mathbf{W}^{in}U_t + \mathbf{W}^{rec}R_{t-1}), && (2.8) \\ Y_t &= \mathbf{W}^{out}R_t. && (2.9) \end{aligned}$$

The $N^{res} \times N^{in}$ matrix $\mathbf{W}^{in}$ and the $N^{res} \times N^{res}$ matrix $\mathbf{W}^{rec}$ comprise the weights of the input connections and the recurrent connections, respectively. $f_{res}(\cdot)$ is an activation function performing component-wise non-linear transformations of the neuron activations (e.g., $f_{res}(\cdot) = \tanh(\cdot)$). In order to compute Y_t, we actually extend R_t with a bias of 1. It can be shown [99] that the echo state property holds if the *spectral radius* ρ, defined as the maximal absolute eigenvalue of $\mathbf{W}^{rec}$, is smaller than 1.

Constructing a $\mathbf{W}^{rec}$ with a spectral radius ρ takes three steps: (1) draw the weights from a normal distribution with a variance of 1, (2) divide them by the spectral radius of the matrix, and (3) multiply the weights by a factor ρ. Figure 2.10 shows that the eigenvalues of the matrix following from the second step are uniformly distributed inside the unit circle.

In order to deal better with the random inter-frame changes observed in the inputs (e.g., due to the spectral analysis), one can introduce leaky integration by replacing Equation (2.8) with

$$R_t = (1-\lambda)R_{t-1} + \lambda\, f_{res}(\mathbf{W}^{in}U_t + \mathbf{W}^{rec}R_{t-1}), \qquad (2.10)$$

with $0 < \lambda \leq 1$. In this case, the reservoir neurons are called Leaky Integrator Neurons [100] with a leak rate λ. All reservoirs in this thesis employ neurons of this type. The dynamical behavior of the reservoir can be expressed in terms of two time constants:

$$\tau_\rho \doteq -\tau_{fr}/\ln(\rho) \quad \text{and} \quad \tau_\lambda \doteq -\tau_{fr}/\ln(1-\lambda), \tag{2.11}$$

with τ_{fr} being the time-shift (e.g., in ms) between frames. It will become clear that τ_ρ determines the memory capacity of the reservoir, whereas τ_λ determines how smooth the readout *patterns* (i.e., temporal evolution) are going to be.

2.3.3 Application of RCN in Robust ASR

The discussion about the robustness of RCNs can be followed easier if at first, we consider a simpler type of such a network without any recurrent connections. Such a system is called an Extreme Learning Machine (ELM) [101–103]. The ELM is defined as a feed-forward neural network (a Multi-Layer Perceptron or MLP) with a randomly fixed hidden layer and a linear output layer whose weights are fixed to minimize the mean squared difference between the computed and the desired outputs. In [101], it is mathematically proven that the ELM is as powerful as a fully trained MLP: a system with N hidden neurons can learn exactly N distinct observations. Moreover, it has been shown in [104] that among the solutions yielding the same training error, the one resulting from the Moore-Penrose pseudo inverse leads to the lowest norm of the output weight matrix. This latter property is, according to [101], the key to a better generalization of the ELM to unseen test data. The various experiments presented in [101] lead to the following conclusions: (1) in terms of generalization, an ELM behaves as well as a Support Vector Machine (SVM) employing a linear kernel [105], and much better than a fully trained MLP, (2) an ELM is much more compact than an SVM (it needs less hidden neurons than an SVM needs support vectors) and (3) the generalization performance of an ELM remains stable over a wide range of hidden units. We argue that good generalization to test data should transfer to good noise robustness. Moreover, we expect that introducing recurrent connections which are also randomly fixed, is bound to maintain the noise robustness while improving the model accuracy. Adding these arguments to the formerly mentioned noise filtering capacity of an RCN, gives us enough reasons to believe in the high potential of RCNs for noise robust CDR.

Obviously, an RCN also resembles an SVM, but one with a hidden space whose size and identity do not follow from a long and delicate supervised training process. Likewise, it resembles a radial basis function (RBF) network which also embeds a fixed hidden layer. However, the RBFs often follow from a clustering procedure and they typically represent local functions in the input feature space, meaning that they only react to inputs that fall in a restricted area of the input space. A reservoir neuron, on the other hand, typically has a non-local activation function. Finally, an RCN resembles the recurrent neural network (RNN) applied in [106] for continuous speech recognition. However, the memory of that network originated from feeding the outputs back to the hidden layer and, importantly, all the network weights were trained by means of back-propagation through time, a method that is found to be very time consuming and likely to yield a sub-optimal solution when the size of the network is large. In an RCN, the optimal weights are found in a straightforward manner, even for a reservoir with several thousands of neurons.

Part II

Acoustic Modeling

3 First Attempts to RCN-Based Noise Robust CDR

This chapter is an edited version of the following original publication:

[1] A. Jalalvand, F. Triefenbach, D. Verstraeten, and J.-P. Martens, "Connected digit recognition by means of reservoir computing," *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1725–1728, 2011.

3.0 Preface

This chapter comprises our first attempts to develop an RCN-based spoken digit recognizer. The research has been conducted in three phases: (1) exploration of the scene by conducting some preliminary experiments to get insight in the potential of reservoirs and the sensitivity of their performance as a function of the reservoir size and the reservoir dynamics, (2) development of an RCN-based continuous digit recognizer for clean speech, and (3) development of a noise robust RCN-based continuous digit recognizer (CDR). All the experiments presented in this chapter are conducted on the Aurora-2 dataset which is composed of clean and noisy speech. A challenge in this stage of the research was that the Aurora-2 utterances only come with a digit or a digit sequence transcription. They are not hand-segmented into silences and digits, meaning that there is no frame-wise labeling of the speech, as normally needed for the training of an RCN.

3.1 Introduction

Although Hidden Markov Models (HMM) have a proven track record when it comes to model the speech acoustics for automatic speech recognition (ASR), they have regularly been challenged by alternative methods. Many of them try to alleviate the state-independency hypothesis underlying the HMM paradigm. One such a suggestion is to model the dynamics and contextual dependencies in speech by means of a Recurrent Neural Network (RNN) [106]. It has been shown that RNN-based systems can indeed attain a good performance, but error back-propagation through time is a rather complex, critical and time consuming training method. The recently proposed Reservoir Computing Network (RCN) [80, 98] may provide an elegant solution to the training problem.

The basic idea of reservoir computing is that complex classifications can be performed by means of a pool of fixed (untrained) nonlinear interacting neurons, called the reservoir, and a set of trained linear classifiers that operate in the reservoir state space. The reservoir state is defined as the collection of the reservoir neuron outputs. The reservoir is nothing but an RNN which offers the capacity to model the dynamics of speech. The linear classifiers can be compared to the hyperplanes in the high-dimensional hidden feature space of an SVM [107], but with this difference that the latter space is obtained after training whereas the reservoir state space is constructed by a random process. The RCN concept has already successfully been applied to different types of problems, including robot control, sequence generation and analysis, and even isolated spoken digit recognition [108, 109]. Recently, we have been able to demonstrate good English phoneme recognition capabilities as well [8].

In the present paper, we propose the first RCN-based connected digit recognizer, and we demonstrate good performance on the Aurora-2 dataset, a dataset composed of clean and noisy speech. In Section 3.2, we review the basics of RCN and in Section 3.3, we provide some information concerning our first steps into the world of reservoir-based digit recognition. In Section 3.4, we describe the development of our RCN-based recognizer of connected digits and in Section 3.5 we make an experimental assessment of the created system. The paper ends with some conclusions and ideas for future work.

3.2 Reservoir Computing Network

A simple RCN system (see Figure 3.1) consists of a reservoir and a set of readout units. The reservoir is a pool of non-linear neurons which are con-

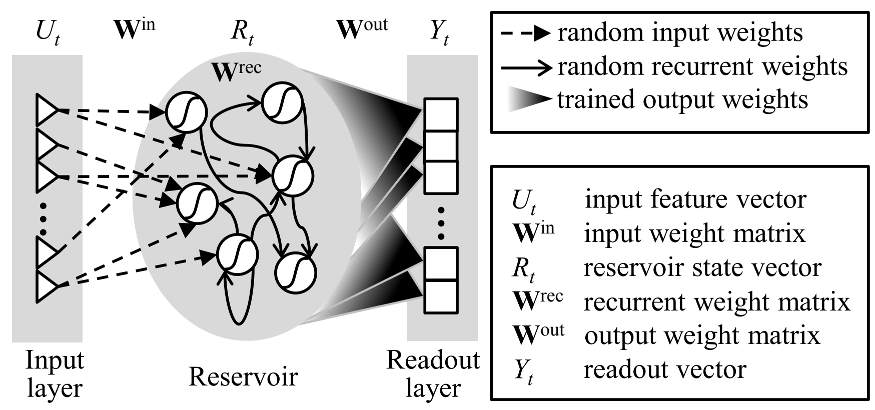

U_t
$\mathbf{W}^{\text{in}}$
R_t
$\mathbf{W}^{\text{out}}$
Y_t
$\mathbf{W}^{\text{rec}}$
random input weights
random recurrent weights
trained output weights
U_t input feature vector
$\mathbf{W}^{\text{in}}$ input weight matrix
R_t reservoir state vector
$\mathbf{W}^{\text{rec}}$ recurrent weight matrix
$\mathbf{W}^{\text{out}}$ output weight matrix
Y_t readout vector
Input layer
Reservoir
Readout layer

Figure 3.1: A basic RCN system consists of a reservoir and a readout layer. The reservoir consists of nonlinear neurons with randomly fixed weights on the input and recurrent connections. Only the weights to the output nodes are being trained.

nected to the inputs via input connections and to each other via recurrent connections. The weights on these connections are randomly generated and kept fixed throughout the system development. The so-called spectral radius, ρ, defined as the largest eigenvalue (in absolute terms) of the recurrent weight matrix, controls the dynamics of the system. Each output node computes a linear function of the reservoir state, and the parameters of that function form the weights of the output connections. They are trained to achieve that a particular output node is high for observations of a particular class (e.g., phoneme or digit) and low for observations of any other class. Since the output node is linear, the output connection weights are obtained by linear regression. Since the output nodes 'read' the reservoir state, they are usually called the *readouts*.

If N^{in} is the number of inputs, N^{res} the number of reservoir neurons (the reservoir size) and N^{out} the number of output neurons, the connection weights are collected in the matrices $\mathbf{W}^{in}$, $\mathbf{W}^{rec}$ and $\mathbf{W}^{out}$, that have the dimensions of $N^{in} \times N^{res}$, $N^{res} \times N^{res}$ and $N^{res} \times N^{out}$, respectively. If U_t, R_t and Y_t represent the vector of inputs, reservoir outputs and readout nodes at time t respectively, the RCN equations can be written as

$$R_t = f_{res}(\mathbf{W}^{in}U_t + \mathbf{W}^{rec}R_{t-1}), \tag{3.1}$$

$$Y_t = \mathbf{W}^{out}R_t. \tag{3.2}$$

The function f_{res} is the so-called activation function of the reservoir nodes. In this work, $f_{res}(a) = \tanh(a)$.

The output weights $\mathbf{W}^{out}$ are determined by means of Ridge regression with the mean squared error as the objective criterion. If the reservoir state and the desired output vector at different times constitute the columns of the matrices $\mathbf{R}$ and $\mathbf{D}$ respectively, and if N^{frm} represents the number of columns (training frames), the output weights are obtained from

$$\hat{\mathbf{W}}^{out} = \underset{\mathbf{W}^{out}}{\arg\min} \left(\frac{1}{N^{frm}} \left|\left|\mathbf{W}^{out}\mathbf{R} - \mathbf{D}\right|\right|^2 + \epsilon \left|\left|\mathbf{W}^{out}\right|\right|^2 \right), \quad (3.3)$$

In this equation, ϵ is the regularization parameter. It is intended to prevent over-fitting to the training data. The solution is obtained in a closed-form [110] as

$$\hat{\mathbf{W}}^{out} = (\mathbf{R}\mathbf{R}^T + \epsilon\,\mathbf{I})^{-1}(\mathbf{D}\,\mathbf{R}^T), \quad (3.4)$$

To extend the integration of information over time, one can substitute the memoryless reservoir neurons by Leaky Integrator Neurons (LIN) [98]. Equation (3.1) then changes to

$$R_t = (1-\lambda)R_{t-1} + \lambda\, f_{res}(\mathbf{W}^{in}U_t + \mathbf{W}^{rec}R_{t-1}), \quad (3.5)$$

with $0 < \lambda \leq 1$. The parameter λ (leak rate[1]) encodes an integration time constant τ (in frames) via $\lambda = 1 - e^{-1/\tau}$.

3.3 First Steps into Reservoir-Based Digit Recognition

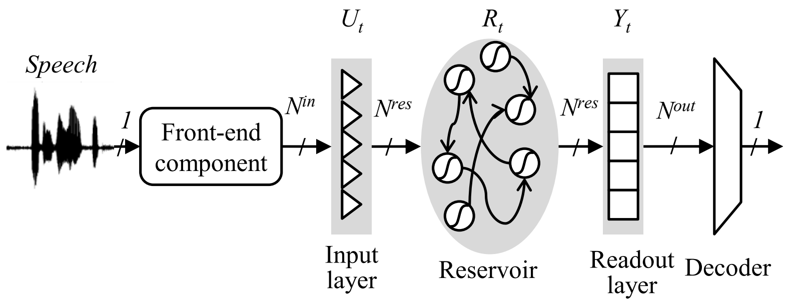
[This section was not in the original paper, but was added here because it recalls my first steps into the world of reservoir computing, and the first insights I gained at the beginning of my research.]

In a very first attempt to devise an RCN-based isolated digit recognizer, I constructed a system with a reservoir of 500 nodes and an architecture as depicted in Figure 3.2.

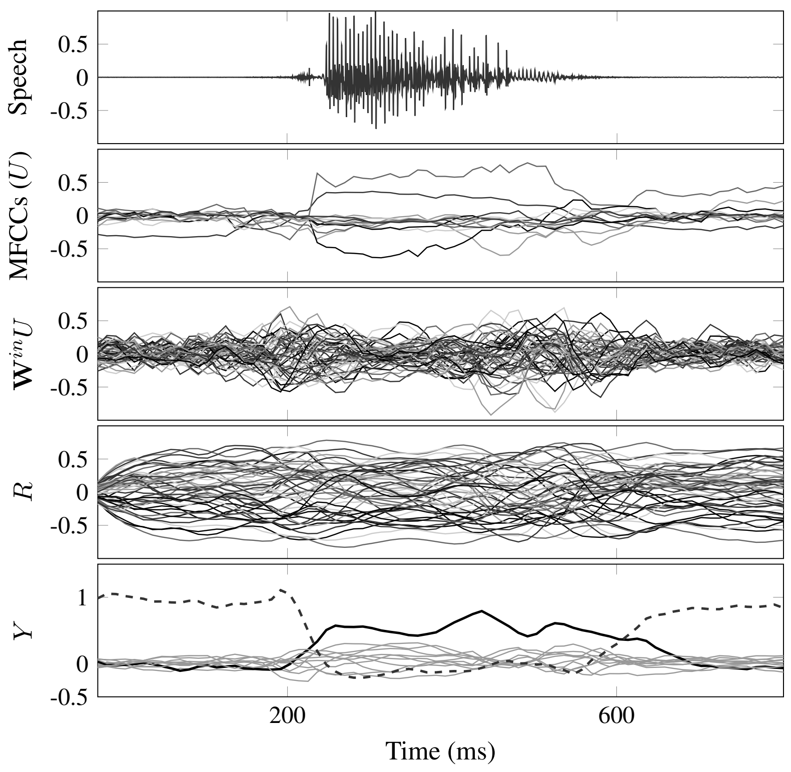
The acoustic features extracted from the input signal were raw MFCCs (plus log-energy) and their first and second derivatives. The output classes were the eleven digits (including the variants *zero* and *oh* for digit 0) and *silence*. During training, each frame was associated with one of these classes. The silence frames at the beginning and the end of the utterance were delimited using a simple energy-based silence detector. This detector used a decision threshold that was given by

$$E^{thr} = (1+\alpha)\min(\hat{\mathbf{E}}) + \beta\max(\hat{\mathbf{E}}), \quad (3.6)$$

[1]Note that in system theory, the leak rate is defined as 1 - λ. However, here we stick to the tradition of the reservoir computing literature.



(a) A simple RCN-based digit recognizer



(b) The input to each component of the RCN-based recognizer for an isolated sample containing digit 4 (FEK_4B.08)

Figure 3.2: A simple RCN-based digit recognizer along with an example of the inputs and outputs of its different components.

with α and β being two control parameters and with $\hat{\mathbf{E}}$ being the frame energy that was obtained after smoothing of the frame-wise energies. During operation, the decoder considered the digit class with the highest accumulated output as the recognized digit.

To get a quick idea of the importance of dynamic modeling offered by recurrent connections and LINs, I trained three systems comprising different types of reservoirs: one with simple neurons and without recurrent connections, one with simple neurons and with recurrent connections, and one with LINs and recurrent connections. The reservoir outputs and the readouts of these systems for a sample containing digit 1 (Aurora file MFK_1B.08) are shown in Figure 3.3.

Although adding the recurrent connections lead to the correct classification, there still is is a strong competition with some of the incorrect solutions. Introducing more memory by means of LINs causes the readouts to be smoother and the correct hypothesis to stand out more clearly. With the latter system I obtained a digit error rate (DER) of about 2.3% on Aurora-2, which was obviously still far above the state-of-the-art of 0.12% [109]. Adding an embedded training stage that makes its own silence-digit-silence segmentation, made it possible to reduce the DER to 1.1%. With a reservoir of 1000 nodes, the DER further dropped to 0.75%.

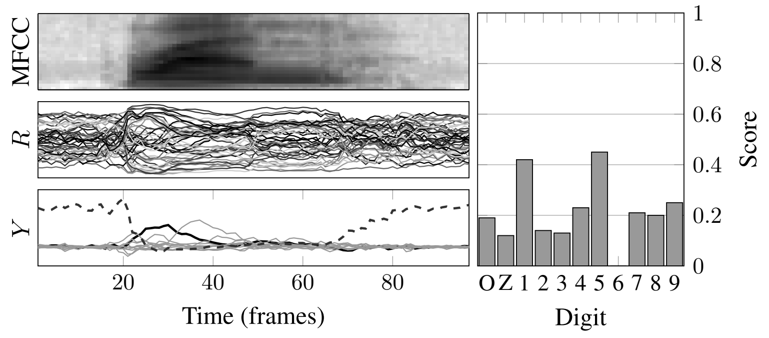
In summary, my preliminary experiments lead to the following insights: (1) recurrency in the reservoir and memory in the reservoir neurons are both effective means of capturing dynamical properties of the speech, and (2) good recognition requires big reservoirs[2]. These insights were at the basis of all my subsequent research.

3.4 Proposed Method

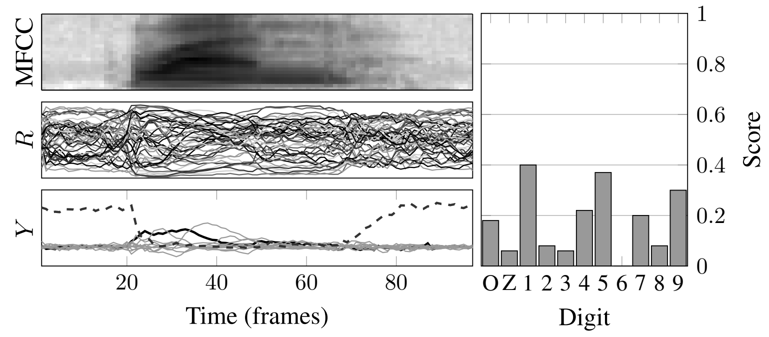
As mentioned in the previous section, we originally adopted the approach of [108] in which there is a single readout for each digit and for silence. However, like in GMM systems and like in [109], we generalized this approach to the case where each digit is modeled as a sequence of sub-word states, and each state is characterized by a readout (see Figure 3.4). During operation, the likelihood of being in a particular state is then determined by the readout associated with that state.

The system proposed here can be viewed as a hybrid RCN-HMM system. In the first stage, a dynamic system (the reservoir) converts the input feature vector sequence into a sequence of vectors in a high-dimensional inner space (the reservoir state space). In the second stage, emission probabilities at a certain time (frame) are computed as a linear combination of

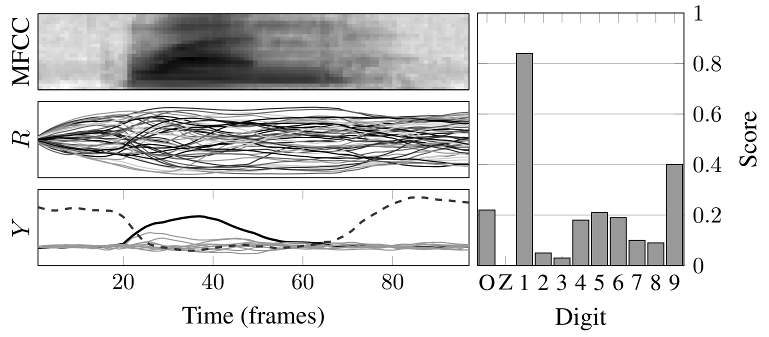
[2]At the time I did this research, there was no experience at all in the reservoir computing community with reservoirs of more than a few hundred nodes.



(a) Reservoir with simple neurons and without recurrent connections



(b) Reservoir with simple neurons and with recurrent connections



(c) Reservoir with leaky integrated neurons and with recurrent connections

Figure 3.3: (left) Acoustic features, reservoir outputs and readouts of three different RCN-based isolated digit recognizers, and (right) the score of each digit for a sample containing digit 1. The readouts of digit 1 and silence are highlighted.

the inner space variables at that time. The fundamental difference between the proposed system and a traditional hybrid system is that the mapping of the input features onto the inner space is traditionally trained, whereas here, it is completely random. We hope that the theoretical basis of SVMs – namely that an arbitrary binary classification in a well chosen (trained) high-dimensional inner space can be performed nearly optimally by means of a hyperplane – will also apply to the randomly created inner space.

In the subsequent sections we describe (1) the input feature sets we have used, (2) the reservoir weight generation scheme we have adopted, (3) the stochastic framework we have conceived for decoding the speech, and (4) the training procedure we have conceived for learning the readout weights from non-segmented isolated and connected digit utterances.

3.4.1 Input Features

As in most state-of-the-art recognizers, we worked with the standard Mel Frequency Cepstral Coefficient (MFCC) setup, delivering 13 static (c_1,.., c_{12} and log$E$), 13 velocity and 13 acceleration features. In theory, the reservoir should be capable of modeling the short-term dynamics of the speech, and therefore, would not need the non-static features. This is indeed confirmed experimentally to a large extent. But nevertheless, we stick to the traditional 39 inputs for two reasons: (1) since the input weights of the reservoir are fixed, more inputs do not raise the number of trainable parameters, and (2) adding the dynamic features does consistently offer a small benefit.

Since we also wanted to investigate the noise-robustness of our RCN-based recognizer, we conducted the final experiments with the noise-robust MSVA features, proposed in [111] (MSVA stands for MFCC, Spectral Subtraction, Spectral Flooring and moving-average smoothing).

3.4.2 Reservoir Weights Generation

The recurrent weights of the reservoir are randomly drawn from a zero-mean Gaussian distribution with variance V. That variance is a control parameter that can be used to change the spectral radius (ρ) of the reservoir. ρ is defined as the largest absolute eigenvalue of the recurrent weight matrix and is proportional to V. Spectral radius is known to determine the dynamical excitability of the reservoir [80, 98].

Traditionally, the input weights are randomly drawn from a uniform distribution between $-\alpha_U$ and α_U. The so-called input scaling factor (α_U) controls the relative importance of the inputs in the activation of the reservoir neurons. In [8], we refined this strategy by dividing the feature set in six sub-groups according to the dimensions (static, velocity, acceleration) and (MFCC, log-energy), and by using a separate input scaling factor for each sub-group. Here we adhere to this latter strategy. Note that traditional Gaussian mixture modeling does not require any input scaling at all since that scaling is encoded in the variances of the individual mixtures. However, the sensitivity of an RCN to the choice of the input scaling seems to become marginal once the reservoir is big enough, as will be the case in our system.

3.4.3 A Probabilistic Framework

The construction of a probabilistic framework first of all involves the creation of a finite state automaton which represents what is spoken (during training) or what can be spoken (during recognition). It also takes into account the topologies of the acoustic models one wants to use for the digits and the silence. Figure 3.4, for instance, shows the automaton that is used during recognition. The aim is to find the joint probability of observing the input sequence, $\mathbf{U}$, along the state sequence, $\mathbf{q}$, through the automaton. The requested probability is computed as

$$P(\mathbf{q}, \mathbf{U}) = \prod_{t=1}^{T} P(q_t|q_{t-1})\, P(U_t|q_t), \tag{3.7}$$

and $P(U_t|q_t)$ must be derived from the readout vector Y_t.

Suppose that the regression minimizes the mean squared error between the readout vector Y_t and the desired output vector D_t, and that all elements of D_t are -1, except the one corresponding to the desired state which is equal to +1. In that case, one can follow the derivations in [65] to show that under favorable conditions the rescaled readout node $y'_{t,q} = 0.5 + 0.5y_{t,q}$ will approximate the posterior probability vector $P(q|U_t)$, with q being any state of the automaton. In order to ensure that the probabilities are positive, we introduce the rescaled readouts as

$$y'_{t,q} = \max(\frac{y_{t,q}+1}{2}, y_o) \qquad 0 < y_o \ll 1 \tag{3.8}$$

and we compute the requested likelihood as

$$P(U_t|q_t) = \frac{P(q_t|U_t)}{P(q_t)}\, P(U_t) = \frac{y'_{t,q_t}}{P(q_t)}\, P(U_t) \tag{3.9}$$

The prior probability $P(q_t)$ is obtained as the mean of $P(q_t|U_t)$ over all training frames.

Obviously, the probabilities $P(U_t)$ in Equation (3.9) can be ignored during the probability maximization process as they are not a function of $\mathbf{q}$.

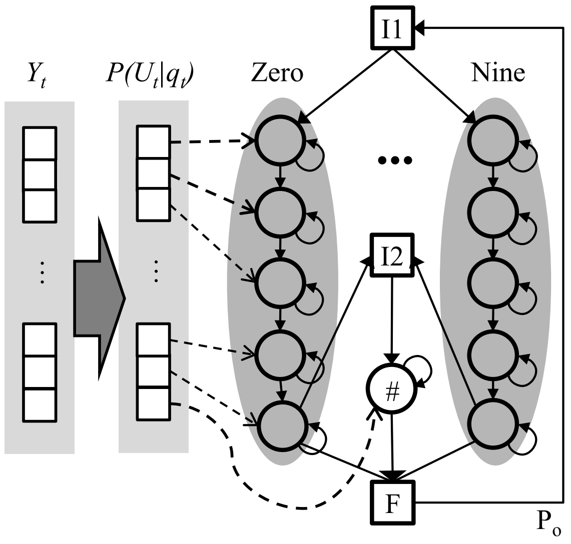


Figure 3.4: During recognition, the utterance model is a parallel loop of 11 digits and a silence, and each digit is modeled by a sequence of five states. On the left, the readout layer is depicted and the arrows indicate the mapping of readouts to state likelihoods.

3.4.4 Training the System

The training procedure is organized in two phases. First, we train a relatively small reservoir on the basis of isolated digit utterances. Then, we train a larger system on the basis of all utterances, connected as well as isolated digit utterances.

In the first phase, only isolated digit utterances are used because for these utterances, it is possible to generate target labels of sufficient quality for determining the readout weights of a first reservoir. In order to obtain these targets, we first perform an energy-based segmentation of each utterance into silence-digit-silence and then presume a linear state progression inside the digit. In successive Viterbi iterations, new target labels are derived from the most likely state sequences that were produced using the actual reservoir, and the readout weights are retrained on the basis of these labels. The process is continued for a couple of times (the number of iterations is not very critical since this is only the first phase of the training).

In the second phase, we also include connected digit utterances. These utterances are modeled as sequences of digits interleaved with optional silences, and surrounded by obligatory silences. Since we have much more utterances now, we can train a much larger reservoir. To start the training of that reservoir, we use the small reservoir emerging from phase 1 to derive initial target labels from a Viterbi alignment of the utterances with their models. From then on, the readout weights are refined in a number of successive Viterbi iterations, each time using the latest reservoir as the acoustical model. The training is continued until saturation of the word error rate (WER) measured on a validation set is observed.

3.4.5 Recognition

During recognition, the model of Figure 3.4 is used as the utterance model. In order to control the trade-off between digit deletion and insertion errors, a word penalty P_o is assigned to the transition from the end state (bottom) to the start (top) state. The value of that penalty will be determined by means of recognition experiments on a validation set.

3.5 Experimental Evaluation

All experiments are conducted on the Aurora-2 database [29]. This database contains clean and noisy utterances, sampled at 8 kHz and filtered with a G712/MIRS characteristic. There are 8440 clean training samples, 2412 of which contain only one digit. We have tested our systems on the clean test data (4004 utterances, 13159 digits) as well as on the noisy test sets A-C. The latter sets were created by artificially adding noise to the clean test data at Signal-to-Noise Ratios (SNR) between 20 and -5dB (see [29]). The vocabulary consists of the digits 0 to 9 and 'oh' (a substitute for 'zero').

During system development, only the clean data is used, and the input features are the 39 MFCCs. The training set is divided into a learning set (about 2/3 of the train data) and a validation set (the remaining 1/3 of the train data). The split was made such that there is no speaker overlap between the two sets. The topological parameters (e.g., the reservoir size) and the control parameters (e.g., how to scale the inputs) are optimized by performing experiments with different parameter values, and by selecting the parameter value which minimizes the WER obtained on the validation set.

Once all the control parameters are set, the final system is trained by repeating the training on the full training set. Since we have no validation set anymore in this stage, we just apply the number of iterations that we usually needed during the development phase. That actually means 4 iterations during phase 1 of the training and 5 more during phase 2.

3.5.1 Setting the Control Parameters

Following our previous work on phoneme recognition [8], we worked with only 50 recurrent connections per reservoir node. As we experienced before that the regularization constant and the safety parameter are not that critical if the reservoir size is large, we did not try to optimize them. They were fixed to $\epsilon = 0.001$ and $y_o = 0.002$, respectively, values that also work well for phoneme recognition. For the input scaling, we also applied the same factors that we used for phoneme recognition.

The only parameters that were optimized for the digit recognition task are the spectral radius and the leak rate of the reservoir neurons. We did this because these parameters determine the dynamic behavior of the reservoir, and because the time scales of the phonemes and the digits are different. The optimization was performed with an isolated digit recognition system with a reservoir of 1000 nodes and a digit model with 3 states. We found a rather broad area in the (ρ,λ) plane where the results remain pretty stable, and we finally selected $(\rho, \lambda) = (0.8, 0.35)$ for all future experiments.

3.5.2 Setting the Topological Parameters

We performed three experiments to investigate two topological parameters: the reservoir size (number of reservoir nodes) and the number of states per digit (the same for all digits). We stick to the same number of states per digit because the reference HMM systems we want our system to compare with, also adopt this strategy. It would thus be unfair to optimize the number of states for each individual digit in our system. Recall that our silence model is always a single-state model.

Table 3.1 summarizes the most important results. In all experiments, the inter-digit transition probability was altered until a good balance between deletions and insertions was attained.

The first experiment shows that the WER decreases with the reservoir size, but it starts to saturate as soon as the number of trainable parameters is larger than 100K. The second experiment reveals that doubling the number of states per digit from 5 to 10 is less effective than doubling the size of the reservoir (compare the improvements in I and II). On the other hand, the third experiment shows that a two-state system only performs slightly worse than a five-state system with the same number of reservoir nodes (and therefore a larger number of trainable parameters).Apparently, we need less states than a GMM-HMM system (best results for 13-16 states), which supports the claim that the reservoir does model dynamical properties of the speech.

Exp.	Nodes	States per digit	Trainable Param.	WER%
I	500	5	27500	2.71
	1000	5	55000	1.72
	2000	5	110000	1.29
	4000	5	220000	1.21
II	500	5	27500	2.71
	500	10	55000	2.33
	2000	5	110000	1.29
	2000	10	220000	1.27
III	4000	2	92000	1.35
	4000	5	220000	1.21

Table 3.1: Validation results for systems with different reservoir sizes and number of states per digit. Also mentioned is the number of trainable parameters.

3.6 Results on the Noisy Test Sets

Once the optimal control parameters were fixed (including the inter-word penalty), we trained a new system with 4000 reservoir nodes and 5 states per digit, but this time we used all the available clean training data. Furthermore, as the raw MFCCs did not offer very good results on noisy speech, we worked with the noise-robust MSVA input features instead. Obviously, we also re-optimized the reservoir parameters for these new inputs. Figure 3.5 shows the average results (test sets A - C) of our system (RCN) as a function of the SNR, as well as those of the reference HMM system described in [111]. Apparently, the reservoir system competes well with the reference system, and for low SNRs it even tends to offer a small benefit.

For test set A, more results are being published in the literature. That is why we also compared our system on this test set with two GMM-HMM systems recently described by Gemmeke [112]: one system in which a Missing Data Technique (MDT), namely imputation, is employed, and another in which Exemplar-based Feature Enhancement (EB-FE) is applied on the MFCC parameters. These figures confirm that our conceptually simple system achieves the same noise-robustness as these much more complex approaches.

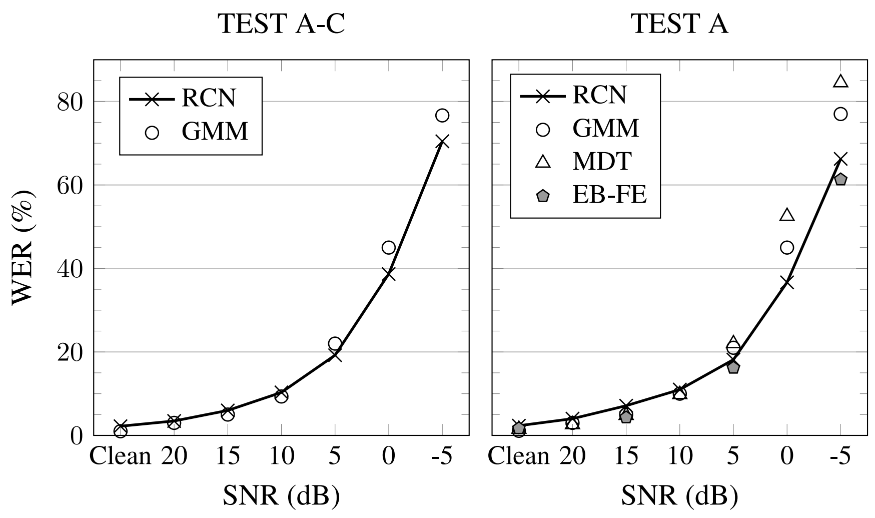


Figure 3.5: Recognition results (WER) as a function of SNR for the reference (GMM) and the reservoir system (RCN). For test A, also results obtained with Missing Data Techniques (MDT) and Exemplar-Based Feature Enhancement (EB-FE) are shown.

3.7 Conclusion and Future Work

We have shown that Reservoir Computing, a fairly recent paradigm developed in machine learning, can be applied to create a good continuous digit recognizer. In combination with noise-robust features, the system competes favorably with a traditional GMM-HMM system, even if the latter is combined with complex noise suppression techniques such as Missing Data Imputation and Exemplar-based Feature Enhancement.

Since we have thus far only spent limited time on the development of our system, we may be able to further improve it soon. We could, e.g., investigate stacked reservoir architectures like the ones we applied in our phoneme recognition work, or big reservoirs in combination with multiple readout vectors. These readout vectors would be defined in different sub-spaces of the reservoir state space, and readouts from different vectors could be allowed to compete with each other.

4 Multi-Layer RCN-HMM Hybrid and Adaptation to Noise

This chapter is an edited version of the following original publication:

[2] A. Jalalvand, F. Triefenbach, and J.-P. Martens, "Continuous digit recognition in noise: Reservoirs can do an excellent job!," *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, p. ID:644, 2012.

4.0 Preface

Although the results obtained in the previous chapter suggested competitive behavior of RCN and GMM-based systems on all noise conditions, there was still a significant gap between the two in the clean speech condition. In that case the GMM-HMM system attained a word error rate (WER) of less than 1% which is relatively about 50% lower than that of RCN-HMM hybrid. Furthermore, our original hope that the dynamical modeling capability would lead to more noise robustness was not yet substantiated.

Therefore, we conducted research that was intended to offer better clean speech as well as noisy speech recognition accuracy than a conventional GMM-HMM system. Furthermore, we investigated whether it would be possible to adapt an RCN-based recognizer to a particular noise environment in an unsupervised way and to achieve a significant gain in noise robustness in this way.

4.1 Introduction

Standard Hidden Markov Models (HMMs) incorporate Gaussian Mixture Models (GMMs) to compute state-level acoustic likelihoods. Such systems have reached a high level of performance, but they remain very sensitive to mismatches between the training and the test circumstances. Many research efforts have been directed towards the development of novel front-end and/or back-end techniques [112–114] for making the systems more resistant to these mismatches.

In this chapter an RCN-HMM hybrid for continuous digit recognition is investigated. The acronym RCN stands for Reservoir Computing Network [99]. It indicates that the transformation of input features to state likelihoods involves a reservoir, defined as a pool of non-linear and recurrently connected computational nodes (called neurons) with randomly fixed weights.

It was already demonstrated in our previous publication [1] that an RCN-HMM hybrid comprising one reservoir can yield good performance for isolated and continuous digit recognition in clean and noisy circumstances (tested on Aurora-2 [29]). However, for large signal-to-noise ratios (SNR) the hybrid was still outperformed by a traditional GMM-based system with the same input features.

In this chapter we report on how we succeeded in creating a new hybrid that either equals or surpasses the HMM system for all tested SNRs. Furthermore, we propose a reservoir-based method for adapting a hybrid that was trained on clean speech to work better in a noisy condition after it has seen some untranscribed digit string recordings representative of that condition.

4.2 Reservoir Computing Network

Figure 4.1 shows the architecture of a basic RCN. It is composed of one reservoir and a layer of linear nodes which 'read out' the reservoir nodes. The weights of the reservoir nodes are fixed (not trained) and drawn from a random distribution. Precautions are taken to guarantee that a stable dynamical system is obtained and that the new inputs and the previous outputs contribute in a balanced way to the new outputs of the reservoir nodes (see [8] for details). The weights of the readout nodes are trained so as to pursue that a particular node is high when the corresponding digit state is visited and low otherwise.

Suppose that U_t, R_t and Y_t, respectively, represent the input vector, the reservoir state vector and the output vector at time t and that the matrices

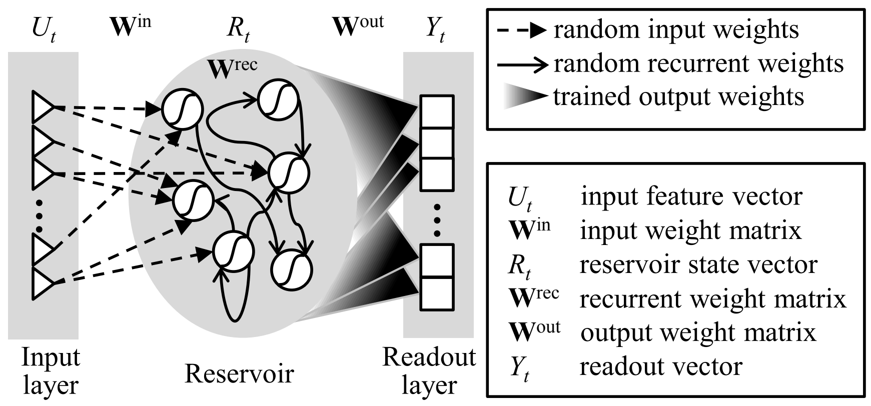

U_t
$\mathbf{W}^{\text{in}}$
R_t
$\mathbf{W}^{\text{out}}$
Y_t
$\mathbf{W}^{\text{rec}}$
random input weights
random recurrent weights
trained output weights
U_t input feature vector
$\mathbf{W}^{\text{in}}$ input weight matrix
R_t reservoir state vector
$\mathbf{W}^{\text{rec}}$ recurrent weight matrix
$\mathbf{W}^{\text{out}}$ output weight matrix
Y_t readout vector
Input layer
Reservoir
Readout layer

Figure 4.1: A basic RCN consists of a reservoir of fixed nonlinear & recurrently connected nodes and a set of trainable output nodes.

$\mathbf{W}^{in}$, $\mathbf{W}^{rec}$ and $\mathbf{W}^{out}$ comprise the weights of the various connections. Then the RCN system performs the following computations:

$$\begin{aligned} R_t &= f_{res}(\mathbf{W}^{in}U_t + \mathbf{W}^{rec}R_{t-1}), && (4.1) \\ Y_t &= \mathbf{W}^{out}R_t. && (4.2) \end{aligned}$$

with f_{res} being a non-linear function (in our case $f_{res}(x) = \tanh(x)$).

To further extend the integration of information over time, the memoryless reservoir nodes are replaced by Leaky Integrator Neurons (LIN) [98]. Equation (4.1) then changes to

$$R_t = (1-\lambda)\,R_{t-1} + \lambda\,f_{res}\left(\mathbf{W}^{rec}\,R_{t-1} + \mathbf{W}^{in}\,U_t\right)$$

with $0 < \lambda \leq 1$ determining the integration time.

4.2.1 An RCN-HMM Hybrid Decoder

The RCN-HMM hybrid we proposed in [1] encompasses a single-state model to consume the inter-digit silences and a left-to-right model with no skips and $S = 5$ states per digit. Since we will test our system on Aurora-2 we discern 11 digits (two variants of '0'), leading to a total of $11 \times 5 + 1 = 56$ HMM states. The joint probability of the acoustic feature sequence $\mathbf{U}$ and an HMM state sequence $\mathbf{q}$ is computed as

$$P(\mathbf{q}, \mathbf{U}) = \prod_{t=1}^{T} P(q_t|q_{t-1})\,P(U_t|q_t), \qquad (4.3)$$

where $P(U_t|q_t)$ is derived from the readouts Y_t (see further) and $P(q_t|q_{t-1})$ is a state transition probability.

The weights of the readout nodes minimize the mean squared distance between the output vectors Y_t and the corresponding desired output vectors D_t in a training set. The desired output vectors D_t point to the state q_t^* of the state sequence $\mathbf{q}^*$ that maximizes the aforementioned joint probability. One finds the weights by solving a set of linear equations (see [1]). However, since a retrained reservoir can lead to another q_t^*, the process is repeated a few times until convergence.

4.3 Improvements of the RCN-HMM Hybrid

Our research on RCN-based continuous phoneme recognition [8] demonstrated that the recognition can be improved by putting two to three basic RCNs in cascade. Apparently, the second RCN is capable of discovering regularities in the shortcomings of the first one, and so on. Even though the cascading approach did not lead to improved large vocabulary recognition, we wanted to test it for noise robust connected digit recognition. The architecture of the new RCN-HMM hybrid is depicted on Figure 4.2.

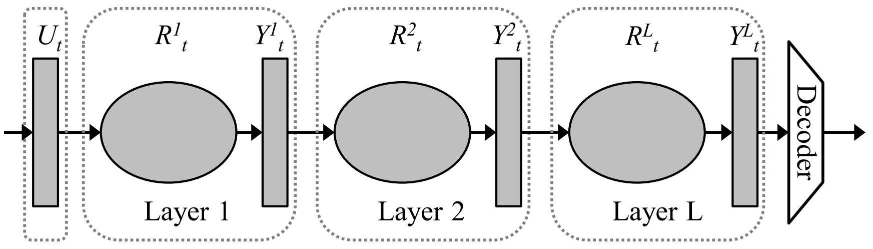


Figure 4.2: A multilayer RCN-HMM hybrid for continuous digit recognition. Each reservoir is stimulated by the readouts of the previous network.

The inputs to the first reservoir are the noise-robust MSVA features proposed in [113], plus their first and second order derivatives. The feature normalization is performed on complete utterances. The inputs to the second reservoir are the readout nodes of the first RCN, and so on.

4.3.1 Training a Multi-Layer Reservoir System

The training of a multilayer system proceeds layer per layer. The training of one layer is achieved by means of a Viterbi-training, as described in [1]. In a nutshell, it is an iterative process consisting of two steps per iteration:

(1) use the current system to align the input vectors U_t to the states q_t of an utterance model derived from the known digit string and (2) solve a set of linear equations to determine the weights of the readout nodes that minimize the mean squared error between the computed and the desired values. In [1] it is indicated how to bootstrap the training of the first layer.

4.3.2 Mapping Readouts to Likelihoods

In [1] we assumed that[1] the reservoir outputs $y_{t,q}$ ($q = 1 \ldots 56$) are good approximations of the posterior probabilities $P(q|U_t)$. Based on that assumption, we formerly used $y_{t,q}/P(q)$ as a proxy of the likelihood $P(U_t|q)$ needed in Equation (4.3). However, when measuring the posterior probabilities $P(q|Y_i)$ on a development set, it turned out that the aforementioned assumptions are violated. Therefore, we now retrieve for each state q a lookup-table to map $y_{t,i}$ to a better estimate of $P(q|U_t)$.

The lookup table for state q is derived from a histogram of the $y_{t,i}$ of the frames of a development set that were assigned to state q and from the global histogram of all the $y_{t,i}$. Starting with bins of 0.01 wide, the bins collecting too few examples in the global histogram are joined with their neighbors: the least populated bin is joined with the least populated neighboring bin and the process is continued until all bins contain enough examples (we used 100 examples for this).

Although the new mapping could have been applied to each readout layer, it was only applied to the last one as only these readouts have to be converted to likelihoods.

4.3.3 Experimental Validation

The proposed modifications were validated on the Aurora-2 benchmark [29].

The readout layers were always trained on clean data (8440 utterances from 110 speakers) but tests were performed on clean and noisy data. We report mean results over test sets A - C and use the word error rate (WER) in percent as the performance measure. We compare the RCN-HMM to a reference GMM-HMM embedding digit models with 16 states and GMMs with three mixtures in each state.

Table 4.1 shows the performances as a function of the number of layers, the number of reservoir nodes per layer and the activation/inhibition of the new output mapping. Per row, the clean speech result, the average result over SNRs between 20 and 0 dB, and the result for SNR = -5 dB are provided. The main findings are that doubling the size of a single layer

[1]making abstraction of a trivial linear mapping

nr of layers	nr of nodes	new mapping	Clean	0-20dB	-5dB
GMM-HMM [111]			**0.98**	16.86	76.66
1	4k	-	2.56	17.55	66.92
	4k	y	2.32	15.68	64.00
	8k	-	2.04	16.42	65.73
2	4k	-	1.37	14.68	61.17
	4k	y	1.29	**13.73**	**59.34**
3	3k	-	1.25	14.99	60.36
	4k	-	1.20	14.49	60.41
4	2k	-	1.46	16.49	61.79

Table 4.1: Continuous digit recognition results (average WER over Test A-C of Aurora-2) for systems with different topologies, sizes and probability mappings.

system is not as helpful as adding an extra layer and that non-linear readout mapping is always beneficial.

Adding one layer to the baseline (which is 1 layer, 4k nodes, old mapping) leads to a large gain (46% relative) for clean speech and significant gains for noisy speech. Adding a third layer further improves the clean speech result, but it does not bring anything extra for the noisy conditions. From Figure 4.3 it follows that for high SNRs the two-layer system competes well with the GMM-based system now, whereas for low SNRs it is consistently better.

4.4 Model Adaptation in RCN-HMM Hybrids

Now that we have a good system, we can try to improve it for a certain condition by adapting it to that condition. The aim is to collect a limited amount of untranscribed recordings made in that condition and to adapt the clean reservoir model in such a way that it performs better on test data recorded in that same condition.

4.4.1 Linear Transformation of the Outputs

A very simple approach is to train a linear transformation of the readouts Y_t to new outputs $Y'_t = \mathbf{A}\,Y_t + b$. The aim is to minimize the mean squared distance between the new outputs and the desired outputs in the adaptation

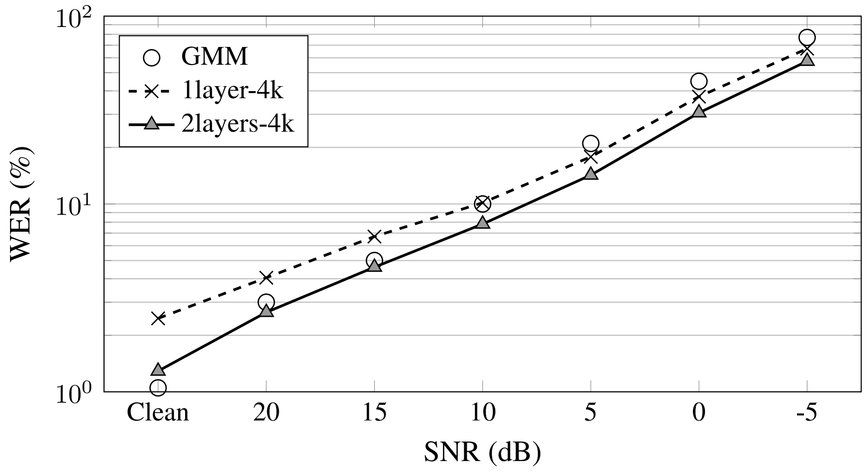


Figure 4.3: Recognition results (WER) as a function of SNR (Test A) for the reference (GMM-HMM), the single layer reservoir system (1layer-4k) and the two-layer system with non-linear output mapping (2layers-4k)

data. As the desired outputs are a priori unknown, they are identified from the states on the most likely path through the digit loop utterance model. Strategies for computing digit confidences and for retaining only the desired outputs corresponding to digits which were recognized with sufficient confidence, did not lead to a better result and are therefore not further considered here.

4.4.2 Retraining the Output Weights

Since the readout nodes are linear, the linear transformation of readouts is equivalent to a linear transformation of the readout node parameters (weights). In order to learn the latter transformation one can see the problem as one of training the readouts with the original training data supplemented with the adaptation data. If R and R_A are matrices whose columns represent the reservoir states of the N^{frm} training and N_A^{frm} adaptation frames, and if $\mathbf{D}$ and $\mathbf{D}_A$ represent the corresponding desired outputs (one column per frame), then the adapted readout weights can be obtained as

$$\begin{aligned}\mathbf{W}_A^{out} &= (\mathbf{R}^T\mathbf{R} + \alpha\mathbf{R}_A^T\mathbf{R}_A + \epsilon\,(N^{frm} + \alpha N_A^{frm})\,\mathbf{I})^{-1} \\ &\quad (\mathbf{R}^T\mathbf{D} + \alpha\mathbf{R}_A^T\mathbf{D}_A) \qquad (4.4)\end{aligned}$$

where α is a factor that controls how much the adaptation data contribute to these weights.

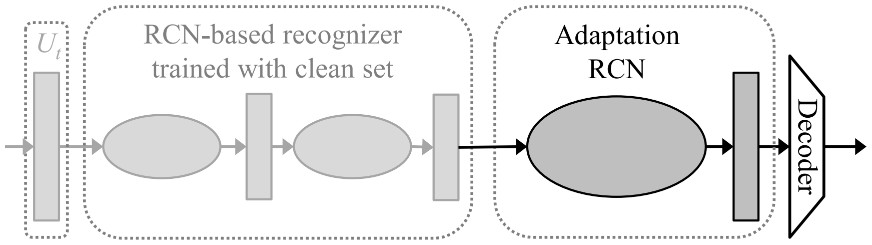


Figure 4.4: An extra layer is trained on the adaptation data. The adaptation RCN is trained using the frame labels provided by the RCN-based recognizer.

The problem with this method is that it needs access to the quite large matrices $\mathbf{R}^T\mathbf{R}$ and $\mathbf{R}^T\mathbf{D}$ and that the inversion of a large matrix is time consuming. So the method is merely used as a reference method against which to compare the other methods.

4.4.3 Adding an Extra RCN

Instead of learning a linear transformation one could also add another RCN layer (see Figure 4.4) that is solely trained on adaptation data. This reservoir then achieves a non-linear transformation with memory. The number of trainable parameters is equal to the number of eligible states times the number of nodes in the added reservoir. In a practical implementation one could create a sufficiently large reservoir and connect more of its nodes to the readout nodes as more adaptation data become available. This way the number of trainable parameters can be gradually increased.

Since the construction of a good lookup-table for mapping readouts to posterior probabilities requires a lot of data, we keep the non-parametric readout mapping for the output of the recognizer and we exploit the adapted readouts in the traditional way (i.e., the simple straightforward mapping).

4.4.4 Experimental Validation

We conducted experiments with 3 minutes of adaptation data taken from the development set. This way we could continue to use the same test sets as before. Table 4.2 shows the performances for Test A with the optimal α (method 2) and the optimal adaptation reservoir size (method 3). The latter was equal to 250.

The main conclusions are that method 3 works as well as method 2 and that it offers a promising gain for SNRs of 20 and 15 dB. Obviously, there

SNR (dB)	WER%	Relative improvement (%)		
	Baseline	Transform	Retrain	Add Layer
Clean	1.48	–8	0	7
20	2.68	14	18	19
15	4.69	19	29	31
10	8.03	5	7	4
5	14.79	9	13	9
0	30.08	–1	3	0
–5	55.66	–8	–5	–8

Table 4.2: Performance of the baseline RCN-based recognizer on Test A and relative improvements obtained with the three adaptation approaches.

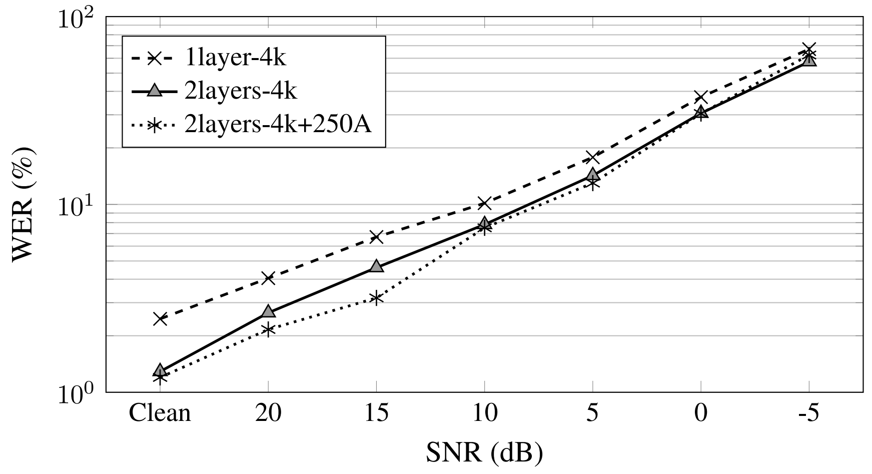


Figure 4.5: Recognition results (WER) for the two-layer system with and without an extra adaptation reservoir of 250 nodes.

is no improvement for clean speech as this condition already matches with the training condition.

The fact that there is no improvement for SNR $\leq$ 10 dB anymore, is owed to the fact that the desired outputs retrieved by the decoder become unreliable for these low SNRs.

In Figure 4.5 the performances of a two-layer system with 4K nodes per layer, with and without an additional adaptation reservoir with 250 nodes, are compared to one-another. The figure shows that the adapted system consistently outperforms the unadapted system as long as there is not too much noise. Obviously, it cannot improve on clean speech because the

System	Baseline WER%	After Adaptation	Relative Improvement
1-layer-4K	19.2	15.7	18%
2-layer-4K	15.3	13.9	10%

Table 4.3: Performance of the baseline RCN-based recognizers on a self-developed Flemish accent digit spoken dataset along with the impact of RCN-based adaptation.

original models were trained for that condition, but apparently, it does not degrade when the SNR becomes low and the labels driving the training of the extra adaptation RCN become unreliable. In summary, the data show that the adaptation RCN can help to maintain a low WER (say lower than 2%) over a larger SNR range.

4.4.5 Evaluation on Self-Developed Flemish Accent Dataset

[This section was not in the original paper, because the experiment was conducted after submitting the paper.]

In order to evaluate the performance of RCN-based digit recognition system and also the impact of adaptation to an unseen condition, a new dataset was collected by asking 80 students (male and female) to record their voices while pronouncing the English digits in real conditions. After pruning invalid data, we were able to provide a diverse dataset containing 800 samples with different types and levels of noise recorded at various places (e.g., home, lab, street), with different microphones and background noises, and obviously, with different Flemish accents. I used 600 samples for evaluation and kept the remaining 200 samples for adaptation. The results of evaluating both pre-trained single layer and 2-layer RC-based recognizers with 4K nodes per layer are listed in Table 4.3. Considering the fact that the conditions are more realistic than the Aurora-2 dataset, e.g., even microphones are different, the performance is not far from the published results.

Moreover, I trained the adaptation RCN using the remaining 200 samples and evaluated the adapted system. The experiments showed about 18% and 10% relative improvement on the single layer and 2-layer systems, respectively. These experiments proved that the promising performance of the RCN-based recognition and adaptation is not only limited to the laboratory provided data, but expands to real data, as well.

4.5 Conclusion and Future Work

We proposed several improvements of a formerly presented RCN-HMM hybrid for continuous digit recognition. The new system now surpasses a GMM-HMM system that is supplied with the same noise robust acoustic features. Moreover, a system trained on clean speech can be adapted by means of a reservoir-based method to a new system that works better in another condition (noise type and SNR). The adaptation is performed in an unsupervised way on the basis of a limited amount (3 minutes) of adaptation data. The improvement is only significant (up to 30% relative) though as long as reliable transcription for the adaptation data can be generated by the non-adapted decoder. Our future goal is to investigate whether the proposed adaptation method also works satisfactory for, e.g., accent-specific recognition where more adaptation data are likely to be available.

5 On Optimizing Reservoir Parameters for Speech Recognition

This chapter is an edited version of the following original publication:

[3] A. Jalalvand, F. Triefenbach, K. Demuynck, and J.-P. Martens, “Robust continuous digit recognition using reservoir computing,” *Computer Speech and Language (CSL)*, vol. 30, no. 1, pp. 135 – 158, 2015.

5.0 Preface

In the two former chapters we showed that it is possible to achieve good continuous digit recognition with big enough reservoirs and with RCN layers stacked on top of each other. However, it took a lot of experimentation time to construct these systems. In fact, the optimal settings of the reservoir parameters appeared to depend on the RCN architecture, the number of inputs, the number of states per digit, etc. Although I gradually developed a good intuition for how to change the reservoir parameters when the properties of the RCN inputs or the targeted RCN outputs changed, I was still a bit frustrated that I could not establish appropriate reservoir parameters without having to perform a number of time consuming pilot experiments first.

As the conclusion of the former papers seemed to be that bigger reservoirs and better system architectures might lead to more accurate and noise-robust systems, I anticipated that it would take many more experiments to identify the most appropriate architecture. Therefore, I got convinced that it would pay off to search for a comprehensible procedure to fix the reservoir parameters to quasi-optimal values in an automatic way. In the present chapter, I present a novel analysis of an RCN as a non-linear dynamical system and I postulate some sensible heuristics that finally allowed me to distill a number of empirical relations between the reservoir parameters and the RCN input and output dynamics. These relations have finally resulted in a recipe that is extremely simple and highly comprehensible.

Following this recipe I then started to investigate big systems, multiple system architectures and multiple training approaches (e.g., multi-style training) and was able to devise new systems showing higher accuracy in matched conditions and more graceful degradation in mismatched conditions than most other systems, including the ones I developed thus far myself.

5.1 Introduction

Despite many years of research, devising an Automatic Speech Recognition (ASR) system that correctly interprets a naturally spoken utterance captured in realistic conditions, is still a big challenge. Significant improvements are still needed before solutions will be available for voice driven applications with strict specifications such as high accuracy and robustness against confounding factors. In this work, we are concerned with continuous digit recognition (CDR). Although somewhat less elaborate than large vocabulary continuous speech recognition, CDR is important because many voice driven applications require the user to provide numeric data (telephone numbers, PIN-codes, account numbers, coordinates, etc.), and providing such information in the form of a spoken digit sequence is quite natural. However, since the sequences are often long, a very high accuracy is needed at the individual digit level to attain a high accuracy at the sequence level. Furthermore, many applications involving CDR must accommodate native as well as accented non-native speakers (e.g., tourists) communicating through mobile devices. The latter means that the speech exhibits a lot of channel variability and that it is often captured in a noisy environment (e.g., on the street).

The CDR problem is commonly formulated as a statistical pattern recognition problem. First, an acoustic front-end converts the raw speech signal into a compact feature representation. Then, a back-end employs stochastic models to retrieve the most likely digit sequence given this representation. The feature representation consists of consecutive acoustic feature vectors representing short fixed length speech slices, called frames. In most cases, the back-end contains one stochastic model per digit, usually a left-to-right multi-state Hidden Markov Model (HMM), with each state having its own model for estimating the likelihood of an observed feature vector being generated by that state. If standard short time spectral features are employed in combination with acoustic models trained on low-noise training utterances, high accuracies (less than 1% of the digits wrong) can be achieved on similarly low-noise test utterances. However, severe degradations occur when the test utterances are noisy: more than 30% errors at a signal-to-noise ratio (SNR) of 10 dB [29]. Consequently, a tremendous effort went into the development of robust speech recognition methods, in particular, techniques for making CDR more resistant to the presence of various types of noises at arbitrary SNRs.

The research has led to a multitude of techniques that can roughly be categorized into four classes. First of all, there is a class of front-end methods that aim at retrieving 'clean' speech feature vectors from the observed noisy vectors so as to employ these cleaned vectors in an otherwise conventional system. Examples of this approach are described in [31, 92, 113, 115–118]. An alternative approach is to hold on to the raw noisy features, but to take the hypothesized impact of the noise into account during the likelihood computations in the back-end. Examples of such model-based approaches are [36, 119–124]. These approaches are in general more powerful than the feature-based approaches, at the expense of being far more computationally taxing. A third option is to adjust the parameters of pre-trained models during actual operation on the basis of recognized outputs and confidence measures computed for these outputs. Examples of such model adaptation techniques can be found in [125–128]. Finally, one can also substitute conventional GMM-HMM systems – in which state likelihoods emerge from state-dependent Gaussian Mixture Models (GMMs) – by hybrid systems encompassing neural networks that are trained to compute posterior state probabilities [129]. Given that no single class of methods is optimal in all respects, most of the solutions presented in the literature incorporate elements of multiple methods.

The present paper builds on our former work on acoustic modeling for continuous speech recognition (phones as well as digits) by means of Reservoir Computing (RC), a technique originally introduced in [80, 99]. Although RC was already applied to isolated digit recognition some time ago [108, 109], we were the first [1, 2] to demonstrate that it offers noise robust CDR on an internationally accepted benchmark.

Reservoir computing is performed with so-called Reservoir Computing Networks (RCNs) which can be considered as a special kind of recurrent neural network, namely one in which not all the parameters are trained (see later). Based on a novel analysis of such an RCN as a non-linear dynamical system, we gained new insight in how to tune its dynamical properties to the observable *input dynamics*, defined as the dynamics of the acoustic features, and to the anticipated *output dynamics*, defined as the dynamics of the variables that are derived from the reservoir outputs to represent the different speech units. By conducting artificial digit state recognition experiments, we could turn these insights into powerful heuristics which significantly facilitate the design of an appropriate reservoir system. The RCN-based systems created in this way compete well with conventional systems in clean conditions and degrade more gracefully in noisy conditions. Control experiments show that the noise robustness mainly follows from the random fixation of the reservoir neurons but that properly tuning the reservoir dynamics is indispensable for combining this with a high accuracy in matched conditions.

In the reminder of this paper, we first review the Aurora-2 experimental framework and discuss formerly proposed solutions to noise robust CDR that were assessed using this framework (Section 5.2). Next, we introduce the fundamental principles of Reservoir Computing (Section 5.3) and explain how to apply it in a speech recognizer (Section 5.4). Then, we elaborate and validate the novel reservoir analysis method and the reservoir design method that is derived thereof (Sections 5.5 and 5.6). In Sections 5.7 and 5.8, we provide compelling evidence of the good CDR performance of the designed systems. The paper ends with some conclusions.

5.2 Continuous Digit Recognition and Aurora-2

The Aurora-2 experimental framework [29] was designed to bolster research on noise robust continuous digit recognition. In this section, we briefly introduce the framework and review a number of methods that have been evaluated using this framework.

5.2.1 The Aurora-2 Framework

The Aurora-2 corpus consists of clean and noise corrupted digit sequences counting 1 to 7 digits per utterance. Each utterance is passed through a G712 or a MIRS filter (see Figure 5.1), and then sampled at 8 kHz [29]. The G712 filter has a flat response in the range from 300 to 3400 Hz and is characteristic of a fixed line telephone connection. The MIRS filter exhibits a rising characteristic which is more characteristic of a mobile link [29, 130]. Since there are two variants of ‘0’ in American English, namely *zero* and *oh*, the vocabulary consists of 11 digits.

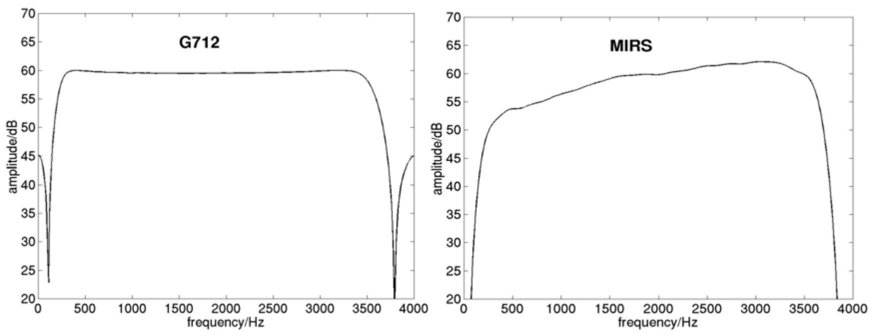


Figure 5.1: Frequency responses of G.712 and MIRS filter [29]

	Train & Test A	Test B	Test C
Noise types	subway babble car noise exhibition hall	restaurant street airport train station	subway street
Filter	G712	G712	MIRS

Table 5.1: Noise types and filters used in different Aurora-2 sets

The data is divided into a training part and an evaluation part. The framework supports two types of experiments: *clean training experiments* in which systems are developed on 8440 clean training utterances from 110 adults, and *multi-style training experiments* in which systems are developed on 8440 noise corrupted versions of the same utterances. The corruptions cover four noise types and five SNRs (∞ (clean), 20, 15, 10 and 5 dB). The evaluation utterances come from speakers that are not present in the training data. They are divided into three test sets; Tests A and B each contains 28,028 utterances covering 4004 different digit sequences, 4 noise types and 7 SNRs (∞ (clean), 20, 15, 10, 5, 0, and -5 dB). The noise types occurring in Test B do not occur in the multi-style training data, while those of Test A do. Test C contains 14,014 utterances covering 2002 different digit sequences, 2 noise types (one matched and one mismatched) and 7 SNRs. Unlike all other utterances, they are passed through a MIRS instead of a G712 filter (see Table 5.1).

Method	Clean	0-20dB	-5dB
GMM-HMM (ETSI)	0.97	38.9	91.5
GMM-HMM (MVN)	0.84	19.7	82.2
GMM-HMM (AFE)	0.77	13.0	69.7
GMM-HMM (VTS)	0.40	(7.3)	-
GMM-HMM (MDT)	-	11.4	-
GMM-HMM (EB)	0.40	(5.6)	(45.8)
GMM-HMM (D-HMM)	0.51	36.0	90.0
GMM-HMM (UD)	-	10.3	-
DBN-GMM-HMM (Tandem)	1.26	21.0	74.6

Table 5.2: WERs (in %) obtained in clean speech training experiments on Tests A - C of Aurora-2. Results between brackets are obtained by employing a noise dictionary and are biased towards the case of multi-style training.

5.2.2 Methods That Were Tested on Aurora-2

The abundance of papers on noise robust digit recognition makes it impossible to present an exhaustive review. Our aim is, therefore, to review methods that were tested on Aurora-2 and that, in our opinion, provide a good image of what state-of-the-art CDR systems can achieve. The mostly used error measure is the average word error rate (aWER) – defined as an average over SNRs 0, 5, 10, 15 and 20 dB – obtained in clean training experiments.

5.2.2.1 Baseline System (ETSI)

At the time the Aurora-2 campaign was launched, a baseline system was made available as a reference. It consisted of the ETSI standard front-end for generating the Mel-Frequency Cepstral Coefficients (MFCCs) proposed in [16] and of a conventional GMM-HMM recognizer with 16-state whole word models embedding a GMM in each state. The models were trained using Maximum Likelihood Estimation (MLE). In a clean speech training experiment, this baseline system yields a poor performance in the presence of noise [29]: the aWER is nearly 39% (see Table 5.2).

5.2.2.2 Mean and Variance Normalization (MVN)

By means of utterance-based normalization of the MFCCs to zero mean and unit variance variables, it is possible to reduce the aWER to 20% with a negligible extra computational load [26].

5.2.2.3 Advanced Front-End (AFE)

By putting two additional blocks in front of the MFCC front-end, namely an adaptive Wiener filter applied to the raw speech signal and a voice activity detector (VAD) block performing some SNR dependent processing of the filtered signal, a large gain in noise robustness with respect to the raw MFCCs can be attained [131]. With this so-called ETSI advanced front-end (AFE), the aWER can be reduced to 13%.

5.2.2.4 Vector Taylor Series Adaptation (VTS)

While front-end methods have shown improved performance on several tasks, they all, by definition, make point-estimates of the clean speech features. Errors in these estimations can cause further mismatch between the features and the acoustic model, resulting in degraded performance. Model adaptation techniques avoid this problem by directly compensating the probability distributions employed by the recognizer.

One example of such an approach is vector Taylor series (VTS) adaptation [122, 125, 132, 133] which improves the recognition in clean as well as in noisy conditions. As it is not fair to compare noise adapted models with models that originate from clean speech training alone, the VTS result is mentioned between brackets in Table 5.2.

5.2.2.5 Missing Data Techniques (MDT)

In [31], a so-called Missing Data Technique (MDT) was proposed. It detects cells of a spectrogram-like time-frequency representation that have become unreliable (or missing) due to noise masking. The values in these cells are then substituted by marginal values [134]. This technique has been continuously improved since its introduction, and an aWER of 11.4% is reported in [135].

5.2.2.6 Exemplar-Based Systems (EB)

Exemplar-based systems rely on the assumption that an arbitrary fragment of e.g., 30 frames from the test utterance of a digit can be represented as a sparse linear combination of suitably selected speech and noise fragments stored in a speech and noise dictionary, respectively. By retaining just the speech components from that combination, one can create an enhanced MFCC stream. Since its introduction in [136], the technique has continuously been improved in order to make it better and faster, and a general review is presented in [137]. From [138], we derive that in combination with acoustic models trained on clean speech only, the aWER for Test A + B (no results for test C) can be reduced to about 6%. However, this result cannot be compared to the result of a clean speech training experiment, because the feature enhancement is obtained with the help of a dictionary of noise samples.

5.2.2.7 Discriminative HMMs (D-HMM)

Traditionally, the parameters of an HMM are trained using MLE, but discriminative training schemes like Maximum Mutual Information (MMI) as proposed in [139, 140] can significantly outperform MLE. However, in [141] it is demonstrated that discriminative training only helps in matched conditions. Using the standard ETSI features, the discriminatively trained HMMs (D-HMM) do very well for clean speech, but they achieve only a minor improvement over the baseline system for noisy speech: the aWER is equal to 36%. We found no figures about D-HMMs in combination with the AFE, but there is no reason to expect a major improvement in that setting either.

5.2.2.8 Uncertainty Decoding (UD)

An appealing approach to the robust recognition, called uncertainty decoding (UD), is proposed in [36, 37]. In UD, there is a model for estimating the amount of uncertainty about the features. During decoding, this model is used to replace the actual observation by a distribution and to compute state likelihoods by integrating over the feature space. Using a model-based joint uncertainty decoding technique (JUD) the aWER can be reduced to 10% [124].

5.2.2.9 Deep Neural Network-Based Approaches (DNN)

Many research groups have conceived systems embedding a discriminatively trained neural network. The best results were obtained with a so-called *tandem* which considers the neural network outputs as a new type of acoustical features replacing the MFCCs in an otherwise conventional GMM-HMM. With a tandem employing a Deep Neural Network (DNN) as the neural component, an aWER of 21% was obtained [142, 143].

5.3 Reservoir Computing Networks (RCN)

Even though the results obtained with neural network-based and discriminatively trained models discussed in the previous section were not so promis-

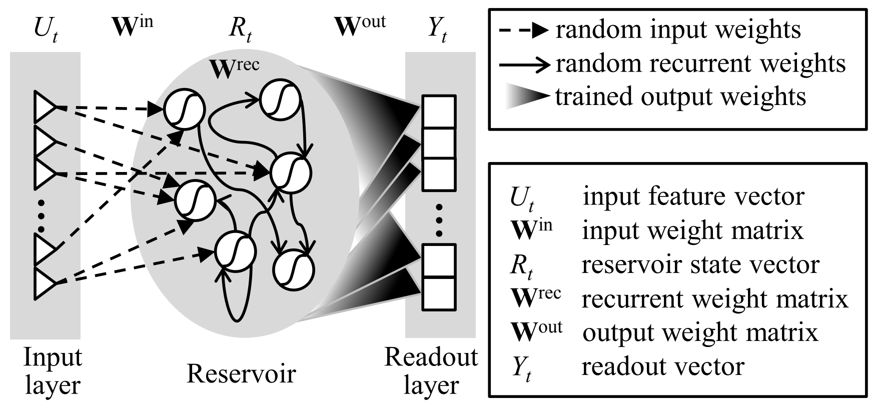

U_t
$\mathbf{W}^{\text{in}}$
R_t
$\mathbf{W}^{\text{out}}$
Y_t
$\mathbf{W}^{\text{rec}}$
Input layer
Reservoir
Readout layer
random input weights
random recurrent weights
trained output weights
U_t input feature vector
$\mathbf{W}^{\text{in}}$ input weight matrix
R_t reservoir state vector
$\mathbf{W}^{\text{rec}}$ recurrent weight matrix
$\mathbf{W}^{\text{out}}$ output weight matrix
Y_t readout vector

Figure 5.2: A basic RCN system consists of a reservoir and a read-out layer. The reservoir is composed of interconnected non-linear neurons with randomly set weights. The readout layer consists of linear neurons with trained weights.

ing, we started to investigate another neural approach which is based on Reservoir Computing [98]. We did so mainly because this technique offers an attractive way of taking the speech dynamics into account.

The basic principle of RC is that information can be retrieved from sequential inputs by means of a two-layer Recurrent Neural Network (RNN) with the following characteristics (see Figure 5.2). The first layer is a hidden layer composed of non-linear neurons which, at time t are driven by actual inputs U_t and delayed hidden layer outputs R_{t-1}. The hidden neurons, also, have randomly fixed coefficients. The second layer consists of linear neurons which are driven by actual hidden layer outputs R_t and which have trainable coefficients. The recurrently connected hidden neurons can be imagined as a pool of interconnected computational neurons, excited by inputs. Such a pool is called a *reservoir*. Together with the linear output neurons, it forms a *Reservoir Computing Network* (RCN). The network outputs Y_t are usually called *readouts* [99] to differentiate them unambiguously from the reservoir outputs R_t.

The reservoir can perform a temporal analysis of the input stream. In order to be effective, it should have the so-called echo state property [99]. The latter states that, with time, the reservoir should forget the initial state it was in. This corresponds to the requirement that a linear filter should have an out-fading impulse response to be suitable for performing a meaningful short-term analysis of a non-stationary signal such as speech. As each reservoir output is the output of a non-linear filter with multiple inputs and as the filter coefficients are chosen randomly, a big enough reservoir will give rise to a large variety of filters. By training the readouts on speech, they will focus on the outputs of those filters that resonate to frequencies which are typical for the dynamics of the speech signal. This leads to the hypothesis that an RCN can filter-out noise-inflicted dynamics (modulations) whose frequencies are a-typical for speech.

5.3.1 Reservoir Network Equations

If there are N^{in} input features and N^{out} readouts and if the reservoir consists of N^{res} neurons, the reservoir network is governed by the following equations:

$$\begin{aligned} R_t &= f_{res}(\mathbf{W}^{in}U_t + \mathbf{W}^{rec}R_{t-1}), && (5.1)\\ Y_t &= \mathbf{W}^{out}R_t. && (5.2)\end{aligned}$$

The $N^{res} \times N^{in}$ matrix $\mathbf{W}^{in}$ and the $N^{res} \times N^{res}$ matrix $\mathbf{W}^{rec}$ comprise the weights of the input connections and the recurrent connections, respectively. $f_{res}(\cdot)$ is an activation function performing component-wise non-linear transformations of the neuron activations (e.g., $f_{res}(\cdot) = \tanh(\cdot)$). In order to compute Y_t, we actually extend R_t with a bias of 1. It can be shown [99] that the echo state property holds if the *spectral radius* ρ, defined as the maximal absolute eigenvalue of $\mathbf{W}^{rec}$, is smaller than 1.

In order to deal better with the random inter-frame changes observed in the inputs (e.g., due to the spectral analysis), one can introduce leaky integration by replacing Equation (5.1) with

$$R_t = (1-\lambda)R_{t-1} + \lambda\, f_{res}(\mathbf{W}^{in}U_t + \mathbf{W}^{rec}R_{t-1}), \qquad (5.3)$$

with $0 < \lambda \leq 1$. In this case, the reservoir neurons are called Leaky Integrator Neurons (LINs) [100] with a leak rate λ. All reservoirs in this paper employ neurons of this type. The dynamical behavior of the reservoir can be expressed in terms of two time constants:

$$\tau_\rho \doteq -\tau_{fr}/\ln(\rho) \quad \text{and} \quad \tau_\lambda \doteq -\tau_{fr}/\ln(1-\lambda), \qquad (5.4)$$

with τ_{fr} being the time-shift (in ms) between frames. It will become clear that τ_ρ determines the memory capacity of the reservoir, whereas τ_λ determines how smooth the readout *patterns* (i.e., temporal evolutions) are.

5.3.2 Fixing the Reservoir Weights

Without any loss of generality, we assume that the reservoir inputs all have zero means and identical variances. In that case, drawing all the input weights from the same zero-mean normal distribution is the logical thing to do to ascertain that every input has the same chance of affecting the reservoir output. Therefore, we draw the input weights from a single distribution with variance α_U^2 and we call α_U the input scaling factor as it controls how strongly the inputs excite the reservoir. In a similar vein, we draw all recurrent weights from a zero-mean normal distribution with variance α_R^2. In that case, the reservoir outputs are also bound to have similar distributions and a similar chance to excite the reservoir neurons. Obviously, α_U and α_R can be used to control (1) the relative importance of the recurrent and the input connections and (2) the level of excitation of the reservoir neurons.

As we will see later, instead of working with full weight matrices, we randomly select K^{in} entries per row of $\mathbf{W}^{in}$ and K^{rec} entries per row of $\mathbf{W}^{rec}$ and we only initialize those entries, leaving the other ones zero. This restriction reduces the amount of computations per time step. Unless K^{rec} is close to 1, the squared norm of every row of $\mathbf{W}^{rec}$ is approximately equal to $K^{rec}\,\alpha_R^2$. This means that $\rho^2 \approx K^{rec}\,\alpha_R^2$, and thus, K^{rec} and ρ can be considered as the independent reservoir parameters instead of K^{rec} and α_R.

5.3.3 Training the Output Weights

The aim of the training is to find the output weights that minimize the mean squared difference between the readouts Y_t and their desired values D_t across N^{frm} available training examples. Introducing the matrices **R** and **D** with columns R_t and D_t respectively, the output weights are the solution of a regularized Tikhonov regression problem [144]:

$$\hat{\mathbf{W}}^{out} = \underset{\mathbf{W}^{out}}{\arg\min} \left(\frac{1}{N^{frm}} \left|\left|\mathbf{W}^{out}\mathbf{R} - \mathbf{D}\right|\right|^2 + \epsilon \left|\left|\mathbf{W}^{out}\right|\right|^2 \right), \tag{5.5}$$

with ϵ being the regularization parameter. The latter is intended to prevent over-fitting to the training data. The solution is obtained in a closed-form [110] as

$$\hat{\mathbf{W}}^{out} = (\mathbf{R}\mathbf{R}^T + \epsilon\,\mathbf{I})^{-1}(\mathbf{D}\,\mathbf{R}^T), \tag{5.6}$$

with **I** representing the identity matrix and $\mathbf{A}^{-1}$ the Moore-Penrose pseudo inverse of **A** [145].

In our experiments, the desired output D_t is always a unit vector referring to the desired HMM-state at time t (see Section 5.4). Furthermore, we observed that including a regularization term did not improve our results. We owe this to the nature of speech (a lot of intrinsic variability) and to the fact that cutting the speech into short frames that are analyzed independently introduces a random variation that affects the equations in very much the same way as the regularization term does. The bottom line is that we did not include a regularization term, i.e., we set $\epsilon = 0$.

5.3.4 Arguments in Favor of Reservoir Systems

An RCN can be considered as an extension of the Extreme Learning Machine (ELM) proposed in [101,146]. The ELM is defined as a feed-forward neural network (a Multi-Layer Perceptron or MLP) with a randomly fixed hidden layer and a linear output layer whose weights are fixed to minimize the mean squared difference between the computed and the desired outputs. In [101], it is mathematically proven that the ELM is as powerful as a fully trained MLP: a system with N hidden neurons can learn exactly N distinct observations. Moreover, it is shown that the ELM generalizes best to unseen data because it employs the least square solution of the Tikhonov regression problem. The various experiments presented in [101] lead to the following conclusions: (1) in terms of generalization ELM behaves as well as a Support Vector Machine (SVM) employing a linear kernel [105], and much better than a fully trained MLP, (2) an ELM is much more compact than an SVM (it needs less hidden neurons than an SVM needs support vectors) and (3) the generalization performance of an ELM remains stable over a wide range of hidden units. We argue that good generalization to test data should transfer to good noise robustness. Moreover, we expect that introducing recurrent connections which are also randomly fixed, is bound to maintain the noise robustness while improving the model accuracy. Adding these arguments to the formerly mentioned noise filtering capacity of an RCN, we have enough reasons to believe in the high potential of RCN for noise robust CDR.

An RCN also resembles an SVM, but one with a hidden space whose size and identity do not follow from a long and delicate supervised training process. Likewise, it resembles a radial basis function (RBF) network using a fixed hidden layer. However, the RBFs often follow from a clustering procedure and they typically represent local functions in the input feature space, meaning that they only react to inputs that fall in a restricted area of the input space. A reservoir neuron, on the other hand, typically exhibits a non-local activation function. Finally, an RCN resembles the recurrent neural network (RNN) applied in [106] for continuous speech recognition. However, the memory of that network originated from feeding the outputs back to the hidden layer and, importantly, all the network weights were trained by means of back-propagation through time, a method that is found to be very time consuming and likely to yield a sub-optimal solution when the size of the network is large. In an RCN, the optimal weights are found in a straightforward manner, even for a reservoir with several thousands of neurons.

5.4 A Basic RCN-HMM Hybrid for CDR

The architecture of an RCN-HMM hybrid for CDR is depicted in Figure 5.3. The front-end generates acoustic inputs U_t and the readouts Y_t

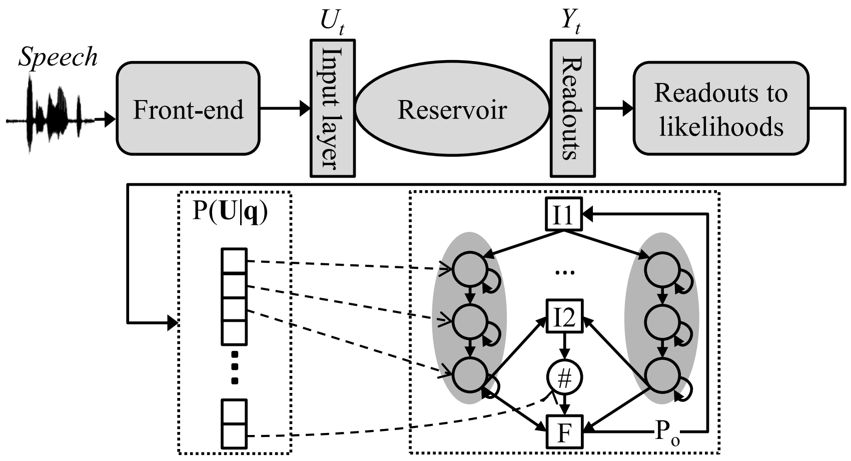


Figure 5.3: Architecture of an RCN-HMM hybrid for CDR. The HMM has two initial states (I1 and I2), one final state (F) and comprises 11 multi-state digit models with a single state silence model (#).

are converted to state likelihoods (see further) before they are supplied to a looped HMM that models the digits with multiple states and the silence with a single state. There is only one transition probability, P_o, namely on the transition from the final state to one of the two possible initial states. It is used to control the balance between digit deletions and insertions.

5.4.1 The Viterbi-Decoder

The joint likelihood of the HMM state sequence $\mathbf{q} = q_1, ..., q_T$ and the input stream $\mathbf{U} = U_1, ..., U_T$ is computed as

$$P(\mathbf{q}, \mathbf{U}) = P_o^{N^{\mathbf{q}}} \prod_{t=1}^{T} P(U_t|q_t), \tag{5.7}$$

with $N^{\mathbf{q}}$ being the number of digits implied by $\mathbf{q}$. In analogy with [65], one can show that if the training corpus were infinitely large and the targets during training were either 0 or 1, readout $y_{t,q}$ would be equal to $P(q|U_t)$ and Bayes' law could be applied to convert it to a likelihood. However, in case of a finite training set, the readouts only approximate the posteriors and they can be outside the interval [0,1]. To fix that problem, we introduce a mapping function, $f_{map}(\cdot)$ and compute the likelihood as follows:

$$P(U_t|q) = f_{map}(y_{t,q})\,\frac{P(U_t)}{P(q)}. \tag{5.8}$$

One way to define the mapping is to create two histograms for each state q: one representing the global distribution of $y_{t,q}$ and one representing the distribution over the times that state q is visited. From these histograms, one can then derive a lookup table for estimating $P(q|y_{t,q})$.

A solution with fewer free parameters is a sigmoid function with a steepness g_q and an offset b_q derived from the estimated $P(q|y_{t,q})$:

$$f_{map}(y_{t,q}) = \frac{1}{1 + e^{-g_q\,(y_{t,q}-b_q)}}. \tag{5.9}$$

Two other alternatives are a state-independent sigmoid and a simple clip-and-scale approach. The latter is given by

$$f_{map}(y_{t,q}) = \frac{\max(y_{t,q}, y_o)}{\max_j(y_{t,j})}, \quad y_o \ll 1. \tag{5.10}$$

In section 5.7, we experimentally test all four approaches as they represent different trade-offs between model detail and model generalization.

5.4.2 Iterative Training

Even though the optimal output weights of the RCN can be obtained in a closed-form, the embedded training of an RCN-HMM hybrid is an iterative process. And since the Aurora-2 corpus is delivered without timing information we even adopt a two-stage training procedure.

In the *first stage*, only isolated digit utterances are considered. They are first segmented into silence-digit-silence by means of an energy criterion and the digit intervals are uniformly divided into digit state segments. The matrix $\mathbf{D}$ derived thereof is employed to train an initial readout layer. Then, three iterations of embedded training are conducted. Per iteration, the outputs of the available RCN are used in a Viterbi-alignment of the training utterances with their silence-digit-silence models. From these alignments a new matrix $\mathbf{D}$ is computed and substituted in Equation (5.6).

In the *second stage*, all available training utterances are considered. They are modeled as sequences of digits interleaved with optional silences.

Starting with the alignments emerging from the system of the first stage, the training is continued until the digit error rate on a validation set saturates.

In terms of training effort, one can note that the reservoir is kept fixed throughout the training, meaning that the matrix $(\mathbf{R}\,\mathbf{R}^T + \epsilon\mathbf{I})^{-1}$ appearing in Equation (5.6) can be computed once, during the first iteration. The matrix $\mathbf{D}\,\mathbf{R}^T$ however changes from one iteration to the other because $\mathbf{D}$ changes. Even though $\mathbf{R}$ does not change across iterations, it is impractical to store it for a big reservoir and/or a large dataset. Therefore, we compute and store $(\mathbf{R}\,\mathbf{R}^T + \epsilon\mathbf{I})^{-1}$ in the first iteration, but recompute the reservoir outputs during each iteration.

5.5 An Efficient Reservoir Design Strategy

From Section 5.3 it follows that the reservoir can be described in terms of six control parameters: N^{res}, K^{in}, α_U, K^{rec}, ρ and λ. However, we argue that the reservoir size is mainly determined by the number of available training examples and that the other parameters should follow from properties of the reservoir inputs and the desired readouts, irrespective of the number of training examples.

In this section, we analyze the RCN as a dynamical system and we postulate some fairly general and comprehensible principles which lead us to the following conclusions: (1) K^{in} and K^{rec} are non-critical and are easy to set, and (2) it is possible to establish for any combination (ρ, λ) a closed-form expression that provides a near-optimal value for the input scaling factor α_U. In the experimental section, we will use the insight gained in the current section to translate empirical findings into a very simple recipe for tuning the reservoir dynamics to the observable input dynamics and the desired output dynamics.

5.5.1 Reservoir Neuron Equations

In order to develop our strategy, we write the input-output equations for reservoir neuron i in the following form:

$$r_{t,i} = (1-\lambda)\,r_{t-1,i} + \lambda\,f_{res}(a_{t,i}), \tag{5.11}$$

$$a_{t,i} = b_{t,i} + c_{t,i}, \tag{5.12}$$

$$b_{t,i} = \sum_{j\in\mathcal{J}_i} w^{in}_{ij}\,u_{t,j}, \tag{5.13}$$

$$c_{t,i} = \sum_{k\in\mathcal{K}_i} w^{rec}_{ik}\,r_{t-1,k}. \tag{5.14}$$

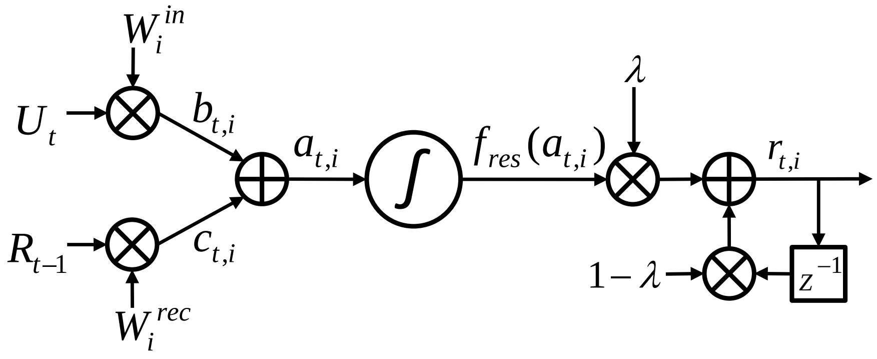


Figure 5.4: Visualization of Equations (5.11) to (5.14) describing Leaky Integrator Neuron i.

The sets $\mathcal{J}_i$ and $\mathcal{K}_i$ collect the inputs and reservoir outputs effectively driving neuron i. The symbols $u_{t,j}$ and $r_{t,i}$ refer to components of U_t and R_t, whereas $a_{t,i}$, $b_{t,i}$ and $c_{t,i}$ respectively refer to the total activation of the neuron and to its input and recurrent components. The equations are visualized in Figure 5.4.

5.5.2 Dynamics of the Desired Readouts

As a state is usually only visited once in a while, a particular readout is anticipated to consist of short pulses interleaved with long pauses. The length of such a pulse is anticipated to lie in some interval $(T_{\min}, T_{\max})$ that can be estimated on the basis of some rough knowledge of the recognition problem and the state definitions. If the length of the pulse is T its spectrum is concentrated in the frequency band $(0, F = 1/T)$. One might thus conclude that the bandwidth of the average readout spectrum will be equal to $F = 1/T_{\min}$.

We could also measure the mean power spectrum of the target readouts and determine the bandwidth from such an experiment. However, this requires the existence of segmented training data (which are lacking in Aurora-2) and it is not expected to yield a result that differs much from the one emerging from the above reasoning in combination with a rough estimate of the minimum state duration.

5.5.3 Basic Principles Underlying the Design Strategy

Our original idea was that in order to contribute effectively to the solution of the problem, a neuron should exhibit a sufficient amount of variation across time, but at the same time it should not saturate too frequently. This led us to the principle that the variance of the neuron stimulation should be large enough but not too large. We chose the variance as this is a robust measure of variation which can also be related to the spectrum of the temporal evolution in an easy way by means of Parseval's theorem.

Although we could explain a lot of results on the basis of this principle, the experiments with a variable number of states per digit did not seem to comply with this principle. It was only then that we realized that the high-frequency components which also contribute to the variance do not contribute to the solution of the problem the reservoir network is aiming to solve. Based on this new insight we finally postulated that it is the in-band variance of a neuron input that has to be constrained.

Based on F, we define the *in-band activation* pattern of a reservoir neuron as the pattern originating from the activation components with frequencies inside $(0, F)$, and we postulate that a reservoir neuron can maximally contribute to the solution of an envisioned recognition problem, if its in-band activation pattern has a preferred strength. In fact, if the activation strength is too small the non-linearity of the neurons will not be exploited and if it is too large, the neurons will saturate too frequently and become insensitive to input changes in these cases. We actually expect that there exists a preferred in-band activation strength that works well for all reservoirs and for an arbitrary bandwidth of the readouts. We further contemplate that variance is a good measure of activation strength.

5.5.4 Fixing K^{in} and K^{rec}

As described in Section 5.3, $\mathbf{W}^{in}$ has K^{in} non-zero entries per row and the values of these entries are drawn independently from a zero-mean Gaussian distribution. Consequently, if K^{in} is sufficiently large, the mean of the input activation component of a neuron will approximately be zero, even if the inputs have non-zero means. Exactly the same holds for the recurrent activation component. Introducing the short notation $E_{t,i}[\cdot\cdot]$ for the mean over time t and nodes i, the mean variance of the input component is given by

$$\begin{aligned} V_b \doteq E_{t,i}[b_{t,i}^2] &= \sum_{j,k=1}^{N_U} E_t[u_{t,j}\, u_{t,k}]\, E_i[W_{ij}^{in} W_{ik}^{in}] \\ &= \sum_{j=1}^{N_U} E_t[u_{t,j}^2]\, E_i[(W_{ij}^{in})^2] \\ &= K^{in}\, \alpha_U^2\, V_U \end{aligned}$$

where V_U is the mean variance of the reservoir inputs. This is so because $E_i[W_{ij}^{in}\, W_{ik}^{in}] = 0$ for $j \neq k$ and equal to K^{in}/N_U for $j = k$ (because the chance of W_{ij} being non-zero is K^{in}/N_U for any value of j) and because $E_t[u_{t,j}^2] = V_U$ for any value of j.

In a similar vein, one can also show that

$$V_c \doteq E_{t,i}[c_{t,i}^2] = \alpha_R^2\, K^{rec}\, V_R = \rho^2\, V_R \tag{5.15}$$

Furthermore, since all the rows of the matrix $\mathbf{W}^{rec}$ are random independent vectors with approximately the same norm, namely $K^{rec}\alpha_R^2$ which is going to be close to the largest eigenvalue of $\mathbf{W}^{rec}$. The latter was experimentally confirmed.

Since the strength of an in-band activation component is proportional to the strength of the the full activation, the above formulas imply that one can freely change K^{in} (or K^{rec}) provided the scale α_U (or α_R) is updated accordingly to maintain V_b (or V_c).

5.5.5 Finding the Input Scale Given the Reservoir Dynamics

Measurements show that there are only very weak correlations between $b_{t,i}$ and $c_{t,i}$, and similarly, between their in-band components $b_{t,i}^*$ and $c_{t,i}^*$. The main reasons for this are that first of all, the two activation components arise from vectors at times t and $t-1$ respectively, and secondly, both components are obtained by projecting the input and reservoir state vectors on independently chosen random hyperplanes in the input and reservoir state space respectively. However, this is no rigorous proof, hence, we conducted an experiment in which we measured V_a, V_b and V_c for different values of ρ and α_U respectively. The results depicted on Figure 5.5 confirm the hypothesis that $V_a \approx V_b + V_c$ and $V_a^* \approx V_b^* + V_c^*$.

Given an optimal value V_{opt} for V_a^* (as postulated) and given some combination (λ, ρ), we will now derive an analytical formula for finding the corresponding optimal α_U.

First of all, we relate V_b^* to V_b. To do so, we need the mean power spectrum $|B(f)|^2$ of the input activation. Since $b_{t,i}$ is obtained as a linear sum of the inputs $u_{t,j}$ and since the weight coefficients are independent, b and u have a zero mean and an equal variance, hence, it can be shown that $E_i[|B_i(f)|^2]$ is proportional to the mean power spectrum of the inputs.

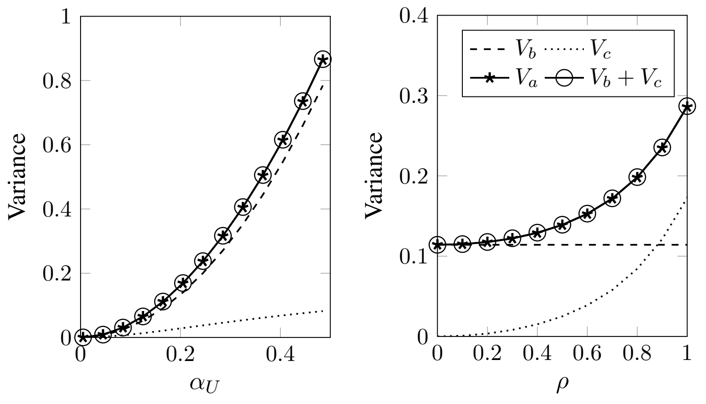


Figure 5.5: V_a, V_b and V_c for different values of and α_U (left) and ρ (right).

Indeed,

$$\begin{aligned}
B_i(f) &= \sum_i W_{ij}^{in} U_j(f) \\
|B_i(f)|^2 &= \sum_{j,k=1}^{N_U} W_{ij}^{in}\, W_{ik}^{in}\, U_j^*(f)\, U_k(f) \\
E_i[|B(f)|^2] &= \sum_{j=1}^{N_U} E_i[W_{ij}^{in}\, W_{ik}^{in}]\, U_j^*(f)\, U_k(f) \\
&= \frac{K^{in}}{N_U} \sum_{j=1}^{N_U} |U_j(f)|^2
\end{aligned}$$

This result means that the power spectrum estimation procedure proposed in the original article can actually be simplified to an estimation of the mean power spectrum of the individual inputs. It also implies that the shape of $|B(f)|^2$ does not depend on K^{in}. This was practically confirmed by comparing the shape of $|B(f)|^2$ for four different values of K^{in} (See Figure 5.6).

By applying Parseval's theorem, we have

$$V_b^* = \phi_b(F)\, V_b, \tag{5.16}$$

$$\phi_b(F) = \frac{\int_{-F}^{F} |B(f)|^2\, df}{\int_{-0.5}^{0.5} |B(f)|^2\, df}. \tag{5.17}$$

Now, we try to relate V_c^* to V_c. This is much more complicated due to the non-linearities involved. Therefore, in order to derive an approximate

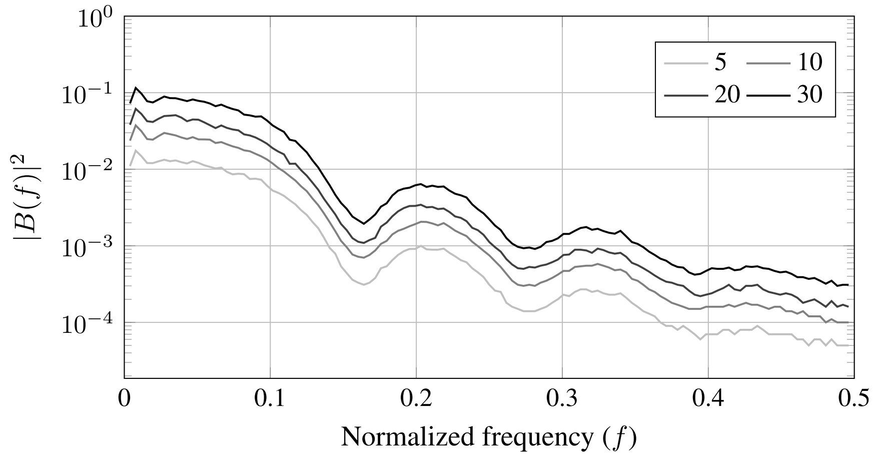


Figure 5.6: The shape of $|B(f)|^2$ for four different values of K^{in}

relation, we make two assumptions: (1) the activation function $f_{res}(x)$ is linear and equal to x, and (2) the recurrent activation component is equal to $\rho\, r_{t-1,i}$. Strictly speaking, the linearity assumption is in contradiction with the requirement that the in-band activation should be large enough to let the non-linearity play some role. However, since we also stated that this role has to remain modest, we expect that results emerging from the linearity assumption will remain valid for a soft and compressing activation function such as tanh. The assumption that $c_{t,i} = \rho\, r_{t-1,i}$ should be seen in the context of Equation (5.15) which says that the expected variance of $c_{t,i}$ is equal to ρ^2 times V_R. The proposed $c_{t,i}$, at least, has this variance and it makes the analysis tractable. The effect of just feeding the neuron output back into the neuron rather than a weighted sum of other neuron outputs is of course difficult to assess, but the experimental validation will confirm that our assumption leads to acceptable results.

Applying Parseval's theorem again, we find that V_c^* is proportional to V_c with a factor that is a function of the power spectrum $|C_{\lambda,\rho}(f)|^2$ of $c_{t,i}$:

$$V_c^* = \phi_c(F,\lambda,\rho)\, V_c, \tag{5.18}$$

$$\phi_c(F,\lambda,\rho) = \frac{\int_{-F}^{F} |C_{\lambda,\rho}(f)|^2\, df}{\int_{-0.5}^{0.5} |C_{\lambda,\rho}(f)|^2\, df}. \tag{5.19}$$

Obviously, as $|C_{\lambda,\rho}(f)|^2$ depends on λ and ρ, it cannot be measured using an arbitrary reservoir anymore. However, it can be derived from the closed-loop transfer function of the neuron (as a linear system). This function is a first-order low-pass filter

$$H_{\lambda,\rho}(z) = \frac{\lambda\rho\, z^{-1}}{1 - (1 - \lambda + \lambda\rho)\, z^{-1}}$$

Consequently, we obtain that

$$\phi_c(F, \lambda, \rho) = \frac{\int_{-F}^{F} |H_{\lambda,\rho}(f)|^2\, |B(f)|^2\, df}{\int_{-0.5}^{0.5} |H_{\lambda,\rho}(f)|^2\, |B(f)|^2\, df} \tag{5.20}$$

The above results means that

$$V_c = \int_{-0.5}^{+0.5} |H_{\lambda,\rho}(f)|^2\, |B(f)|^2\, df$$

and consequently, that

$$V_c = \phi(\lambda, \rho)\, V_b, \qquad \phi(\lambda, \rho) = \frac{\int_{-0.5}^{+0.5} |H_{\lambda,\rho}(f)|^2\, |B(f)|^2\, df}{\int_{-0.5}^{+0.5} |B(f)|^2\, df}$$

and

$$\phi_c(F, \lambda, \rho) = \frac{\int_{-F}^{+F} |H_{\lambda,\rho}(f)|^2\, |B(f)|^2\, df}{\int_{-0.5}^{+0.5} |H_{\lambda,\rho}(f)|^2\, |B(f)|^2\, df}$$

The final equation then becomes

$$V_a^* = [\phi_b(F) + \phi(\lambda, \rho)\, \phi_c(F, \lambda, \rho)]\, V_b \tag{5.21}$$

In the following section, we will discuss how to exploit this relation during the design of a good reservoir.

5.5.6 Proposed Design Strategy

We propose to fix K^{in} and K^{rec} (almost free choice, as will be demonstrated) and to estimate the expected state duration T_{state} (based on knowledge about the task), first. Then, conduct two small-scale experiments to determine $|B(f)|^2$ and V_{opt}, the optimal value for V_a^*:

1. Estimate the power spectrum $|B(f)|^2$ of the input activations by means of a memoryless reservoir (meaning that $(\lambda, \rho) = (1, 0)$) that is just big enough to ensure that the recurrent weight matrix is sparse for the chosen K^{rec}.

2. Test networks with reservoirs with the same size and reservoir dynamics but with different values of α_U. Establish the α_U offering the highest classification accuracy and use Equation (5.21) to derive V_{opt}. The size of the reservoir is not critical (just take it large enough to get at least a reasonable performance).

After these experiments one can start to design the envisaged (large) reservoirs by trying different combinations (λ, ρ) and by deriving the optimal α_U going with this combination from

$$\alpha_U^2 \, K^{in} \, V_U = \frac{V_{opt}}{\phi_b(F) + \phi(\lambda, \rho) \, \phi_c(F, \lambda, \rho)}. \tag{5.22}$$

(derived from Equations (5.15) and (5.21)). Obviously, the design time is then going to be proportional to the number of combinations (λ, ρ) to consider. In the experimental section, it will be demonstrated that we can unequivocally tune the reservoir dynamics to the recognition problem, which eliminates the necessity of considering more than one combination (λ, ρ).

5.6 Validation of the Design Strategy

The aim of this section is to seek experimental validation of the proposed strategy and to establish the V_{opt} to impute in Equation (5.22).

5.6.1 Experimental Framework

In this stage, we conduct different digit state recognition experiments within the Aurora-2 context. The aim is to define tasks implying different output dynamics and different numbers of readouts. Therefore, we consider left-to-right digit models of different lengths (different number of states S per model), we label the states and perform digit state recognition by replacing the digit loop in Figure 5.3 by a digit state loop that can generate an arbitrary state sequence.

By changing S, one changes the output dynamics (F) and the number of readouts ($11 \times S$ digit states plus one silence state). From a histogram of the digit durations, we derive that most digits are longer than 250 ms. We, therefore, choose $F = S/250$ as the readout bandwidth (in kHz). Note that if S is low, each state may encompass acoustically diverse speech frames which are hard to cluster in the acoustic feature space, whereas if S is high, the states represent acoustically more similar speech frames. Due to this, a larger S does not necessarily lead to a more difficult task.

In each experiment, an RCN-HMM hybrid is trained on two-thirds of the clean training utterances and validated on the remaining third of these utterances. The State Error Rate (SER) on the validation set is used as the evaluation measure. It is obtained by counting the minimum number of state deletions, insertions and substitutions that transform the recognized state sequence into one that is compatible with the spoken digit sequence, given the left-to-right and no-skip models of these digits. As before, a penalty P_o is employed to control the deletion/insertion balance.

In all experiments, the 39 acoustic features per frame are generated by a standard MFCC analysis (30 ms Hamming-windowed frames, frame shift of 10 ms) yielding 12 mel-cepstral coefficients, a log-energy and the first and second order derivatives thereof. The feature extraction is followed by an utterance-wise normalization that creates zero-mean and unit-variance inputs ($V_U = 1$). The ultimate mean and variance normalized features are denoted by the acronym MVN.

Unless stated otherwise, we employ a lookup-table (see Section 5.4) to map the readouts to the interval [0,1] and we choose $S = 3$ as a default.

5.6.2 Variance as a Measure of Activation Strength

If variance were a good measure of activation strength then the theory would predict that the SER is independent of K^{in} and K^{rec} as long as the variances V_b and V_c are maintained.

To verify this hypothesis for K^{in}, we test a reservoir with $K^{rec} = 0$ for three values of V_b $(0.01, 0.02$ and $0.05)$ and another one with $(\rho, K^{rec}, V_b) = (0.8, 1, 0.02)$. To verify the hypothesis for K^{rec}, we test a reservoir with $(K^{in}, V_b) = (10, 0.02)$, and three values of ρ (0.1, 0.5 and 0.8). Note that we verified experimentally that as long as ρ is not too close to 1, keeping ρ constant is equivalent to keeping V_c constant. In all experiments, we set $\lambda = 1$.

Figure 5.7 largely supports the envisioned predictions. There is only one strong exception, namely the case that the neurons have no recurrent connections and are driven by only 1 or 2 inputs. In that case the rows of the input matrix can no longer be expected to have the same norm, as assumed in the theory. From now on, we always work with $K^{in} = K^{rec} = 10$.

5.6.3 Existence of a Universal V_{opt}

Let us now establish whether there exists a single value for V_a^* that is quasi-optimal for different combinations of (λ, ρ, F). We, first, keep F fixed and examine different combinations (λ, ρ). For each of them, we determine

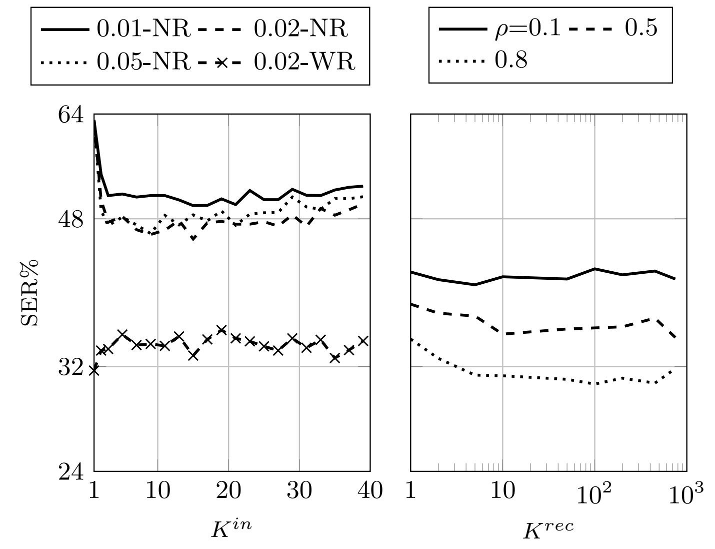


Figure 5.7: (Left) State Error Rate (SER) as a function of K^{in} for three values of V_b and $V_c = 0$. Three reservoirs have no recurrent connections (NR), the fourth one (WR) has 1 recurrent connection per neuron. (Right) SER as a function of K^{rec} for three values of ρ and for α_U = 0.4, K^{in} = 10, $\lambda = 1$. All reservoirs comprise 750 neurons.

$\lambda\backslash\rho$	0.1	0.2	0.4	0.6	0.8
1.00	0.018	0.020	0.017	0.022	0.019
0.33	0.021	0.024	0.028	0.030	0.031
0.18	0.032	0.036	0.040	0.038	0.041

Table 5.3: In-band activation strength V_a^* at the optimal V_b for 15 combinations of ρ and λ.

the optimal V_b and compute the corresponding optimal V_a^* from Equation (5.21). In Table 5.3, one finds these values for a reservoir with 750 neurons. The table suggests that the optimal V_a^* depends on λ and to some extent also on ρ. However, Figure 5.8 shows that for small values of λ, the interval of quasi-optimal V_b's becomes wider and that $V_a^* = 0.035$ (the circles in Figure 5.8) always leads to a quasi-optimal SER. This means that Equation (5.21) is not significantly violated and that it offers an acceptable means of expressing the relation between V_a^* and V_b at one particular F.

Let us now check if Equation (5.21) remains suitable when F is changed.

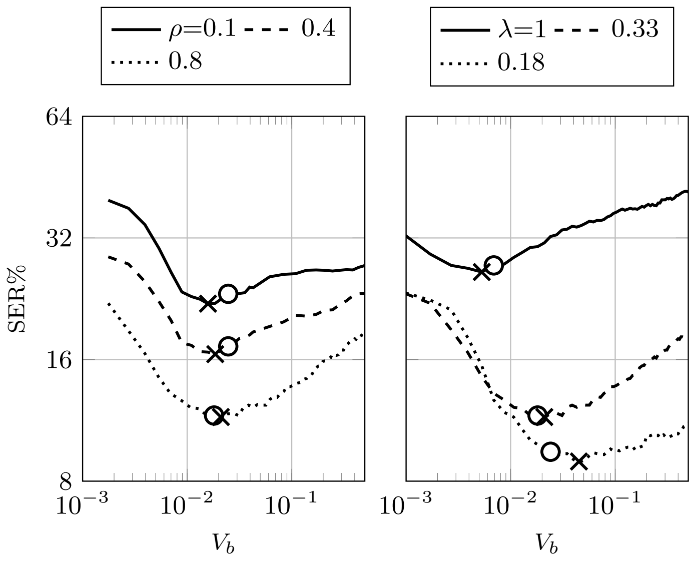


Figure 5.8: State Error Rate (SER) as a function of V_b for three values of ρ at $\lambda = 0.33$ (left) and for three values of λ at $\rho = 0.8$ (right). The reservoirs have 750 neurons. Also indicated are the optimal V_b's (x) and the V_b's computed using Equation (5.22) for V_{opt} = 0.035 (o).

$S\backslash V_a^*$	0.01	0.025	0.035	0.05
1	21.9	20.4	21.3	22.1
3	12.9	12.1	12.1	12.6
5	12.6	12.1	12.0	11.7

Table 5.4: SER (in %) as a function of V_a^* for three systems with respectively 1, 3, and 5 states per digit.

Therefore, we consider RCN-HMM hybrids for state recognition tasks corresponding to S = 1, 3 and 5. We employ reservoirs with $(N^{res}, \lambda, \rho) = (750, 0.33, 0.8)$ and different values of V_b, namely those values for which Equation (5.21) yields values of V_a^* that are equal to 0.01, 0.025, 0.035 and 0.05. According to the results listed in Table 5.4, the value of 0.035 for V_a^*, that was established for $S = 3$ remains a good value here too, even though it yields a slightly suboptimal result for the not so interesting case $S = 1$.

The conclusion of our experiments is that Equation (5.22) leads to a good approximation of the optimal V_b in all cases, provided $V_{opt} = 0.035$. This value corresponds to a standard deviation of 0.19. Presuming a Gaussian distribution for the in-band activation, this means that 5% of the in-band activation samples are larger than 0.375, ensuring that the non-linearity is effectively starting to play. This is a clear confirmation of the basis principle we postulated.

5.7 A Single-Layer RCN-HMM Hybrid

In this section we develop RCN-HMM hybrids for CDR. They comprise one basic RCN, designed with $K^{in} = K^{rec} = 10$ and $V_{opt} = 0.035$. The size of the reservoir, the number of states per digit and (ρ, λ) are the dependent parameters.

5.7.1 System Development

During system development, hybrids are trained on two-thirds of the clean training utterances of Aurora-2 and validated on the remaining clean training utterances. The evaluation measure is the WER. We perform experiments to set the reservoir dynamics, to decide about the readout-to-posterior mapping and to investigate the effect of N^{res} and S on the WER.

5.7.1.1 Reservoir Dynamics

In order to derive practical heuristics for finding good dynamic parameters, we first consider a reservoir of 750 nodes with $S = 3$ and we assess the validation WER as a function of τ_λ and τ_ρ (see Equation (5.4)).

Figure 5.9 supports the notion of τ_ρ and τ_λ being two independent parameters to tune the reservoir to the task it is intended for: the optimal value of one of them is the same in a broad range of values for the other. Therefore, we propose to proceed as follows: (1) track the WER as a function of τ_λ for reservoirs with τ_ρ = 50 ms (corresponding to ρ = 0.82) and determine the optimal τ_λ, (2) track the WER as a function of τ_ρ for reservoirs with τ_λ = 100 ms and determine the optimal τ_ρ, (3) try to link the optimal values to the dynamical properties of the task in the hope to find a good and simple heuristic. Note that steps (1) and (2) can be run in parallel.

Figure 5.10 shows the two tracks for three values of S. The left track reveals that leaky integration can significantly reduce the WER. Its positive effect is maximal if its time constant reaches a certain point that is clearly a function of S, and conversely, of the output dynamics. The data support the expectation that leaky integration can suppress fluctuations in the reservoir outputs that originate from random fluctuations in the acoustic features. In particular, they are consistent with the heuristic to fix τ_λ to the average digit

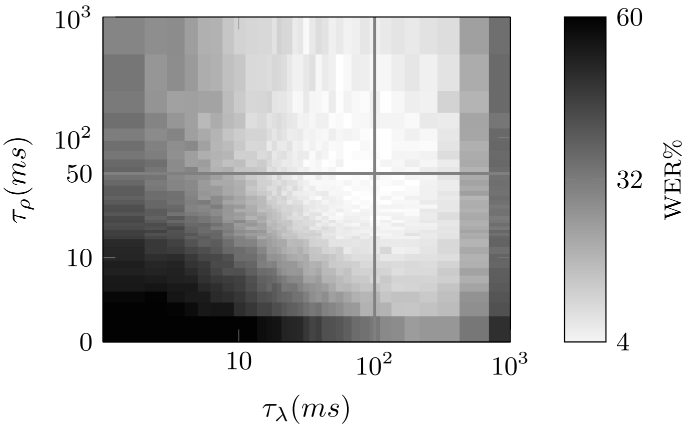


Figure 5.9: WERs (in %) on the validation set as a function of $(\tau_\lambda, \tau_\rho)$ for a system comprising a reservoir with 750 nodes and 3 states per digit.

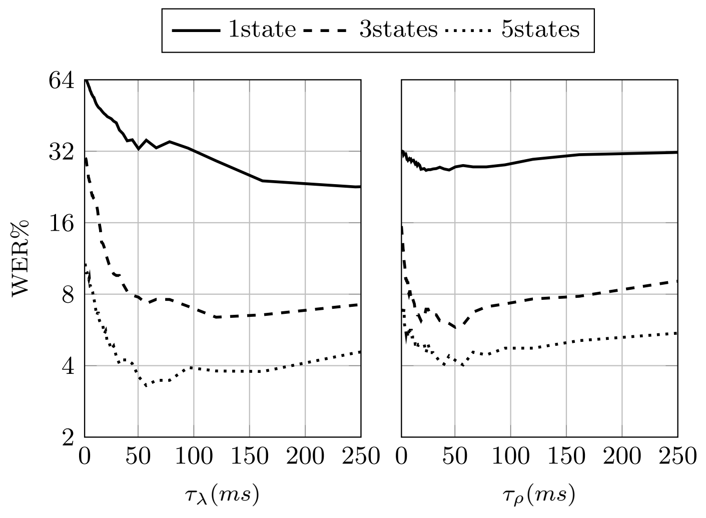


Figure 5.10: WERs (in %) on the validation set as a function of τ_λ with $\tau_\rho = 50$ ms (left) and as a function of τ_ρ with $\tau_\lambda = 100$ ms (right). Depicted are results for different values of S. The reservoir size is 750.

state duration T_{state}. The right track reveals that recurrence is a powerful tool for improving the performance, as well. The optimal point seems to be much less dependent on S. This agrees with our expectation that the memory of the reservoir should be adapted to the dynamical properties of

	lookup table	local sigmoid	global sigmoid	clip at Y_o
WER	3.63	3.87	3.94	4.01

Table 5.5: WERs (in %) on the validation set obtained with four readouts-to-posterior mapping methods. The reservoir size is 750.

the reservoir inputs, and consequently to the bandwidth F_B of the power spectrum $|B(f)|^2$. Since that spectrum is a low-pass spectrum, F_B can be defined as the frequency where $|B(f)|^2$ drops below its half maximal value. The optimal $\tau_\rho = 50$ ms then follows from the heuristic that τ_ρ (ms) = $0.35/F_B$ (kHz) where F_B (kHz) is obtained by converting the normalized frequency F_B (running from 0 to half the frame rate of 100 Hz) to kHz. Note that the just formulated heuristic is exactly the relation between the bandwidth of a first order linear system and the time constant of its impulse response. From now on, we use this relation to fix τ_ρ.

5.7.1.2 Readout-To-Posterior Mapping

Table 5.5 lists the WERs obtained with a reservoir of 750 nodes for four readout-to-posterior mapping methods. The differences between methods are modest but nevertheless, the difference between the state-specific lookup-table method and the two global methods is almost statistically significant (Wilcoxon test, $p < 0.10$ [147]). We, therefore continue to use the lookup-table method.

5.7.1.3 Reservoir Size and Number of States per Digit

Figure 5.11 shows the WER on the validation set as a function of the number of states per digit for different sizes of the reservoir. The WER decreases with the reservoir size, even though a reservoir of 16K neurons already encompasses 1.7M trainable parameters, whereas, there are only 0.8M training frames. This indicates that the risk of over-training the readouts is low, a conclusion that confirms our earlier findings [9]. Our data suggest that even bigger reservoirs could offer further improvements. However, we did not try that here because, as we will explain later on, the training of systems with very big reservoirs becomes impractical on standard computers and the anticipated improvements can be achieved more effectively by systems encompassing multiple reservoirs of manageable sizes.

The WER decreases with the number of states until it saturates at $S = 7$. Consequently, there is no need to use more than 7 states per digit.

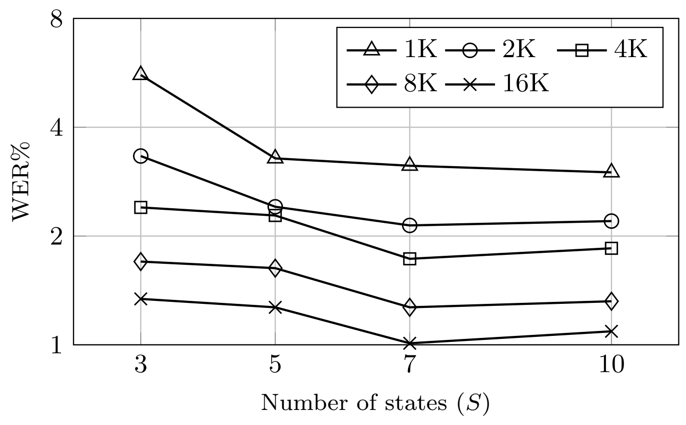


Figure 5.11: WERs (in %) on the validation set as a function of S for six values of the reservoir size

5.7.2 System Evaluation

During system evaluation, the output weights of the reservoir network are trained on the full training set using the same reservoir parameters and number of iterations that were employed in the development stage. Furthermore, both clean and multi-style training are considered now. All systems are evaluated on all the Aurora-2 test sets (A-C).

In Table 5.6, the WERs of eight RCN-HMM hybrids are listed next to the WERs of two baseline GMM-HMM systems: one that employs the same acoustic features (MVN) and one that works on the advanced front-end features (AFE).

The RCN-HMM systems encompassing a reservoir of 16K or more neurons clearly outperform the baseline GMM-HMM systems in mismatched conditions, and are competitive with a GMM-HMM system working in matched conditions on the same acoustic features. However, the GMM-HMM working with AFE's still stands out in the clean test condition.

The RCN-HMM data in Table 5.6 confirm that the improvements due to more reservoir neurons and digit states are in line with those observed during system development. Although further enlarging the reservoir is bound to lead to further improvement, we did not pursue this path because the required computational resources increase steeply then (see Section 5.7.4) and also because multi-layer RCN-HMM systems will turn out more effective in this respect (see Section 5.8).

Method	Clean			Multi		
	Clean	0-20dB	-5dB	Clean	0-20dB	-5dB
GMM-HMM (MVN)	0.84	19.7	82.2	1.77	8.5	59.1
GMM-HMM (AFE)	**0.77**	13.0	69.7	**0.81**	8.2	58.3
RCN-HMM (2K, 7st)	2.17	13.9	62.9	3.43	10.9	57.7
RCN-HMM (4K, 7st)	1.56	12.8	62.6	2.34	9.4	55.6
RCN-HMM (8K, 7st)	1.31	12.0	61.9	1.92	8.3	53.3
RCN-HMM (16K, 7st)	1.08	11.3	**61.8**	1.47	7.4	50.4
RCN-HMM (24K, 7st)	0.88	**10.9**	62.0	1.31	**6.9**	**50.1**
RCN-HMM (8K, 3st)	1.60	15.9	73.0	2.21	11.0	67.1
RCN-HMM (8K, 5st)	1.49	12.8	64.0	2.10	8.3	52.9
RCN-HMM (8K, 7st)	1.31	12.0	61.9	1.92	8.3	53.3
RCN-HMM (8K, 10st)	1.12	12.7	63.9	1.82	8.8	56.0

Table 5.6: WERs (in %) per condition (SNR-range) for test sets A - C. Both clean training (left) and multi-style training (right) experiments are considered. The values in bold mark the best systems for the different conditions. The RCN-HMM systems are supplied with MVN features and are labeled with their reservoir size and number of states per digit. The GMM-HMM systems use 16 states per digit and 3 gaussians per state.

5.7.3 Control Experiments

So far, we have established that RCN-based systems offer more noise-robustness than GMM-based systems, but we did not yet establish why that is. Is it due to the ability of the reservoir to filter-out noise inflicted modulations whose frequencies fall outside the speech band (between 0 and F), or it is due to the random fixation of the reservoir weights? To answer these questions, we have performed control experiments with systems encompassing an 8K reservoir and 7 state digit models.

In Figure 5.12, we depicted the WER as a function of the SNR of the test sets (A-C) for the GMM-HMM (MVN) system and for four RCN-HMM systems which are labeled by their τ_λ and τ_ρ (in ms). System RC-00-00 is a system incorporating a reservoir without recurrent connections and with memoryless neurons and system RC-35-50 is the one with the optimal dynamics. The figure clearly shows that the noise degradation curve of the GMM-HMM system is steeper than that of the RCN-HMM hybrids and that all RCN-HMM curves have a similar steepness. This means that the noise robustness (slope of the curve) follows from the random fixation of the reservoir weights. Good reservoir dynamics on the other hand signif-

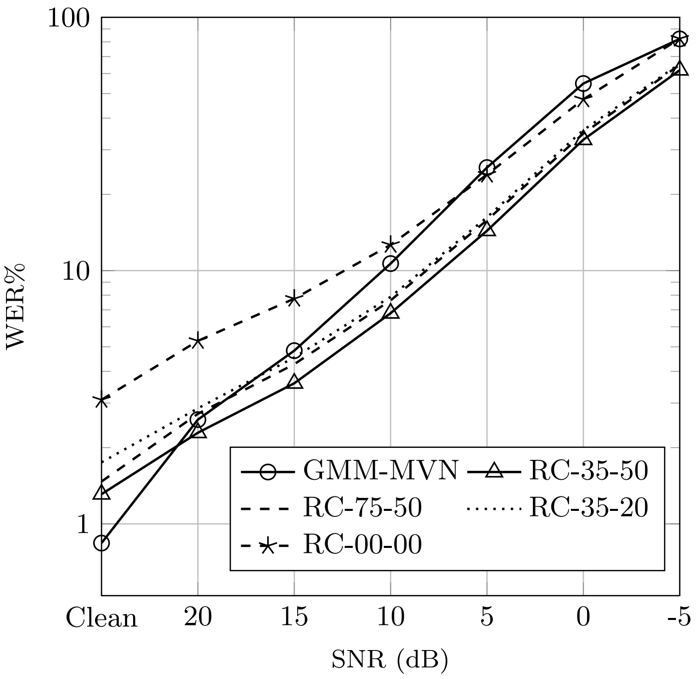


Figure 5.12: Trained on clean samples and tested on test set A-C of Aurora-2 framework, the noise robustness of RCN-HMM hybrids and a GMM-HMM system are compared. All systems are supplied with mean and variance normalized acoustic features (MVN). The RCN-HMM hybrids are encoded by their time constants τ_λ and τ_ρ in ms, respectively. The system RC-00-00 denotes a system without leaky integration and without recurrent connections.

icantly improve the accuracy. The same conclusions also hold for multi-style experiments.

5.7.4 Computational Resources

In order to give the reader a concrete idea about the time needed to train a reservoir with N^{res} nodes and N^{out} readouts on N^{frm} training frames, we decompose the training into six actions and assess the expected number of operations for each action (see Table 5.7). As a reference, we also give actual times (in minutes) consumed by each action on a 3.4 GHz single core CPU for the case of a reservoir with 16K neurons, 78 readouts ($S = 7$) and training on the complete Aurora-2 training set ($\approx$ 1.5M frames). From the

Action	Complexity	Time
Compute $\mathbf{R}$	$\mathcal{O}(N^{frm}\, N^{res})$	36
Update $\mathbf{RR}^T(*)$	$\mathcal{O}(N^{frm}\, (N^{res})^2)$	651
Invert $(\mathbf{RR}^T + \epsilon\, \mathbf{I})$ $(*)$	$\mathcal{O}((N^{res})^3)$	6
Align to find $\mathbf{D}$	$\mathcal{O}(N^{frm}\, N^{out})$	58
Compute $\mathbf{DR}^T$	$\mathcal{O}(N^{frm}\, N^{res}\, N^{out})$	5
Find $\mathbf{W^{out}}$ from Eq.(5.6)	$\mathcal{O}((N^{res})^2\, N^{out})$	<1

Table 5.7: Time complexity (function of parameters) and actual times (in minutes, measured on a 3.4 GHz core) consumed by the different steps of the full training of a reservoir with 16K neurons and 78 readouts. Items tagged with $(*)$ are only needed in the first training iteration.

figures one can derive that the computation of $\mathbf{RR}^T$ in the first iteration is by far the most time consuming part. Fortunately, it is also the easiest part to run in parallel on multiple cores. Pruning in the Viterbi alignment can reduce the time consumed in the further iterations.

5.8 A Multi-Layer RCN-HMM Hybrid

Research on deep neural networks has shown that the performance of a neural network can be increased by stacking multiple hidden layers on top of each other. However, since in the case of an RCN the hidden layer (the reservoir) is not trained, it does not make much sense to stack reservoirs. Instead, we stack complete reservoir networks (reservoir + readouts) as shown in Figure 5.13. Each reservoir network, from now on called a layer, is driven by the readouts of the preceding layer or by the acoustic inputs (layer 1) and the layers are trained one by one. There is already empirical evidence from phone recognition [9] that cascading reservoir networks can improve the recognition.

5.8.1 System Development

Since we see no advantage in doing it otherwise, we consider the digit states as the training targets in each layer. Consequently, each layer has the same number of readouts. However, as will be illustrated in Section 5.8.1.3, the readouts driving the higher layers are smoother than the acoustic features driving the first reservoir. This means that the higher layer reservoirs actually need other control parameters than the first layer reservoir. Another difference is that the task of the higher layers is to adjust the hypotheses of

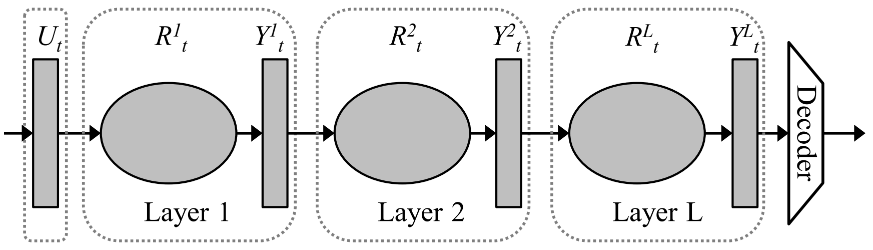


Figure 5.13: Architecture of a multi-layer reservoir network that can be embedded in the RCN-HMM hybrid

the former layer, whereas the task of the first layer is to bridge the big gap between the acoustic features and the state hypotheses.

Another point is that the readouts of the first layer no longer have zero means and unit variances. However, this is not so problematic, because the zero mean input weights will still cause the input activations to be zero mean. Furthermore, since each digit state has the same prior probability, one can assume the corresponding readouts to have an equal variance that can be substituted as V_U in Equation (5.15).

5.8.1.1 Preferred In-Band Activation Strength

Since the inputs of the higher layers are much closer to the target readouts than those of the first layer, we experimentally verified whether the formerly found preferred in-band activation strength of 0.035 still is an acceptable value. If so, this can be considered as an indication that the preferred activation strength may be problem independent, as hypothesized.

5.8.1.2 Reservoir Dynamics

To verify whether our heuristics for finding the reservoir dynamics also hold for the higher layers, we repeated the experiment that gave rise to Figure 5.10, but now we just did it for 7 states per digit. The results in Figure 5.14 support our claim about the purpose of leaky integration. As the readouts exhibit less random fluctuations than the acoustic features (as illustrated in Section 5.8.1.3), the choice of τ_λ is not that critical anymore. Any value not exceeding the average digit state duration is acceptable now. The heuristic for τ_ρ now yields a value of 130 ms which also seems to be a suitable value. Actually, this need for more memory agrees with the utilization of longer windows in the higher layers of other hierarchical systems, such as the one proposed in [74].

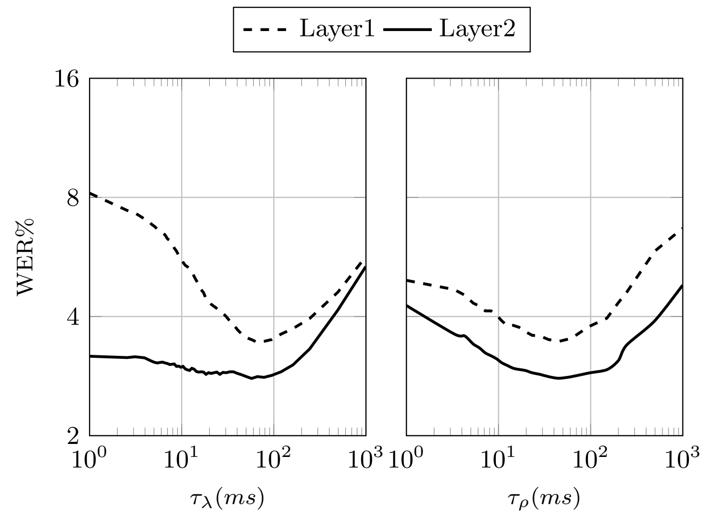


Figure 5.14: WERs (in %) on the validation set for a single-layer and a two-layer system as a function of the τ_λ (with τ_ρ = 50 ms) (left) and τ_ρ (with τ_λ = 100 ms) (right) of the last layer. The reservoirs consist of 750 neurons.

5.8.1.3 Deep Architectures

Although this is not necessary, we stick to multi-layer systems with equally large layers for the time being. This is partly for convenience but also because equal layers yield the minimal training time for a given number of trainable parameters: the number of trainable parameters is proportional to the sum of the reservoir sizes whereas the training time is proportional to the sum of squares of the reservoir sizes. Note that in the recognition phase, the computational complexity is determined by the total number of trainable parameters, irrespective of the number of layers involved.

To create our deep architectures, we use the same design choices for layers 2 and higher. This means: V_{opt} = 0.035, $\tau_\lambda = T_{state}$ and $\tau_\rho = 130$ ms. The results depicted in Figure 5.15 support the following conclusions:

1. Any single layer system can be improved by adding extra layers, but the attainable improvement decreases when the reservoir size increases. Nevertheless, even for a reservoir size of 16K, the attainable improvement is still as large as 30% relative.

2. For a given number of trainable parameters, multi-layer systems outperform single-layer systems, but again, the improvement decreases

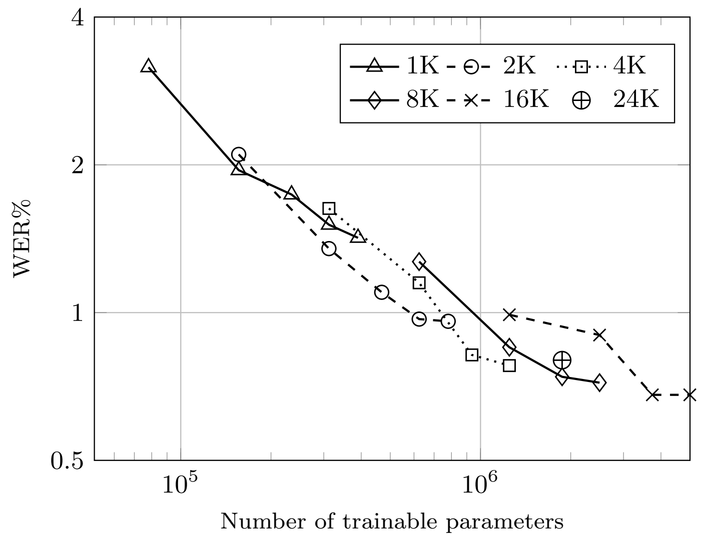


Figure 5.15: WERs (in %) on the validation set as a function of the number of trainable parameters (under the constraint that the reservoirs in each layer have the same size). Each curve represents the results obtained with systems of different depths using reservoirs of one particular size. The marks encode the reservoir size.

when the reservoir size increases. In fact, the gain of a 4×4K system over a 1×16K system is about 20% relative, whereas the gain of a 3×8K system over a single-layer 24K system has already dropped below 8%.

3. The larger the reservoirs become, the fewer layers are needed to reach the asymptotic performance of multi-layer systems using that reservoir size.

Normally, the performance curve as a function of the number of layers is convex, but for the case of 16K reservoirs this is no longer the case. The latter probably indicates that some over-training has finally occurred. This is not so surprising since there are already 1.24 million trainable parameters per layer in that case. To provide a more qualitative impression of what the different layers achieve, we considered a clean example of digit *zero* and we plotted some of the normalized MFCCs as well as the readouts of the subsequent layers that correspond to the subsequent digit states (Figure 5.16). It is clear that the readouts of all layers are smooth functions of time and that the competition between the desired readout and the other readouts diminishes when traversing the layers.

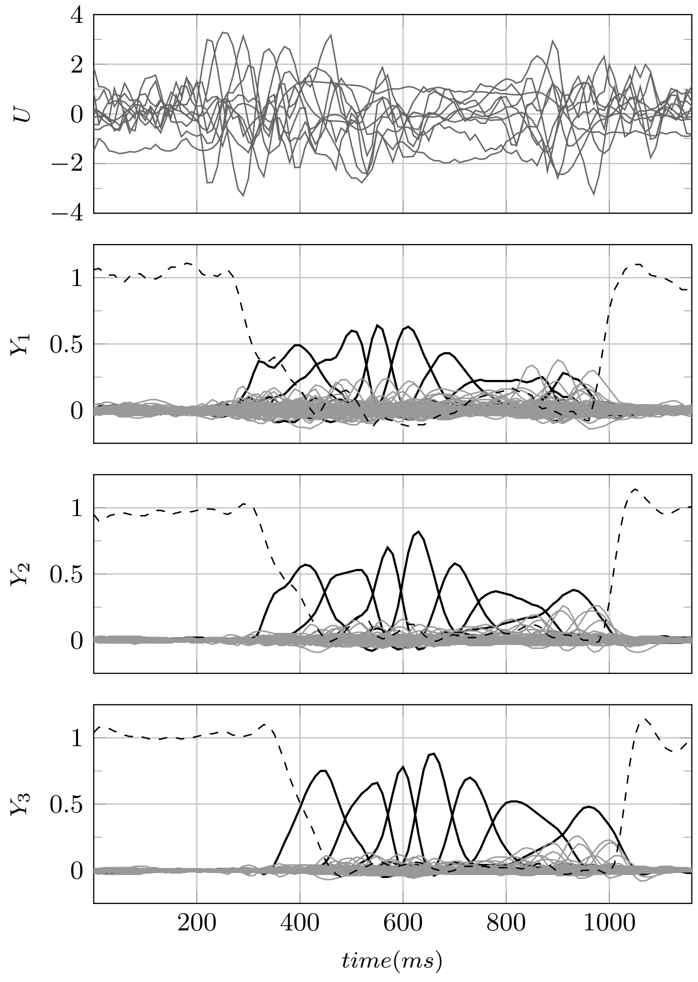


Figure 5.16: The acoustic inputs of a clean sample of digit *zero* and the readouts of the different layers of a 3×4K system with 7 states per digit. The *zero* and silence states are shown in black and dashed, respectively.

5.8.2 System Evaluation

Following the development experiments, we tested multi-layer systems with 8K and 16K reservoirs. As before, we retrained the systems on the full training set, but of course using the control parameters established during the development phase.

The results for 8K (see Table 5.8) demonstrate that in matched conditions adding layers leads to significantly better results at the price of a neg-

Layer size				Clean			Multi		
L1	L2	L3	L4	Clean	0-20dB	-5dB	Clean	0-20dB	-5dB
GMM-HMM (MVN)				0.84	19.7	82.2	1.77	8.5	59.1
GMM-HMM (AFE)				0.77	13.0	69.7	0.81	8.2	58.3
8K	-	-	-	1.31	12.0	61.9	1.92	8.3	53.3
8K	8K	-	-	1.03	11.4	62.0	1.49	7.1	51.0
8K	8K	8K	-	0.77	11.5	62.8	1.25	6.9	51.3
8K	8K	8K	8K	0.75	12.0	64.2	1.12	6.9	52.4
16K	-	-	-	1.08	11.3	61.8	1.47	7.4	50.4
16K	16K	-	-	1.06	11.1	62.1	1.52	7.1	52.7
16K	16K	16K	-	0.90	11.3	63.3	1.30	6.4	50.6
24K	-	-	-	0.88	10.9	62.0	1.31	6.9	50.1
2K	4K	16K	-	1.10	12.7	63.1	1.83	8.5	55.3
16K	4K	2K	-	0.86	11.0	62.7	1.22	6.5	50.0

Table 5.8: WERs (in %) per condition (SNR range) for test sets A - C obtained with multi-layer RCN-HMM hybrids using MVN features. Both clean training (left) and multi-style training (right) experiments are considered. The GMM-HMM and the single-layer results are copied from Table 5.6 to provide a compact presentation of all results.

ligible loss in performance in mismatched conditions. In the clean training experiment, only the clean condition can be considered matched, be it that Test C already introduces some mismatch (MIRS instead of G712 filter) in that condition as well. In the multi-condition training experiment, both the clean and the 0-20 dB conditions can be considered matched as most of the SNRs involved were also present during the training. The findings for the matched conditions are in very good agreement with the results emerging from the development phase.

The results for 16K (see Table 5.8) demonstrate more or less the same tendencies, but the differences are smaller. Note that just like in the development phase, the improvement (in matched conditions) caused by the second layer is much smaller than that caused by the third layer.

Out of curiosity we also evaluated some additional systems composed of layers of different sizes. Two typical examples are listed in the last two rows of Table 5.8. They show that in case of a big first layer, the correction work of the higher layers can be accomplished by smaller layers (16K + 4K + 2K works as well as 3 x 16K) which does not come as a surprise. When the first layer is small, the corrections to make in the higher layers are ob-

	Set	Clean	20	15	10	5	0	-5	0-20dB
Clean	A	0.78	1.30	2.36	5.0	11.5	29.2	59.4	9.9
	B	0.78	1.46	2.48	4.5	11.1	28.2	58.3	9.6
	C	0.91	1.54	2.91	5.7	12.5	30.0	57.6	10.5
	Average	0.81	1.41	2.52	4.9	11.5	29.0	58.6	9.9
Multi	A	1.25	1.12	1.51	3.0	6.1	17.1	46.7	5.8
	B	1.25	1.31	1.92	2.9	6.8	19.0	48.1	6.4
	C	1.43	1.26	1.77	3.4	7.2	19.1	47.5	6.5
	Average	1.29	1.22	1.73	3.0	6.6	18.3	47.4	6.2

Table 5.9: WERs (in %) for test sets A - C obtained with a 3-layer hybrid RCN-HMM and AFE as the input features.

viously more difficult to make and the 2K + 4K + 16K system performs significantly worse. However, we did not investigate to what extend this is due to the fact that the outputs of the the first layer now have a larger bandwidth, which calls for other reservoir parameters than the ones we derived from the outputs dynamics of a first layer comprising a big reservoir.

5.8.3 System Evaluation Using the AFE

Recalling the significant gain obtained by replacing MVN features by advanced front-end (AFE) features in a GMM-HMM recognizer, we were curious to examine the effect of doing the same in an RCN-based recognizer. The results for a 3 $\times$ 8K system driven by AFEs are listed in Table 5.9.

Apparently, removing some of the noise effects in the front-end also helps an RCN-based system to some extend (in the clean training experiment for instance, the aWER is reduced from 11.5% to 9.9%). However, its effectiveness is much smaller than in GMM-HMM systems.

5.9 Conclusions

We have studied reservoir-based acoustic modeling for noise robust continuous digit recognition. A reservoir based acoustic model computes the state likelihoods in an HMM by means of a two-layer recursive neural network. This network is peculiar in the sense that it consists of a hidden layer of recurrently connected non-linear neurons with fixed (i.e. non-trained) coefficients – called a reservoir – and an output layer of linear neurons with trained coefficients which 'read out' the outputs of the reservoir.

A major contribution of the paper is the introduction of a novel analysis of the reservoir as a non-linear dynamical system, an analysis that also offers insight in the behavior of the system. Based on that insight, empirical findings are translated into a few very simple and comprehensible rules which permit anyone to design a reservoir system that is properly tuned to the dynamical properties of the reservoir inputs (can be measured) and the dynamical properties of the expected outputs (can be estimated from knowledge of the recognition problem). These simple rules made it feasible to investigate the performance of RCN-based systems as a function of reservoir size and number of states per digit in a systematic way.

The main objective of our work was of course to demonstrate that reservoir computing can lead to robust acoustic models. In a first step, we developed a single layer RCN-based continuous digit recognition (CDR) system, driven by mean and variance normalized MFCC features. From experiments on Aurora-2 we established that this system is indeed significantly more robust against the presence of noise than a traditional GMM-HMM system working with the same acoustic parameters, whilst it is competitive in the clean condition. Under noisy conditions, the RCN-HMM system even outperforms a GMM-system working with the Advanced Front-End features, but the latter system still stands out under clean conditions.

The experimental data presented in the paper also reveal that noise-robustness mainly follows from the random fixation of the reservoir weights. Properly tuning the reservoir dynamics mainly raises the accuracy in matched conditions without compromising the noise-robustness.

Motivated by the success of single-layer systems, we also considered multi-layer systems comprising multiple reservoirs. We demonstrated that such systems can further improve the results for the matched test conditions, again without compromising the noise robustness in the noisy conditions. Our best multi-layer systems are now competitive with the AFE-GMM system in clean conditions as well. Our present system outperforms all other neural-based approaches we know of that were recently evaluated for continuous digit recognition. A particular advantage of reservoir based models seems to be their robustness against over-training.

We hope that our research will motivate others to develop new ideas that can further improve the performance of the RCN-based speech recognition systems and that, in time, can contribute to the design of more noise robust systems for other tasks such as large vocabulary speech recognition.

6
Combining GMM and RCN for Acoustic Modeling

This chapter is an edited version of the following original publication:

[4] A. Jalalvand, K. Demuynck, and J.-P. Martens, "Noise robust continuous digit recognition with reservoir-based acoustic models," *Proceedings of the International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, p. ID:99, 2013.

6.0 Preface

The results of my previous work show that although RCN-HMM hybrids offer more robustness to noise than the competitors, they are regularly outperformed by these competitors in clean speech conditions. This observation motivated me to search for combinations of GMMs and RCNs in a single system that hopefully inherits the strengths of both acoustic modeling techniques. In this chapter, I investigate three model combination techniques: two so-called tandem techniques and a state-level likelihood fusion technique. The experiments show, unfortunately, that only marginal improvements of the performance in matched conditions can be achieved, and moreover, that these improvements come at the expense of a degraded robustness against the presence of noise. As raising the noise robustness remains the major over-all objective of my research, I also investigated more in detail the effects of the front-end on the recognition accuracy. Furthermore, I introduce bi-directional reservoir systems (which can take account of the right context as well) and establish that they are effective in further improving the noise robustness, be it at the expense of a small loss in the clean speech performance again.

6.1 Introduction

Enhancing the noise robustness of automatic speech recognition (ASR) systems is still an active area of research. In this work, we focus on robust continuous digit recognition (CDR). CDR is essential for the recognition of spoken numerical data (e.g., PIN-codes) in many applications which are often operated in a noisy environment (e.g., in the street) and utilized by accented non-native speakers (e.g., tourists) as well as native speakers. The absence of a language model also makes CDR an attractive setup to evaluate the robustness of acoustic modeling techniques.

A modern ASR system treats CDR as a statistical pattern recognition problem which aims at finding the most likely interpretation of a stream of acoustic feature vectors emerging from an acoustic front-end. The recognition is achieved by a back-end comprising one left-to-right multi-state Hidden Markov Model (HMM) per digit. In most systems, the likelihood to observe a given feature in a certain state is estimated by means of a Gaussian Mixture Model (GMM). Although state-of-the-art systems can reach high accuracy on low-noise test utterances, they are still susceptible to severe degradations in noisy conditions.

In recent years, many strategies for improving the noise robustness have been proposed [148]. Most of them deal with the speech signal pre-processing in the acoustic front-end and aim to retrieve acoustic features that represent the clean speech component of a noisy signal [26, 113, 131]. Other methods take the impact of the noise into account in a consistent way during the likelihood computation in a GMM-based back-end [124]. Finally, there have also been several attempts to improve robustness by means of alternative acoustic modeling techniques such as neural networks [142] or Support Vector Machines (SVM) [149].

In recent work, we investigated the potential of Reservoir Computing Network (RCN) [99] as an alternative approach. Reservoir Computing employs reservoir networks – a particular kind of Recurrent Neural Networks (RNN) [79] – as complex dynamical systems that can analyze an incoming input vector stream. The hypothesis is that such a dynamical system can be designed to focus on the speech dynamics, and thus, to be less sensitive to the dynamics of the noise. We were already able to devise an RCN-based

CDR system that attains competitive recognition accuracies in clean conditions and that outperforms most other systems in noisy conditions [1, 2].

In this paper, we extend this previous work. In particular, we investigate whether RCN-based systems can profit from front-end techniques that were shown to work well in combination with traditional GMM acoustic models. Furthermore, we investigate more complex hierarchical RCN-based systems comprising multiple reservoirs which resemble to some extend the deep neural architectures that have been proposed for continuous speech recognition [9].

Since reservoirs only build up some memory of the recent past, we also test bi-directional systems combining two reservoirs: one that processes the speech frames from left to right and one that processes them from right to left. Such bi-directional systems were already demonstrated to improve phone recognition in continuous speech [9], but they were not yet applied to CDR. Finally, we also study ways of combining reservoir networks and GMMs in a single system.

The rest of the paper is organized as follows: Section 6.2 provides a concise outline of the basic principles of RCN, Section 6.3 describes ways of integrating reservoir networks in an RCN-HMM hybrid, Section 6.4 reviews three approaches that were tested for combining reservoir networks with GMMs in a single system and Sections 6.5 and 6.6 summarize the experimental setup and the results obtained with it. The paper ends with conclusion and future work.

6.2 Reservoir Computing Networks

The basic principle of reservoir computing is that information can be retrieved from sequential inputs by means of a two-layer recurrent neural network with the following characteristics (see Figure 6.1). The first layer is a sparsely connected hidden layer, composed of non-linear neurons which, at time t, are driven by inputs U_t and by delayed hidden layer outputs R_{t-1}. The output layer consists of linear neurons which are driven by the hidden layer outputs R_t. Important is that the weights of the hidden neurons are fixed, and only the weights of the output layer are optimized according to a least squares linear regression.

The hidden layer can be envisioned as a reservoir of recurrently interconnected computational neurons, driven by inputs. Together with the output layer, it forms a *reservoir computing network*. The network outputs Y_t are usually called *readouts* [99] so as to differentiate them unambiguously from the reservoir outputs R_t. In order to become less sensitive to random inter-frame changes in the inputs (e.g., changes due to the spectral analysis

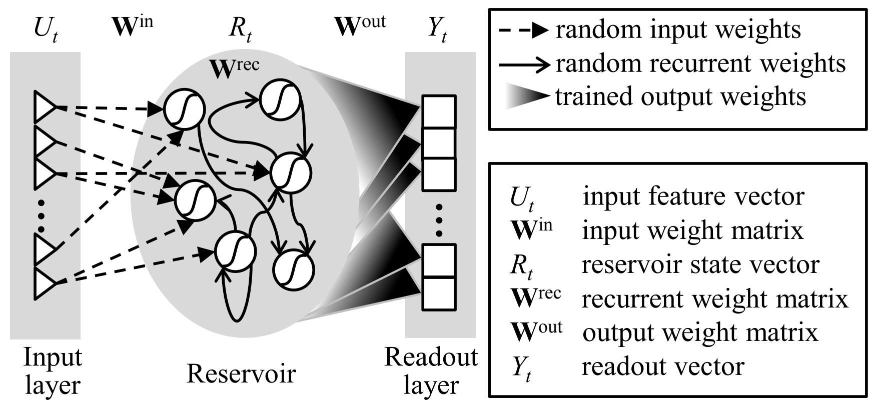

U_t
$\mathbf{W}^{\text{in}}$
R_t
$\mathbf{W}^{\text{out}}$
Y_t
$\mathbf{W}^{\text{rec}}$
Input layer
Reservoir
Readout layer
random input weights
random recurrent weights
trained output weights
U_t input feature vector
$\mathbf{W}^{\text{in}}$ input weight matrix
R_t reservoir state vector
$\mathbf{W}^{\text{rec}}$ recurrent weight matrix
$\mathbf{W}^{\text{out}}$ output weight matrix
Y_t readout vector

Figure 6.1: A basic RCN system consists of a reservoir and a readout layer. The reservoir is composed of interconnected **non-linear** neurons with randomly **fixed** weights. The readout layer consists of **linear** neurons with **trained** weights.

or the ambient noise), one can introduce leaky integration in the reservoir neurons (so-called Leaky Integrator Neurons [100]). The resulting reservoir network is governed by the following equations:

$$R_t = (1-\lambda)R_{t-1} + \lambda\, f_{res}(\mathbf{W}^{in}U_t + \mathbf{W}^{rec}R_{t-1}), \quad (6.1)$$

$$Y_t = \mathbf{W}^{out}R_t, \quad (6.2)$$

with a leak rate λ between 0 and 1, with $\mathbf{W}^{in}$ and $\mathbf{W}^{rec}$ containing the input and recurrent weights to the reservoir neurons, and with $\mathbf{W}^{out}$ containing the weights of the output neurons.

As mentioned before, the weights of the hidden neurons are fixed by means of a random process characterized by four control parameters (see [3] for more details). These parameters are: (1) α_U, the maximal absolute eigenvalue of the input weight matrix $\mathbf{W}^{in}$, (2) ρ, the maximal absolute eigenvalue of the recurrent weight matrix $\mathbf{W}^{rec}$, (3) K^{in}, the number of inputs driving each reservoir neuron and (4) K^{rec}, the number of delayed reservoir outputs driving each reservoir neuron. The first two parameters control the strengths of the input and the recurrent stimulations of a reservoir neuron, the latter two control the sparsity of the input and recurrent weight matrices. Together with λ they constitute the reservoir control parameters which have to be properly set in order to assure that the reservoir is well behaved. Note that any effective reservoir should at least have the so-called echo state property. It states that, with time, the reservoir should forget about the initial state it was in. That is also why a reservoir network was originally called an Echo State Network [99]. It was shown in [99] that the echo state property holds if ρ – called the spectral radius – is smaller than 1.

The reservoir can be envisioned as a predefined but complex non-linear dynamical system that performs a temporal analysis of the input stream. We claim that such a system can extract features that are not so easily corrupted by the presence of noise whose dynamics differ from the speech dynamics.

The output weights are determined so that they minimize the mean squared difference between the readouts Y_t and the desired readouts D_t across the training examples [2]. As a consequence, they follow from a set of linear equations. The desired output D_t is a unit vector with a non-zero entry at the position corresponding to the desired HMM-state at time t.

6.3 Speech Recognition with Reservoirs

In this section, we introduce the architectures that were conceived to perform CDR by means of a reservoir network.

6.3.1 A Hybrid RCN-HMM

Like any other neural network based hybrid system [150], a hybrid RCN-HMM assumes that every network output corresponds to an HMM state. It transforms these outputs to state likelihoods and performs a standard Viterbi search for the best path through a looped HMM. So, the readouts $y_{t,i}$ (with i indexing the network outputs) are transformed to new outputs $z_{t,i}$ that are appropriate estimations of the scaled likelihoods $P(U_t|q_t = i)/P(U_t)$. Using these outputs, one can determine the best state sequence as

$$\hat{\mathbf{q}} = \arg\max_q P(\mathbf{q}|\mathbf{U}) = \arg\max_q \prod_{t=1}^{T} z_{t,q_t}\, P(q_t|q_{t-1}),$$

and derive the digit sequence thereof. The admissible state sequences can represent an arbitrary sequence of digits (possibly interleaved with silences). An extra transition probability P_0 is introduced on the transition from the final to the initial state that controls the balance between deletions and insertions.

The reservoir network can be a simple network with one reservoir, but it can as well be a hierarchical network, obtained by stacking multiple reservoir networks (called layers) on top of each other (see Figure 6.2). The argument for cascading layers is that new layers can correct some of the

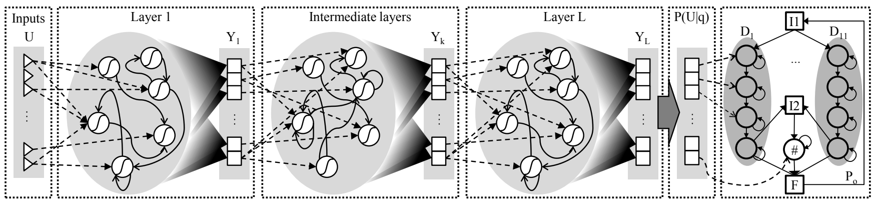


Figure 6.2: Architecture of an RCN-HMM hybrid comprising a multi-layer reservoir network for CDR. The HMM has two initial states (I1 and I2), one final state (F) and it comprises 11 multi-state digit models (D1 ... D11) and a single state silence model (#)

mistakes made by the preceding layers because they offer additional temporal modeling capacity and a new inner space to model the state distributions. This argumentation is supported by experiments showing enhanced digit and phone recognition in continuous speech [2, 9]. The layers are trained one after the other and per layer good settings of the reservoir control parameters emerge from an efficient user-controlled search procedure (see [3]).

The readouts of each layer are assumed to represent the same set of HMM states, and consequently, their weights are trained so as to minimize the mean squared difference between the computed readouts Y_t and the same desired readouts D_t. Under these circumstances, the readouts are in each layer assumed to adhere to posterior probabilities $P(q_t = i|U_t)$ and $z_{t,i} = y_{t,i}/P(q_t = i)$ is an appropriate estimation of the envisioned scaled likelihoods. However, since $y_{t,i}$ is not confined to [0,1] and since likelihoods have to be positive, one has to map it to a variable that is confined to [0,1], before it can be imputed in the formula for $z_{t,i}$. As explained in [2] this mapping can be accomplished by a sigmoid function, a non-parametric function derived from histograms or a simple clip-and-scale method. Since we established experimentally that the three methods yield basically the same results, we opted for the simple clip-and-scale method here:

$$z_{t,i} = \frac{\max(y_{t,i}, y_o)}{\max_j(y_{t,j})} \frac{1}{P(q_t = i)}, \quad y_o \ll 1 \tag{6.3}$$

6.3.2 Bi-Directional RCN-HMM

Reservoirs only provide a fading memory of the past and make no use of the future. On the other hand, the theory of co-articulation states that a phone is also influenced by the forthcoming phone. In order to account for such anticipation as well, we introduce bi-directional reservoir networks.

Such a network encompasses two identical reservoirs, a forward reservoir that processes the data stream from left-to-right and a backward reservoir that processes them from right-to-left (see Figure 6.3).

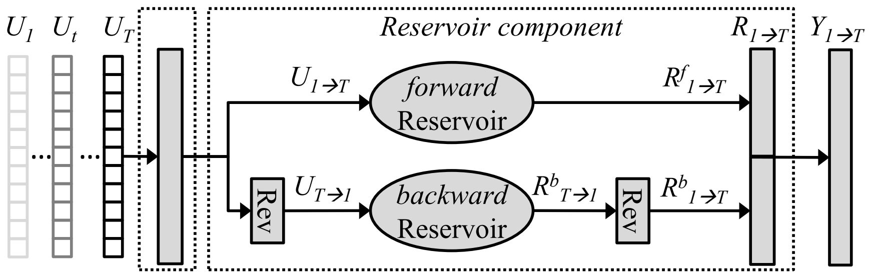


Figure 6.3: Architecture of a bi-directional RCN. The forward reservoir processes the inputs $U_{1\to T}$ whereas the backward reservoir processes the inputs $U_{T\to 1}$. The outputs of the latter reservoir are then time reversed before combining them with the outputs of the former reservoir.

The forward reservoir state at time t is the state that it has reached after having processed the input vectors $U_1 \dots U_t$. The backward reservoir state at time t is the state that it has reached after having processed the input vectors $U_T \dots U_t$, where T is the length of the input sequence **U**. The two reservoir states at time t are supplied to a single output layer which computes the readouts $y_{t,i}$ to be employed in the equations, as originally proposed in [9].

6.4 Model Combination

It is clear that the RCN-based likelihoods are computed in a large and randomly fixed high-dimensional feature space (the reservoir state space) that is affected by long-term dynamics. On the other hand, the GMM-based likelihoods are computed in a well conditioned low-dimensional feature space (the acoustic feature space) that solely describes local dynamics. We argue that this implies that they may attribute complementary information, and consequently, that it makes sense to combine them in a single system. In this section, we propose three approaches to do so, two tandem approaches and one likelihood-fusion approach.

6.4.1 An RCN-GMM Tandem (T-RCN-GMM)

In a so-called RCN-GMM tandem, the readouts Y_t would be supplied as acoustic features to a traditional GMM-HMM system. However, it is not

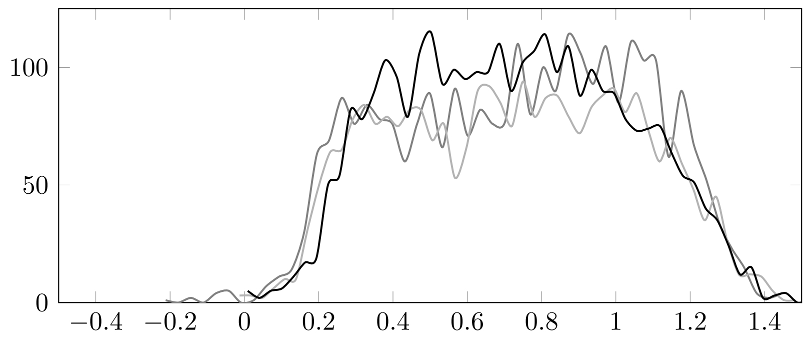


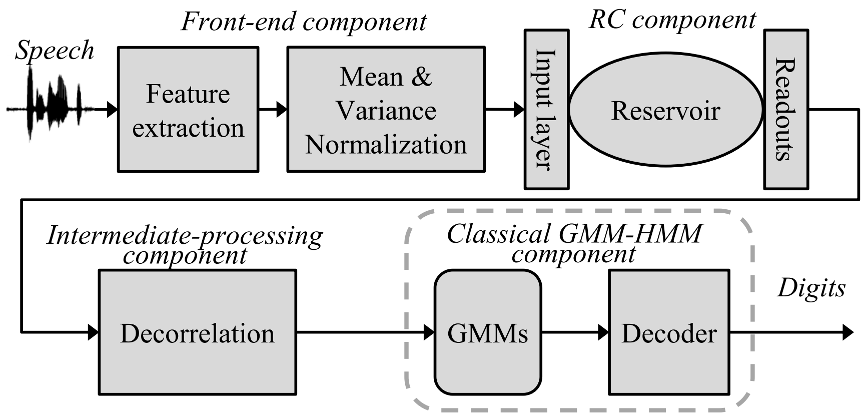
Figure 6.4: Histograms of three randomly selected readouts on the frames that were assigned to the their corresponding states by the Viterbi search.

evident that the Y_t qualify well as inputs to a modeling system that relies on mixtures of Gaussian distributions with diagonal covariance matrices. Therefore, it is common in MLP-GMM tandems [151] (MLP stands for Multi-Layer Perceptron) to perform a non-linear transformation of the Y_t to make them more Gaussian-like and to apply a decorrelation to attain independent features.

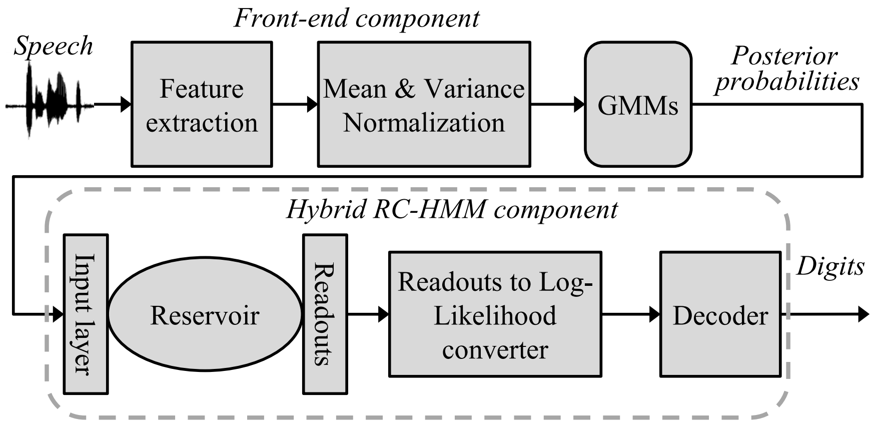
Typical non-linear transformations employed on MLP-outputs are a logarithm or an inverse sigmoid. Their main aim is to reduce the skewness of the MLP output distributions. However, reservoir network outputs are linear combinations of zero mean reservoir state variables. Hence, they are not hard-limited to the range of [0,1] and therefore, and consequently, they do not exhibit skewed distributions, as is demonstrated by the histograms depicted in Figure 6.4. Just to be sure, we did a few experiments with non-linear transformations, and they confirmed that no transformation is needed and that the T-RCN-GMM architecture can be reduced to the scheme depicted in Figure 6.5(a). The decorrelation is maintained and is achieved by means of a Mutual Information Discriminative Analysis (MIDA) [152], a technique that can be regarded as a special form of Linear Discriminant Analysis (LDA).

6.4.2 A GMM-RCN Tandem (T-GMM-RCN)

Another type of tandem is a GMM-RCN tandem in which the likelihoods computed by the GMMs are supplied to an RCN-HMM back-end. Since RCN does not make any assumptions regarding the distributions of the individual inputs nor about the correlations between these inputs, the GMM outputs can be supplied to the RCN-HMM component without any transformation or decorrelation, as shown in Figure 6.5(b). Note that since we keep



(a) An RCN-GMM tandem architecture. The RCN outputs are decorrelated before they are supplied to the GMM-HMM component.



(b) A GMM-RCN tandem architecture. The GMM outputs can be supplied as such to the RCN-HMM component.

Figure 6.5: An RCN-HMM tandem architecture: the front-end, the RCN component, the intermediate-processing of the readouts and the GMM-based decoder.

K^{in} fixed, increasing the number of GMMs (by raising the number of digit state) does not raise the computational load of the RCN-HMM component.

6.4.3 Likelihood Fusion (F-GMM-RCN)

Since reservoir-based likelihoods are computed in a high-dimensional feature space which was randomly fixed and designed to expose long-term memory effects, they may differ considerably from GMM likelihoods that are computed in a low-dimensional space of well established local features. Consequently, it seems sensible to fuse those likelihoods in the Viterbi search.

If the two likelihood sets apply to the same HMM states (digit states + silence state) the fusion is straightforward to achieve by considering a weighted mean of the two log-likelihoods as the state log-likelihood that drives the Viterbi search. Obviously, such a state-level combination scheme assumes that the composing acoustic models are kind of time synchronous, meaning that they support state transitions at the same time instances. This may not be entirely true but it drastically simplifies the decoding. In our experiments, we pursued time synchrony by imposing the state-level segmentation provided by the reservoir during GMM training. Actually, it is a reasonable assumption in our case because both types of acoustic models in our experiments were bootstrapped from exactly that segmentation of the training data.

There are two popular ways of computing a mean likelihood. One is to compute a (weighted) linear combination of likelihoods, another is to compute a (weighted) linear combination of the log-likelihoods. The former strategy is believed to be ideal for reducing the effects of noise on the likelihoods, whereas, the latter is believed to be preferable for combining complementary information streams as it better complies with the log-linear combination of likelihoods in the Viterbi search. As we contemplate that the two likelihoods attribute complementary information, we opt for the log-linear combination approach. For simplicity, we consider just one weight, irrespective of the state. The value of this so-called stream weight is determined from recognition experiments on the development data.

6.5 Experimental Setup

In this section we present the experimental framework that was adopted to test the proposed approaches.

6.5.1 Speech Corpus

All experiments are conducted on the Aurora-2 database [29]. This database contains clean and noisy utterances, sampled at 8 kHz and filtered with either a G712 or a MIRS filter. There are 8440 clean training samples, each counting 1 to 7 digits. The corpus also includes various noise corrupted copies of each clean utterance. The noisy utterances were created by artificially adding different noise types in different degrees, leading to Signal-to-Noise Ratios (SNR) between 20 and -5dB. The vocabulary consists of the digits 0 to 9 and the letter 'oh' (a substitute for 'zero'). We adhere to the training and test sets that were defined as part of the Aurora-2 benchmark. We have trained systems on clean speech (i.e., clean speech training) and on clean + noisy speech (i.e., multi-style training) and tested them on the test sets A - C.

6.5.2 Front-End Setups

We investigate three acoustic feature sets: the conventional MFCCs (log energy and $c_1 \dots c_{12}$), the 24 log Mel filterbank log-energies (MelFB), and the ETSI Advanced Front-End features (AFE) (denoised $c_0 \dots c_{12}$ without non-speech frame dropping) [131]. In all cases, the analysis is performed on 30 ms Hamming-windowed frames and the hop size between frames is $\tau_{fr} = 10$ ms. In order to provide some context information, each feature set is supplemented with Δ and Δ^2 features. The frame-wise feature extraction is followed by an utterance-wise normalization that creates zero-mean and unit-variance features.

Note that due to the zero mean input weights of the reservoir, the reservoir activations and the reservoir outputs will be zero mean, even in the presence of noise. Consequently, mean normalization is not a necessity in the case of RCN-based systems. Nevertheless, for reasons of uniformity and because it does hardly take extra CPU-time we do apply it in all system configurations.

6.5.3 Reservoir Component Setup

Based on previous work, the reservoir control parameters were determined in the same way for each layer. Defining $\tau_\lambda \doteq -\tau_{fr}/\ln(1-\lambda)$ as the leaky integration time constant and T as the expected state duration, we select $(\rho, \tau_\lambda, K^{in}, K^{rec}) = (0.8, T, 10, 10)$. The parameter α_U is chosen so that the average variance of the reservoir outputs reaches a certain level [3]. Since layers 2 and 3 see basically the same type of inputs, α_U is taken the same for both layers.

The size of the reservoir – defined as the number of neurons it contains (N^{res}) – is considered to be an independent variable. Note that since K^{in} and K^{rec} are kept fixed to 10, the CPU-time needed for calculating the readouts scales linearly with the size of the reservoir.

The reservoir networks are trained by means of a Tikhonov regression [144]. When comparing bi-directional to uni-directional systems we compare systems with the same number of trainable parameters. This implies that a bi-directional system contains two reservoirs of half the size of the reservoir embedded in the uni-directional system it is compared to.

Each digit is modeled by a 7-state left-to-right HMM whilst the silence is modeled by a single state.

6.5.4 Model Combination Setup

In order to investigate the proposed model combination strategies, we needed a GMM-based component incorporating states that directly map to the reservoir network outputs. It was created with the SPRAAK toolkit[1], meaning that it works with semi-continuous HMMs, that is, all GMMs select members from the same global pool of Gaussians that emerged from an unsupervised clustering procedure. Consequently, there is an extensive parameter tying which is beneficial for small databases such as Aurora-2. The number of Gaussians and the number of selected Gaussians per state are determined automatically from the size and the statistics of the data, so that the risk of over-fitting is low.

6.5.5 Evaluation Setup

In the development phase, two thirds of the training set are used for training, the held out third is used for control parameter optimization (e.g., the transition probability P_o). In the final evaluation phase the acoustic models are trained on the complete training set but using the control parameters that were optimal in the development phase. In this paper, we only report the results of the final evaluation experiments.

We report average Word Error Rates (WERs) on tests A-C for all SNRs, and we consider both clean speech training and multi-style training. In multi-style training the training set consists of clean utterances as well as utterances with SNRs between 20 and 5dB.

6.6 Experimental Results

In this section we review the results obtained with the proposed approaches and we compare them to reference results published in the literature.

[1]SPRAAK: Speech Processing, Recognition and Automatic Annotation Kit [http://www.spraak.org]

6.6.1 Reference Systems

First of all, we report some state-of-the-art reference system performances (see Table 6.1). In particular, we consider the ML-based GMM systems using AFE-features proposed in [153], the ML-based and MCE-based GMM systems proposed in [113] and [26], two GMM systems embedding more sophisticated back-ends based on joint uncertainty decoding (JUD) and Vector Taylor Series (VTS) respectively [124] and the tandem system embedding deep belief networks reported in [142].

The figures show that the impact of the front-end on the WER is more important in the case of clean speech training experiments than it is in the case of multi-style training. In the case of clean speech training, the AFEs attribute a lot of noise robustness while maintaining the accuracy in clean speech conditions. In the case of multi-style training, it does not offer more noise robustness but it leads to a higher accuracy than MVN in the clean speech condition. The figures further show that advanced back-end techniques (JUD and VTS) lead to a significant gain in noise robustness, but it is not clear how they affect the accuracy in the clean speech condition (the papers do not mention these figures). Finally, the deep belief networks do not seem to be effective in any of the conditions.

6.6.2 Self-Developed GMM Systems

As indicated before, we need a GMM system with the same number of states as the RCN-system if we want to apply state-level likelihood fusion in an elegant way. Furthermore, if we want to investigate the impact of the front-end, we also need to work with self-developed GMM-HMM systems for each choice of the front-end. We denoted these self-developed GMM-HMM systems by the front-end they embedded and by the acronym SPRK (see Table 6.1) to designate that they were developed using the SPRAAK toolkit.

Compared to the reference GMM (AFE) system the self-developed systems offer a much superior clean speech performance, but at the expense of lower noise robustness, as mainly manifested in the clean speech training experiments. The very bad performance of the MelFB features in the clean speech training case may be due to the fact that the very large negative log-energies which appear in clean speech are completely absent in noisy speech. Imposing a limit on these negative values would be a simple technique to reduce the effect.

System	Clean			Multi		
	Clean	0-20	-5dB	Clean	0-20	-5dB
GMM (AFE) [153]	0.77	13.2	69.9	0.83	8.4	59.2
GMM (MVN) [113]	0.84	19.7	82.2	1.77	8.5	59.1
GMM (MVN-MCE) [26]	0.41	15.7	77.2	0.92	6.4	55.3
GMM (VTS) [124]	-	9.4	-	-	-	-
GMM (JUD) [124]	-	10.3	-	-	-	-
T-DBN-GMM [142]	1.26	21.0	74.6	-	-	-
GMM (MelFB-SPRK)	0.24	51.2	92.9	0.39	7.1	58.2
GMM (MVN-SPRK)	0.24	20.7	81.1	0.59	8.3	66.3
GMM (AFE-SPRK)	**0.20**	15.5	74.2	**0.39**	6.2	54.2
RCN (MelFB)	0.59	12.3	73.9	0.98	6.1	51.1
RCN (MVN)	0.78	11.8	63.5	1.28	7.1	52.0
RCN (AFE)	0.82	10.0	58.4	1.31	6.2	47.5
biRCN (MelFB)	0.75	10.9	64.0	1.07	**5.5**	45.5
biRCN (MVN)	0.96	11.1	60.5	1.47	6.3	47.1
biRCN (AFE)	0.86	**9.0**	**54.4**	1.43	5.8	**43.3**
T-GMM-biRCN (AFE)	0.87	16.2	73.1	1.28	7.3	54.0
T-biRCN-GMM (AFE)	0.77	10.6	58.7	1.19	5.7	43.9
F-GMM-biRCN (AFE)	0.53	10.8	63.8	0.80	5.4	46.6

Table 6.1: Comparing average WERs (in %) per condition for test sets A-C of Aurora-2 using a 3-layer hybrid RCN-HMM for both clean and multi-style training.

6.6.3 Impact of the Front-End on RCN-HMM Hybrids

In a first experiment, we test three acoustic feature sets in combination with RCN-HMM hybrids incorporating a three-layer reservoir network, with each layer embedding a reservoir of 8K neurons. The results in Table 6.1 show that for clean speech training, the AFE features lead to significant improvements in noise robustness. The differences are significant in moderate noise conditions (from 12.3% to 10.0% WER for SNRs between 0 and 20 dB) and substantial in the strong noise condition (from 73.9% to 58.4% WER for a SNR of -5 dB). In the case of multi-style training, the impact of the front-end is much more modest, as it was for the GMM-based systems. Like in [9], we also tested larger reservoirs (up to 32K nodes) and more layers, but they yield only a small extra gain in performance for a substantial increase of the computational load.

It is clear that the noise robustness of RCN-HMM hybrids is better than

	Set	Clean	20	15	10	5	0	-5	0-20dB
Clean	A	0.82	1.64	2.26	4.8	10.6	26.1	55.5	9.1
	B	0.82	1.66	2.39	4.2	9.5	24.4	53.3	8.4
	C	0.93	1.75	2.73	5.1	11.0	27.0	54.2	9.5
	Avg.	0.86	1.68	2.46	4.7	10.3	25.8	54.4	9.0
Multi	A	1.37	1.30	1.69	2.8	5.8	15.6	42.6	5.4
	B	1.37	1.53	2.01	3.0	6.2	16.8	43.8	5.9
	C	1.54	1.48	2.13	3.2	6.7	16.7	43.5	6.1
	Avg.	1.43	1.44	1.94	3.0	6.2	16.3	43.3	5.8

Table 6.2: WERs (in %) for test sets A - C obtained with a 3-layer bi-directional hybrid RCN-HMM and the AFE features.

that of GMM-based systems with a traditional back-end, but that they cannot compete with the self-developed GMMs on clean speech utterances. An important finding is that with clean speech training an RCN-HMM with a simple back-end can compete with a GMM system incorporating a much more complex VTS-based back-end.

6.6.4 Impact of Bi-Directional Processing on RCN-Based Systems

In a second experiment, we test three-layer bi-directional reservoir systems with two 4K-node reservoirs per layer (biRCN-HMM). The results in Table 6.1 show that bi-directional processing offers extra noise robustness at the expense of a small loss in clean speech performance. The bi-directional system now compares favorably to the best GMM system, the one embedding a VTS back-end.

Comparing the average WERs for the 0-20dB conditions shows that a bi-directional system yields around 10% and 6% relative improvement (on clean and multi-style training) over a unidirectional system, without much changing the complexity of that system. Table 6.2 provides the WER of such a 3-layer bi-directional RCN-HMM per test set and per SNR.

6.6.5 Impact of Model Combination

In a third experiment we investigate the effect of the proposed model combination techniques on the system performance and the noise robustness. We employed our best system that utilizes the AFE as the front-end and a bi-directional reservoir as the RCN-component (biRCN-AFE).

In the case of tandem systems, feeding the GMM likelihoods into a reservoir system is apparently not a good idea. Supplying the reservoir network outputs to a GMM system does not hurt, but it does not help either. Likelihood fusion on the other hand can help to improve the clean speech recognition performance while maintaining most of the noise robustness. However, the bottom line is that model combination does not lead to improved noise robustness.

6.7 Conclusion and Future Work

In this paper we studied reservoir based acoustic modeling for noise robust continuous digit recognition. A reservoir based acoustic model computes the state likelihoods in an HMM by means of a two-layer recursive neural network. This network is peculiar in the sense that it consists of a hidden layer of recurrently connected non-linear neurons with fixed (i.e., non-trained) coefficients – called a reservoir – and an output layer of linear neurons with coefficients that can be trained using a simple Tichonov regression method. A particular advantage of reservoir networks is that they are not easily over-trained.

The main objective of our work was to demonstrate that an RCN-based system comprising a cascade of reservoir networks can outperform GMM-HMM systems in noisy conditions with different front-ends. The introduction of noise robust features (AFE) and bi-directional reservoir networks clearly lead to lower WERs, both in matched and mismatched conditions. Our present systems now outperform all other neural-based approaches we know of that were recently evaluated for continuous digit recognition.

We also investigated different combinations of RCN-based and GMM-based systems to find a way of improving the reservoir performance in clean conditions. Particularly, we introduced RCN-GMM and GMM-RCN tandems as well as a simple fusion approach to combine the reservoir and GMM likelihoods. Our experiments showed that although adding the information from a GMM marginally improves the performance in the matched condition, it degrades the performance in the mismatched environments.

Given the above observations, our future research will investigate more front-end and back-end approaches that can further improve a hybrid RCN-HMM system. One direction is to train a reservoir network to denoise the acoustic features, another is to combine an RCN-HMM with uncertainty decoding. Another plan is to evaluate the potential of reservoir systems in the context of noise robust large vocabulary recognition (e.g., Aurora-4).

Part III

Feature Denoising and Image processing

7 Denoising the Acoustic Features Using RCNs

This chapter is an edited version of the following original publication:

[5] A. Jalalvand, K. Demuynck, and J.-P. Martens, "Feature enhancement with a reservoir-based denoising auto-encoder," *Proceedings of the International Symposium on Signal Processing and Information Technology (ISSPIT)*, p. 6, 2013.

7.0 Preface

In spite of many techniques that can be employed in the back-end of an ASR, front-end approaches which aim to enhance the acoustic features are more appealing because they can be almost equally effective at a much lower computational cost. One interesting front-end technique to deal with additive noise is feature denoising. A system that converts noisy features to clean features is called a denoising auto-encoder (DAE). The motivation of using a reservoir computing network (RCN) as a DAE is that thanks to its long-term memory, RCN may better distinguish between the speech and noise dynamics than e.g., multi-layer perceptrons, the most popular neural network type that has been used for DAE. The experiments described in the current chapter show that the RCN-based DAE does a good job, but that it is not more powerful than the more conventional signal processing approaches such as Wiener-filtering and Voice-activity detection that are applied in the ETSI advanced front-end (AFE).

7.1 Introduction

If one wants to apply automatic speech recognition (ASR) on mobile devices, one needs an ASR that is both very accurate and robust against the presence of noise and channel distortions. Despite many years of work, achieving robust recognition remains a big challenge.

Since an ASR is composed of a front-end (for extracting the acoustic feature vectors) and a back-end (for decoding these feature vectors), two types of techniques can be envisioned to tackle the problem. One is to enhance (denoise) the feature vectors in the front-end [113, 131], and the other is to take account of the noise during feature decoding in the back-end [124, 134].

Typical feature enhancement is achieved by signal processing techniques such as Wiener filtering (single-channel) and beam-forming (multi-channel). They form the basis of robust feature extractors such as the Advanced Front-end (AFE) [131] and SPLICE [154]. A bit less popular technique is to train a neural network to convert noisy feature vectors to clean vectors [117]. Such a network is called a Denoising Auto-Encoder (DAE).

In previous work, we showed that a non-linear dynamical system with memory, such as a reservoir computing network (RCN) [98, 99] can offer good noise robust recognition starting from non-denoised features [1, 2]. Nevertheless, more recently we discovered that their noise robustness can be further increased by supplying them with denoised features such as the AFE features [4]. The question we want to tackle here is whether a system working with standard features and incorporating an RCN-based DAE would be able to outperform a standard system working with AFE features.

The rest of the paper is organized as follows: Section 7.2 describes how to construct an RCN-HMM hybrid for continuous digit recognition (CDR), Section 7.3 reviews the way we have developed several RCN-based DAEs and Sections 7.4 and 7.5 summarize the experimental framework (Aurora-2) and the results obtained within this framework. The paper ends with conclusions and future work.

7.2 A Hybrid RCN-HMM for Speech Recognition

An RCN-HMM hybrid works with an HMM that represents the task and a neural network that is supposed to convert the inputs U_t at time t into HMM state likelihoods. The search for the best path through the HMM

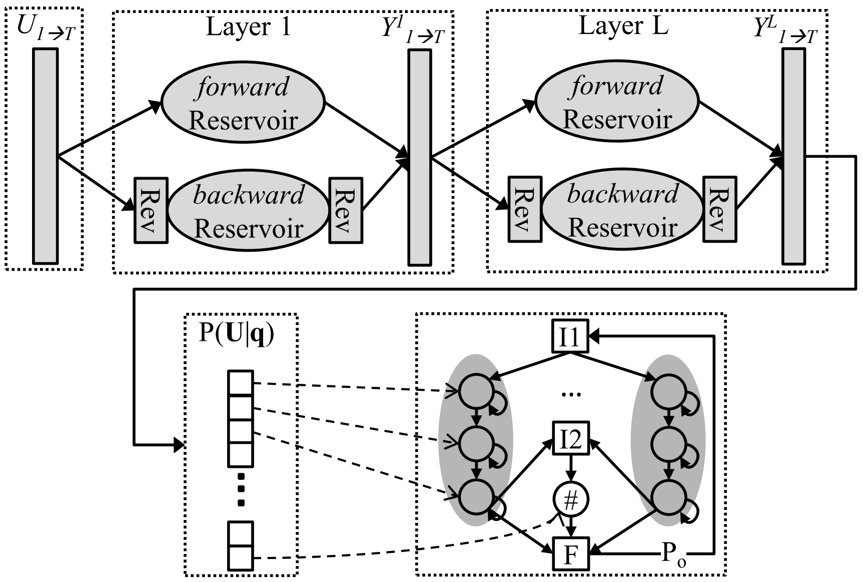


Figure 7.1: Architecture of an RCN-HMM hybrid comprising a multi-stage bi-directional reservoir networks for CDR. The HMM has two initial states (I1 and I2), one final state (F) and it comprises 11 multi-state digit models and a single state silence model (#)

is found using a Viterbi search. In the case of an RCN-HMM hybrid, the readouts $y_{t,i}$ (with i indexing the readouts) are assumed to resemble the posterior probabilities $P(q_t = i|U_t)$. This means that $z_{t,i} = y_{t,i}/P(q_t = i)$ is a scaled likelihood and consequently, that the best state sequence follows from

$$\hat{\mathbf{q}} = \arg\max_q P(\mathbf{q}, \mathbf{U}) \approx \arg\max_q \prod_{t=1}^{T} z_{t,q_t}\ P(q_t|q_{t-1}),$$

Figure 7.1 shows the architecture for the case of continuous digit recognition (CDR) and a multi-stage RCN in which each network output is supplied to the next stage [4]. The transition probability P_0 which is added to the digit loop controls the balance between deletions and insertions.

Since $y_{t,i}$ is not confined to [0,1] it is first mapped to that interval before computing $z_{t,i}$. The mapping is achieved by a simple clip-and-scale approach, as described in [2,4]. The different stages of the RCN are trained independently, one after the other.

As in [4,9], we use bi-directional RCNs in the different layers. Such an RCN encompasses two (identical) reservoirs: one reservoir processes the frames from left-to-right while the other processes them from right-to-left (see Figure 7.1). The readouts at time t are then computed as concatenation of the two reservoir states reached after having processed input vector U_t.

7.3 An RCN-Based Denoising Auto-Encoder

By artificially adding realistic noise to a clean speech file and by filtering the clean speech before or after adding the noise, one can create 'noisy' samples and measure the impact of the distortions on the feature vectors by comparing the noisy with the clean speech feature vectors. It is also possible then to teach a neural network-based denoising auto-encoder (DAE) [155] to reconstruct the clean feature vectors from the noisy feature vectors. We argue that a complex non-linear dynamical system with memory such as an RCN should be able to do a good job in feature denoising. To limit the complexity of the overall system, the developed RCN-DAEs incorporate just one unidirectional reservoir.

Since we use MFCCs (Mel-scale Frequency Cepstral Coefficients) as the feature vectors, and since these MFCCs emerge from a multi-stage process (See Figure 7.2), the first question is at which position in this processing scheme, the insertion of a DAE would be the most effective. Traditionally, speech enhancement is often acting upon the Discrete-time Fourier Transform (DFT) [156, 157] (at position P1 in Figure 7.2). On the other hand, denoising in the log Mel-frequency domain [158] (at position P2) or the MFCC domain [159, 160] (at position P3) are more appealing due to the much lower dimensionality. A high dimensionality normally results in a higher computational cost [161]. In the case of an RCN-DAE however, this is not true since each reservoir neuron is only simulated by a few inputs. Therefore, we can easily consider an RCN-DAE at all positions.

A second question that needs to be answered is whether it is beneficial to denoise the static as well as the dynamic input features, or it suffices to denoise the static features alone and to derive the denoised dynamic features thereof.

7.4 Experimental Setup

In this section, we present the experimental framework that was adopted to investigate the potential of the different system configurations presented in the previous sections.

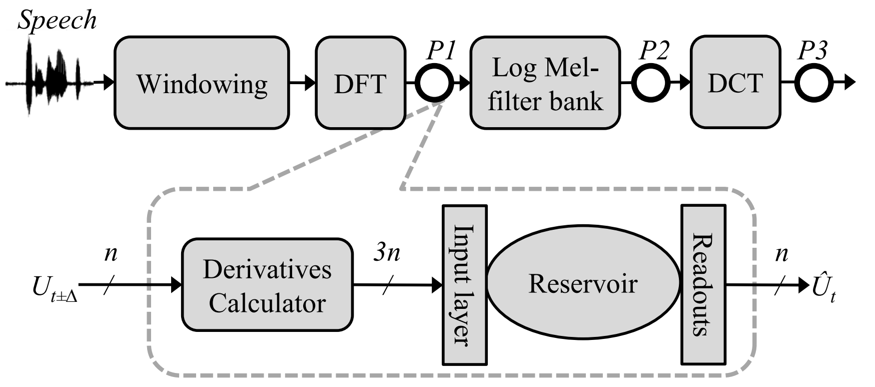


Figure 7.2: Possible points to apply the denoising system (top) and their structure in more details (down)

7.4.1 Speech Corpus: Aurora-2

The Aurora-2 corpus consists of clean and noise corrupted digit sequences counting 1 to 7 digits per utterance. Each utterance is passed through a G712 or a MIRS filter, and then sampled at 8 kHz [29]. Since there are two variants of '0' in American English, namely *zero* and *oh*, the vocabulary is composed of 11 digits.

The data is divided into a training part and an evaluation part. The framework supports two types of experiments: clean training experiments in which systems are developed on 8440 clean training utterances from 110 adults and multi-style training experiments in which systems are developed on 8440 noise corrupted versions of the same utterances. The corruption is randomly chosen out of four noise types and five SNRs (∞ (clean), 20, 15, 10 and 5 dB). The evaluation utterances come from speakers that are not present in the training data and they are divided into three tests. Tests A and B each contain 28,028 utterances covering 4004 different digit sequences, 4 noise types and 7 SNRs (∞ (clean), 20, 15, 10, 5, 0, and -5 dB). The noise types occurring in Test B do not occur in the multi-style training data, while those of Test A do. Test C contains 14,014 utterances covering 2002 different digit sequences, 2 noise types (one matched and one mismatched) and 7 SNRs. Unlike all other utterances they passed through a MIRS instead of a G712 filter (see Table 7.1).

	Train & Test A	Test B	Test C
Noise types	N1: subway N2: babble N3: car noise N4: exhibition hall	N1: restaurant N2: street N3: airport N4: train station	N1: subway N2: street
Filter	G712	G712	MIRS

Table 7.1: Noise types and filters used in Aurora-2 dataset

7.4.2 Evaluation Results

We report Word Error Rates (WERs) on tests A-C and we consider both clean speech training (only clean speech used during training) and multi-style training (both clean and noisy speech used during training).

In the final evaluation phase, both the acoustic models and the DAE are trained on the complete training set, using the control parameters that were found optimal in a development phase during which two thirds of the training set are used for training and the remaining third for control parameter optimization (e.g., the Viterbi decoder penalty, P_0 of the recognizer). In this paper, we only report the results of the final evaluation phase for each experiment.

7.4.3 Reference Systems

In order to set a reference, we first report some state-of-the-art system performances (see Table 7.2). In particular, we consider the ML-based GMM systems using AFE-features proposed in [153], the ML-based and MCE-based GMM systems proposed in [113] and [26], two GMM systems embedding more sophisticated back-ends based on joint uncertainty decoding (JUD) and Vector Tylor Series (VTS) respectively [124], a GMM system that utilizes SPLICE to enhance the features [28], and the tandem system embedding deep belief networks and GMM (T-DBN-GMM), reported in [142]. The figures show that advanced back-end techniques (JUD and VTS) lead to a larger gain in noise robustness than advanced front-end techniques, but it is not clear from the papers how much degradation they induce for clean speech recognition.

7.4.4 Front-End Setups

We will investigate three different acoustic feature sets: MFCCs (log energy and $c_1 \ldots c_{12}$), Mel filterbank features (MelFB) (24 log energies),

System	Clean			Multi		
	Clean	0-20	-5dB	Clean	0-20	-5dB
GMM (AFE) [153]	0.77	13.2	69.9	0.83	8.4	59.2
GMM (MFCC) [113]	0.84	19.7	82.2	1.77	8.5	59.1
GMM (SPLICE) [28]	0.55	17.6	83.7	-	12.7	-
GMM (MFCC-MCE) [26]	0.41	15.7	77.2	0.92	6.4	55.3
GMM (VTS) [124]	-	9.4	-	-	-	-
GMM (JUD) [124]	-	10.3	-	-	-	-
T-DBN-GMM [142]	1.26	21.0	74.6	-	-	-
RCN (MelFB)	0.74	11.0	63.8	1.06	**5.4**	45.0
RCN (MFCC)	0.93	10.7	59.7	1.46	6.2	46.5
RCN (AFE)	0.84	**8.9**	**54.4**	1.40	5.7	**43.2**

Table 7.2: Comparing average WERs (in %) per condition for test sets A-C of Aurora-2 using a 3-layer hybrid RCN-HMM for both clean and multi-style training.

and the AFE features (denoised $c_0 \dots c_{12}$ without dropping non-speech frames) [131]. In all cases, the analysis is performed on 30 ms Hamming-windowed frames and the hop size between frames is $\tau_{fr} = 10$ ms. Each feature set is supplemented with Δ and Δ^2 features.

Before supplying the feature vectors to the ASR, an utterance-wise normalization that creates zero-mean and unit-variance inputs per feature is performed.

7.4.5 RCN-HMM Hybrid Setup

Following our previous work, the reservoir control parameters of each reservoir are determined in the same way. Defining $\tau_\lambda \doteq -\tau_{fr}/\ln(1-\lambda)$ as the leaky integration time constant and T as the expected state duration, we select $(\rho, \tau_\lambda, K^{in}, K^{rec}) = (0.8, T, 10, 10)$ and chose α_U so that the average variance of the reservoir outputs reaches a certain level [3]. In this work we utilize a 3-layer RCN-HMM system with 8K nodes per layer, and since layers 2 and 3 see basically the same inputs, α_U is taken the same for both layers.

Note that the size of the reservoir – defined as the number of neurons it contains (N^{res}) – is considered to be an independent variable. Moreover, since K^{in} and K^{rec} are kept fixed to 10, the CPU-time needed for calculating the readouts scales linearly with the size of the reservoir.

The reservoir networks are trained by means of a Tikhonov regression [144] and each digit is modeled by a 7-state left-to-right HMM whilst the silence is modeled by a single state. The target outputs encode the visited HMM state at time t.

7.4.6 RCN-Based DAE Setup

In line with [162], we contemplate that ρ and λ are the only task-dependent control parameters of a reservoir. Consequently, even though denoising is a completely different task than digit state recognition, we keep $K^{in} = K^{rec} = 10$ and we maintain the same method as before for computing α_U. Furthermore, since leaky integration is a form of smoothing corresponding to some low-pass filters, we argue that no leaky integration should be applied here as the spectrum of the denoised inputs is not expected to have a smaller bandwidth than that of the noisy inputs. Consequently, there is only one parameter left to optimize, namely ρ.

In all experiments, irrespective of the chosen feature set, we have used a 2-layer RCN with a unidirectional reservoir of 1K neurons per layer. The target outputs are obviously the envisioned clean feature vectors.

7.5 Experimental Results

In this section, we review the experiments we conducted to assess the capacity of an RCN to denoise the features and the capacity it has to further raise the robustness of the CDR system.

7.5.1 Denoising Capacity of an RCN

For the assessment of the denoising capacity of the RCN, we compute the correlation between the denoised and the clean MFCC features that will be supplied to the ASR. I.e., we consider all 39 features here. Note that since the RCN-based ASR always works with utterance-wise mean and variance normalized features, the correlations are computed after this normalization step as well.

We conducted two types of experiments: one in which the test samples come from Test A and represent conditions that are present during training (i.e., ‘Seen’) and one in which the test samples come from Test C and represent another channel and unseen SNRs (i.e., ‘Unseen’).

In a first phase, we denoise only the static features and investigate the effectiveness of the DAE imputed at each of the three positions P1, P2 and P3 (see Section 7.3). The results are depicted in Figure 7.3.“Ref” denotes the correlation existing between the noisy and the clean features. The data

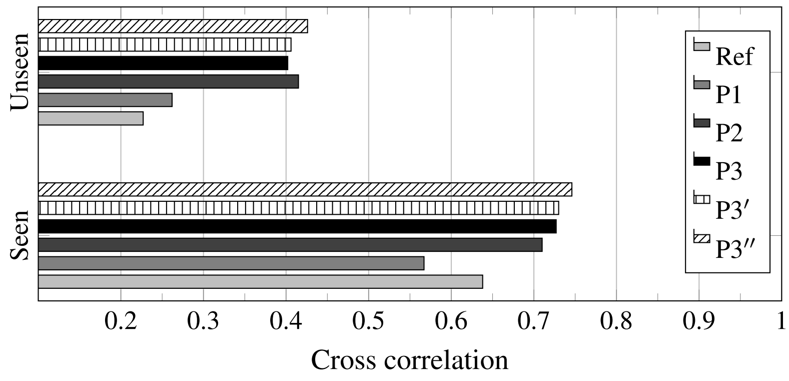


Figure 7.3: The correlation between the output of the DAE and the clean normalized features, by denoising the data stream in different points.

show that denoising in the DFT-domain is not working well but denoising in the two other domains does. In fact, denoising the MelFB and the MFCC features seem to be equally effective both in ‘Seen’ and ‘Unseen’ conditions. We will, therefore, evaluate them both in combination with CDR.

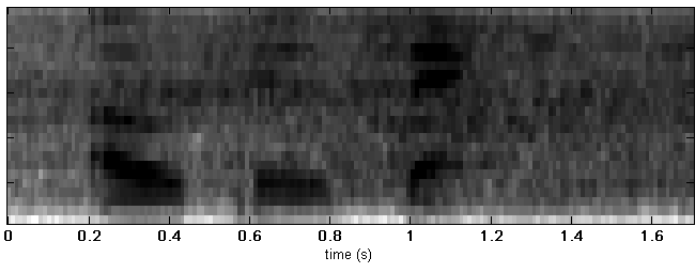
In a second phase, we constructed a DAE that was trained to denoise the dynamic as well as the static MFCCs instead of just the static features. This approach leads to the correlations marked by P3′. There seems to be no benefit in working this way: the computational load increases and the accuracy is not any better.

Finally, we wanted to assess the limits of the RCN-based DAE by raising the reservoir size from 1K to 4K neurons. The results obtained with the larger reservoirs (marked as P3″) are only slightly better than the ones obtained with the smaller reservoirs.

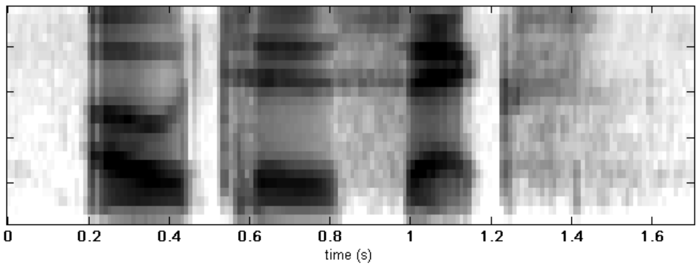
In order to illustrate the effect of the DAE, we have depicted on Figure 7.4 the MelFB spectrograms for a noisy speech sample (SNR = 5dB) before and after denoising, together with the clean speech spectrogram. It is especially noteworthy that the DAE does an excellent job in the silence parts. This is partly due to the large number of non-speech (silent) frames in the training data.

7.5.2 Need for a Denoising Front-End in an RCN-Based CDR

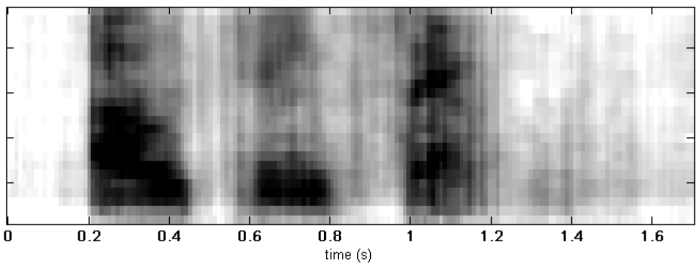
In a first recognition experiment we test the three acoustic feature sets described in the previous section in combination with a three-layer bi-directional RCN-HMM hybrid with 8K neurons per layer. The results of



(a) Noisy MelFB features



(b) Clean copy



(c) Denoised MelFB

Figure 7.4: Denoising MelFB features of a sample with street noise of SNR 5dB from Test C.

this experiment are listed in the bottom section of Table 7.2.

Apparently, in the clean speech training experiment there is only a little difference between the feature sets in the matched condition (clean). In the mismatched conditions, the AFE does lead to more robustness, but the gain is much less impressive than it is for GMM-based systems. This finding seems to confirm the hypothesis that a reservoir can filter out a large part of the noise without ever having been confronted with noise during training.

In the multi-style training experiment, where the mismatch between the

Configuration	Features	Clean			Multi		
		Clean	0-20	-5dB	Clean	0-20	-5dB
NoDAE + Baseline	MFCC	0.93	10.7	59.8	1.46	6.2	46.5
	AFE	0.84	8.9	54.4	1.40	5.7	43.3
DAE + Baseline	MFCC	2.08	8.9	53.4	3.35	8.0	50.0
	AFE	2.56	9.4	49.7	3.57	8.1	45.9
DAE + Retrained	MFCC	1.36	8.6	54.5	1.77	6.6	48.4
	AFE	1.38	7.7	50.0	1.85	6.2	43.6

Table 7.3: Comparing average WERs (in %) on Test sets A-C per configuration: (1) NoDAE + Baseline means no DAE and a baseline RCN-HMM recognizer, (2) DAE + Baseline means that the features are denoised but the denoised features are used as inputs to the baseline recognizer, and (3) DAE + Retrained means that the features are denoised and that the recognizer was retrained on denoised features.

test and the training remains much smaller, the positive effect of the AFE is much smaller, and is only significant at the SNR of -5 dB. Nevertheless, the data suggest that if the DAE could do a better job than the AFE, it could lead to a more noise robust CDR, as well.

7.5.3 Can RCN-Based DAE Outperform the AFE?

In order to answer the question, we conducted experiments with a front-end incorporating an RCN-based DAE at position P3. We considered three configurations: (1) **NoDAE + baseline**: MFCC or AFE features are used in combination with the baseline acoustic models (2) **DAE + baseline**: MFCC or AFE features are denoised by a DAE and used in combination with the baseline acoustic models (3) **DAE + retrained**: MFCC or AFE features are denoised by a DAE and used in combination with acoustic models that are retrained on the denoised features. In the case of retraining, we maintained the distinction between clean speech training and multi-style training, but of course, the clean speech training results have to be interpreted with care now, because after all, the DAE has already seen the noisy training examples that are present in the multi-style training corpus.

The results listed in Table 7.3 show that the RCN-based denoiser (DAE + retrained + MFCC) does not outperform the AFE as a denoiser (NoDAE + baseline + AFE). The DAE seems to cause a drop in the clean speech results compared to the AFE, but for the rest it leads to basically the same noise robustness as the AFE.

The last line of the table shows however that a combination of the two denoising approaches (AFE and DAE) does yield an extra gain in strongly mismatched conditions (in particular the combination of clean speech training and noisy test samples), but this gain comes at the expense of a significant loss in the performance for matched conditions (clean test samples). In the multi-style training experiment, the mismatch between training and test conditions is never more than moderate, and consequently, the AFE features do not benefit from an extra denoising by a DAE.

As could be expected, the DAE + Baseline configuration is characterized by a strong detrimental effect in matched conditions because the features used during training and test have become different even in these conditions.

In a control experiment, we also trained and tested the RCN-based DAE by inserting it at position 2 in the MFCC computation chain. In line with the correlations measured before, the emerging systems achieve very similar results showing no preference for positions P2 and P3.

7.6 Conclusion and Future Work

Recently, we were able to show that reservoir computing is a viable acoustic modeling technique for trying to attain more robust automatic speech recognition. Experiments on the Aurora-2 benchmark demonstrated that our RCN-based systems already outperform the most complex GMM-HMM based systems on the task of continuous digit recognition.

The main objectives of the present paper were: (1) to establish whether reservoir networks can be trained to denoise the feature vectors, (2) to investigate how much benefit can be attained by applying denoising in the front-end of the RCN-recognizer, (3) to establish whether RCN-based denoising can outperform traditional signal processing based techniques (e.g., like in the ETSI advanced front-end (AFE)), and (4) to find out whether the two denoising techniques are maybe complementary.

First of all, we could show that an RCN-based Denoising Auto-Encoder (DAE) imputed at a suitable position in the MFCC front-end can lead to increasing correlations between the features of the noisy and the clean speech utterances.

Next, we could demonstrate that feature denoising in the front-end of an RCN-based ASR is beneficial, although not to the same extent as with traditional GMM-based ASR systems. A somewhat disappointing result is that the improvements generated by an RCN-based DAE are smaller than those generated by the AFE which is based on well established signal processing algorithms for removing noise. However, by combining the two techniques, a small gain is demonstrated in severely mismatched conditions, but this extra gain comes at the expense of a significant degradation in the matched condition.

In the near future, we will compare the RCN-based DAE with the AFE in situations where the noise is more non-stationary. Furthermore, we want to investigate whether the principles forming the basis of the uncertainty decoding in GMM-HMM systems can be transferred to an RCN-HMM system and whether this could lead to further improvements of the noise robustness. Finally, we would also like to extend our research to noise robust large vocabulary speech recognition (e.g., Aurora-4).

8 Handwritten Digit Recognition Using RCNs

This chapter is an edited version of the following original publication:

[6] A. Jalalvand,W. De Neve, R. Van de Walle, and J.-P. Martens, "Noise robust handwritten digit recognition with reservoir computing networks," *Journal of Machine Learning Research*, submitted, 2014

8.0 Preface

Reservoir Computing Networks (RCNs) have already been used in different areas of artificial intelligence such as robotics and brain machine interfacing. In this chapter, I demonstrate that (1) the tuning strategy that was conceived for speech processing is also applicable to image processing, (2) RCN-based systems can offer state-of-the-art handwritten digit recognition, both in the absence and in the presence of noise and (3) RCN-based systems can denoise images and achieve good noise robust recognition by supplying these images to a recognizer that was solely trained on clean images. My comparative experiments demonstrate that the proposed RCN-based handwritten digit recognizer achieves an error rate of 0.81% on the clean test data of the well-known MNIST benchmark and that the proposed RCN-based denoiser can reduce the error rate on the noisy test data from 37% to 2%.

8.1 Introduction

Nowadays, touchscreen devices such as smart-phones and tablets have become more popular than ever. Therefore, many companies plan to replace traditional input modes like keyboard and mouse with more elegant modes such as speech and handwriting. Due to the different handwriting styles, there is a lot of variability in the images of the same character, making automatic handwriting recognition (HWR) a challenging task.

The presence of background noise is another source of variability the HWR system may have to deal with. In fact, in applications such as address recognition on parcels or full text recognition from digital scans of old manuscripts or typed documents, noise corrupted images such as the one depicted in Figure 8.1 are the norm. None of the available optical

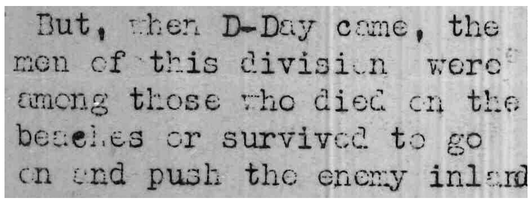

But, when D-Day came, the
men of this division were
among those who died on the
beaches or survived to go
on and push the enemy inland

Figure 8.1: Part of the military newspaper "The Stars and Stripes" published in 1944.

character recognition (OCR) systems seems capable of producing a reliable transcription of the text from such an image.

The aim of the present paper is to show that reservoir computing networks (RCNs) have great potential for achieving good performance in HWR from noise corrupted images. We demonstrate this on the MNIST [163] dataset, a handwritten digit recognition task (HDR) used by many research groups to benchmark their technologies. HDR benchmarks such as the well-established MNIST task not only facilitate the comparison of new technologies to the state-of-the-art, HDR on itself is also highly relevant as it is an indispensable component of any application involving the processing of handwritten numerical data such as telephone numbers, PIN-codes, account numbers and coordinates.

More than two decades ago, Multilayer Perceptrons, MLPs [164] were among the first classifiers that were tested on MNIST. In [163], an MLP with two computational layers of neurons was reported to reach a digit error rate (DER) of 2.95%, and a later study [165] reported a DER of 1.60%. Employing MLPs with more layers was long time believed to yield no significant improvement. However, new training methods, permitting a better exploitation of multiple hidden layers, were recently discovered and gave

System	DER% (Original training set)	DER% (Enriched training set)
2-layer MLP	2.95 [163]	2.45 [163]
2-layer MLP	1.60 [165]	0.70 [165]
DBN	1.03 [166]	-
DBM	0.95 [167]	-
CNN	0.95 [163]	0.80 [163]
DCN	0.83 [168]	0.35 [169]
Large CNN	0.60 [170]	0.39 [170]
Multi-CNN	-	0.23 [171]

Table 8.1: Reference results on MNIST using the original training set and using an expanded version of the training set (for example, by applying deformation). The presented DERs are accompanied by a reference to the paper introducing the technique that was used.

rise to the emergence of Deep Neural Networks (DNNs). Differences in the details lead to DNNs of various types. Two of them, Deep Belief Networks (DBNs) and Deep Boltzmann Machines (DBMs) have also been tested on MNIST (see Table 8.1). Roughly speaking, they achieve a DER of about 1%.

Note, however, that long before deep neural networks became successful, significant improvement over a standard 2-layer MLP was achieved by means of a Convolutional Neural Network (CNN) [163] that acts like a feature extractor. In fact, one of the main points of criticism raised against an MLP was that its hidden neurons see the whole image and are therefore bound to overlook the undoubtedly present local topological relations between adjacent pixels in sub-regions of the image [163]. Therefore, the idea was to scan the image, to filter the pixels appearing in the emerging sub-regions by means of trainable filters and to down-sample the filtered outputs so as to create a rich and compact feature representation that constitutes a more suitable input to the MLP-based classifier. The first results obtained with the CNN approach were already mentioned in [163]. With a DER of 0.95%, CNN-based systems can still be considered as the state-of-the-art for HDR.

Obviously, there is no reason why the concepts of a CNN and a DNN could not be combined. Deep Convolutional Neural Networks are the exponent of that idea, leading to a DER of 0.83%.

Another idea that induced a significant boost in HDR, was the idea of enriching the original training dataset with new images, obtained by deforming the raw training images. In [165], for instance, elastic deformations were applied to the raw images achieving a convincing drop in DER from 1.6% to 0.7%. Since then, basically all novel methods have shown to benefit from such an enrichment of the training set (see right column of Table 8.1). By also introducing separate DNNs for different digit widths (6 classes), it was even possible to achieve human-competitive performance (0.23%) [171].

In spite of the spectacular performances achieved in clean conditions, all aforementioned approaches fail dramatically when recognizing digits from noisy samples. In [166], for instance, it was shown that a DBN trained on clean samples, fails completely when recognizing noisy samples. The DER raises to 33.8% when the digits are partially masked by square blocks and to 66.1% when the digits are surrounded by a black border (see Figure 8.2). Consequently, new research has been directed towards improving the robustness of HDR against the presence of noise. In general, one can roughly

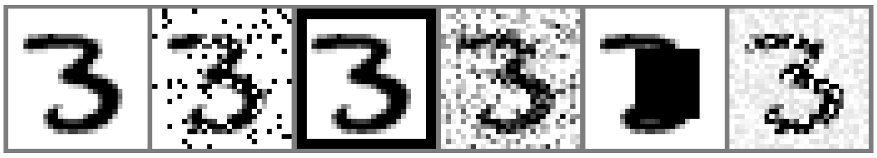


Figure 8.2: From left, a clean MNIST sample and its corresponding noisy versions which are salt & pepper, border, Gaussian, block, and speckle, respectively.

distinguish three approaches: (1) add noise to the training examples and perform a so-called *multi-style training* of the neural network, (2) make the classifier intrinsically more robust against the effects induced by noise, for example, by using a sparsely connected DBN rather than a conventional densely connected one [166] and (3) remove a large part of the noise from the input image before presenting it to the classifier. In [166], it was argued that due to the noise, a lot of neurons are driven into saturation and are therefore not contributing to the recognition anymore. By training it on noisy images, the standard DBN could be made much more effective. The DER could be reduced from 33.8% to 8.7% for the case of block noise and from 66.1% to 1.9% for the case of border noise.

In another study [172, 173], a stacked sparse DBN-based denoising auto-encoder (SSDA) is trained to denoise the images. In such a system, one SSDA per noise type was trained and the denoised image is obtained as a linear combination of the individual SSDA outputs. Feeding these images to a DBN trained on clean samples induced a dramatic improvement. The average error rate was reduced from 34.3% (an average over five noise types) to 2.4%. Examples of the noise types are depicted in Figure 8.2 and

Table 8.4 (Section 8.5.3) lists the improvements per noise type. As the combination weights are determined by a weight prediction module, the latter system was called an adaptive multi-column SSDA (AMC-SSDA) system.

It is now generally acknowledged that conventional DNNs perform well, but they are still hard to train (it takes a lot of time and the hyperparameters of the training process must be set properly). We, therefore, investigated whether reservoir computing networks (RCNs) [80, 98, 174] can offer an elegant alternative.

In [3], we conceived a method for making the design of an RCN as straight forward as possible: one only needs to specify what the bandwidths of the RCN inputs and outputs are and the method automatically produces good values for all the hyperparameters of the RCN. Obviously, the size of the reservoir remains a free parameter whose optimal value depends on the number of available training examples and the envisioned compromise between accuracy and computational cost. The conceived method sped up the design tremendously. The same procedure is used in this paper, proving experimentally that it can also be applied with success to other tasks such as handwriting recognition.

The rest of this paper is organized as follows. First, it briefly recalls the general principles underlying RCNs (Section 8.2) and it proposes RCN architectures for performing HDR (Section 8.3). Then, it describes an experimental study of these architectures for the recognition of clean handwritten digits (Sections 8.4 and 8.5). In the second part of the paper, the focus is on the noise-robustness of the RCN architectures (Section 8.6). The paper ends with some conclusions and some ideas for future work.

8.2 Reservoir Computing Network (RCN)

In its simplest form, an RCN is a neural network with two particular computational layers: (1) a hidden layer of recurrently interconnected non-linear neurons, driven by inputs and by delayed feed-backs of its outputs and (2) an output layer of linear neurons, driven by the hidden neuron outputs (Figure 8.3). A fundamental point is that the input weights and the recurrent connection weights are fixed after being initialized randomly, and that only the output weights are optimized (trained) for solving the targeted problem.

The recurrently interconnected hidden neurons constitute a *reservoir* (a pool) of computational neurons. The reservoir can be viewed as a nonlinear dynamical system that analyzes a stream of inputs, obtained, for instance, by scanning an image from left to right. The outputs are called *readouts* [99] so as to differentiate them unambiguously from the reservoir

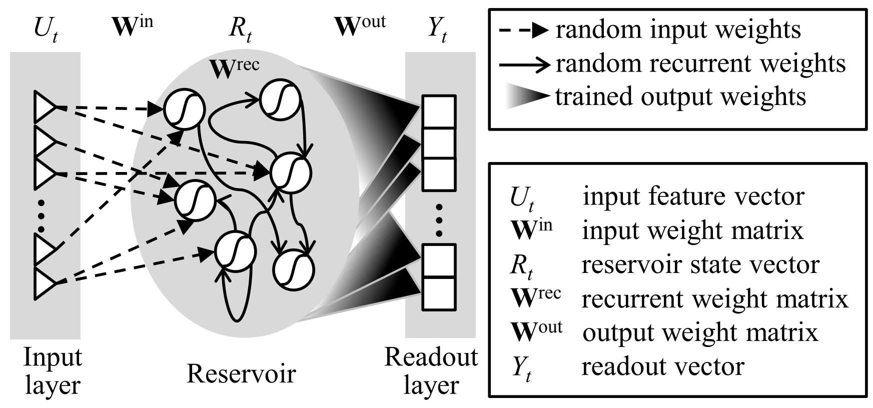

U_t
$\mathbf{W}^{\text{in}}$
R_t
$\mathbf{W}^{\text{out}}$
Y_t
$\mathbf{W}^{\text{rec}}$
Input layer
Reservoir
Readout layer
random input weights
random recurrent weights
trained output weights
U_t input feature vector
$\mathbf{W}^{\text{in}}$ input weight matrix
R_t reservoir state vector
$\mathbf{W}^{\text{rec}}$ recurrent weight matrix
$\mathbf{W}^{\text{out}}$ output weight matrix
Y_t readout vector

Figure 8.3: A basic RCN consists of a reservoir and a readout layer. The reservoir is composed of interconnected **non-linear** neurons with random weights. The readout layer consists of **linear** neurons with **trained** weights.

outputs. If U_t, R_t and Y_t represent the reservoir inputs, the reservoir outputs and the readouts at time t, respectively, the RCN equations can be written as follows [99]:

$$R_t = (1-\lambda)R_{t-1} + \lambda\, f_{res}(\mathbf{W}^{in}U_t + \mathbf{W}^{rec}R_{t-1}) \tag{8.1}$$

$$Y_t = \mathbf{W}^{out}R_t$$

with λ being a constant between 0 and 1, with f_{res} being the non-linear activation function of the reservoir neurons (we used *hyperbolic tangent* in this work) and with $\mathbf{W}^{in}$, $\mathbf{W}^{rec}$ and $\mathbf{W}^{out}$ being the input, recurrent and output weight matrices, respectively. The constant λ is called the leak rate because (if one makes abstraction of f_{res}) Equation (8.1) represents a leaky integration of the neuron activation.

Each individual input is normalized so that it has a zero mean and unit variance over the training examples. The weights of the hidden neurons are fixed by means of a random process that is characterized by four parameters [3, 162]: (1) α_U, the maximal absolute eigenvalue of the input weight matrix $\mathbf{W}^{in}$, (2) ρ, the maximal absolute eigenvalue of the recurrent weight matrix $\mathbf{W}^{rec}$, (3) K^{in}, the number of inputs driving each reservoir neuron and (4) K^{rec}, the number of delayed reservoir outputs driving each reservoir neuron. The first two parameters control the absolute and the relative importance of the inputs and the delayed reservoir outputs in the reservoir neuron activation. The latter two control the sparsity of the input and the recurrent weight matrices. Any effective reservoir should at least have the so-called echo state property, stating that with time, the reservoir should forget the initial state it was in. That is also why a reservoir network was originally called an Echo State Network [99]. It was shown in [99] that the echo state property holds if ρ, also called the spectral radius of the recurrent weight matrix, is smaller than 1.

The output weights are determined so that they minimize the mean squared error between the *computed* readouts Y_t and the *desired* readouts D_t over the training examples [2]. If an RCN is trained for recognition, the desired output D_t is a unit vector with one non-zero entry encoding the desired class at time t. If it is trained for feature denoising, D_t is the desired clean feature vector at time t. In both cases, the output weights emerge as the solution of an over-determined set of linear equations.

An RCN can be considered as an extension of the Extreme Learning Machine (ELM) proposed in [101]. An ELM is a two-layer MLP with a randomly fixed hidden layer of non-linear neurons followed by an output layer of linear neurons whose weights are determined so as to minimize the mean squared difference between the computed and the desired outputs. According to [101, 104], a key property of the ELM is its ability to generalize to unseen data. We argue that the recurrent connections inside the RCN can further increase the accuracy and the generalization ability of the ELM, an argument that was already experimentally confirmed by speech recognition experiments [3].

8.3 RCN-Based Architectures for Image Processing

In many neural network-based HDR systems, the input is a pixel array representing the whole digit image [173, 175]. However, in order to exploit the dynamic properties of an RCN, we need to create a sequential input stream. This can be achieved by scanning the image in a similar way as in a CNN.

8.3.1 Readouts

The outputs of each RCN that will be encompassed in the recognizer correspond to the 10 digits and to the white space which is present in each digit image. By introducing this white space and by envisioning an image as a digit surrounded by white space, we can achieve that the digit readouts will mainly react to features that are typical for the digit they represent.

8.3.2 Basic Architecture

A trivial procedure leading to the desired input stream is horizontal scanning: the image is scanned column-wise from left to right and the subsequent columns (called frames) form the input vector sequence (see Figure 8.4). The score of a certain digit can then be obtained by accumulating the corresponding RCN outputs (readouts) over time.

During our speech recognition research [3,9], we learned that bi-directional processing and deep RCNs improve the accuracy of the system. Bi-directional processing means that each RCN contains two reservoirs: one that processes the data in the normal order and an identical one that processes them in the reversed order (see Figure 8.4). Although these two reservoirs are identical, they yield different outputs at a particular time t as they rely on a totally different history to create the outputs.

A deep RCN is obtained by stacking multiple RCNs, as depicted in Figure 8.4. Each layer of the deep RCN is a basic bi-directional RCN. The layers are trained one after the other using the same desired outputs in every layer. The argument for cascading layers is that a new layer can correct some of the mistakes made by the preceding layers because it offers additional temporal modeling capacity and a new inner space in which to perform the classification.

8.3.3 More Complex Architectures

Horizontal image scanning seems to be the obvious choice since it is also suitable for continuous HWR. However, for isolated digit recognition one can consider vertical scanning as well. Moreover, one can imagine various ways of combining the two scanning directions in a single system. The ones we propose here are depicted in Figure 8.5:

Combination of input features

A simple combination strategy is to supply the RCN with the concatenation of one row and one column at each time step (see Figure 8.5(a)). Obviously, this approach presumes a square image leading to an identical number of frames per scanning direction.

Weighted sum of scores

Another strategy is to make two independent systems working in parallel: one (H-RCN) using horizontal and one (V-RCN) using vertical scanning. The digit scores can then be obtained as a linear combination of the scores

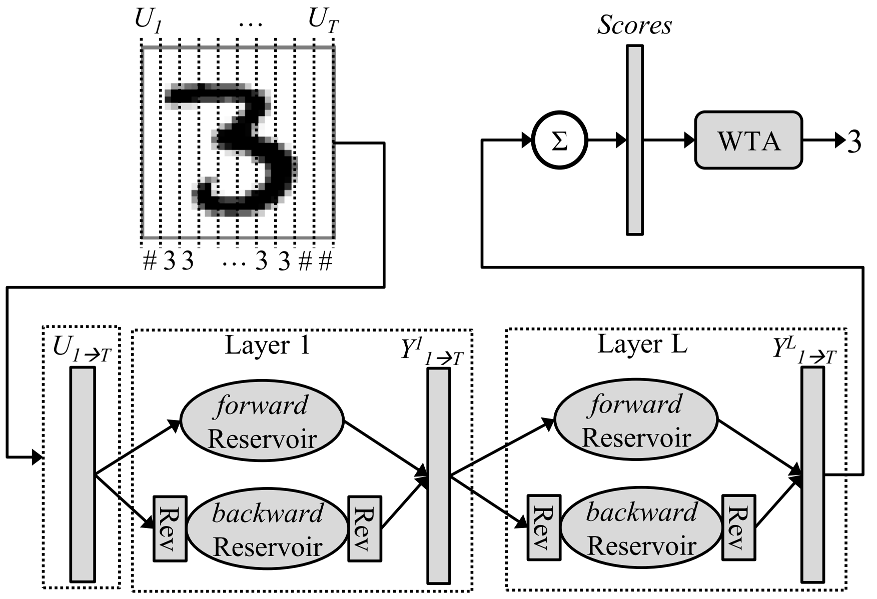


Figure 8.4: Architecture of a deep RCN comprising bi-directional processing in each layer. The forward reservoir processes the inputs $U_{1\to T}$ whereas the backward reservoir processes the inputs $U_{T\to 1}$. The outputs of the latter reservoir are then time reversed before combining them with the outputs of the direct reservoir. The digit scores are obtained by accumulating (the Σ component) the RCN outputs across time.

emerging from the two sub-systems (see Figure 8.5(b)). The advantage of this approach is that it can also be applied to rectangular images.

Combination of readouts

The third option is to supply the combined readouts of the V-RCN and the H-RCN to the final digit recognition RCN (see Figure 8.5(c)). Obviously, this approach again presumes a square image.

8.4 Experimental Setup

In this section, we present the experimental framework that was set-up in order to assess the potential of the proposed system configurations.

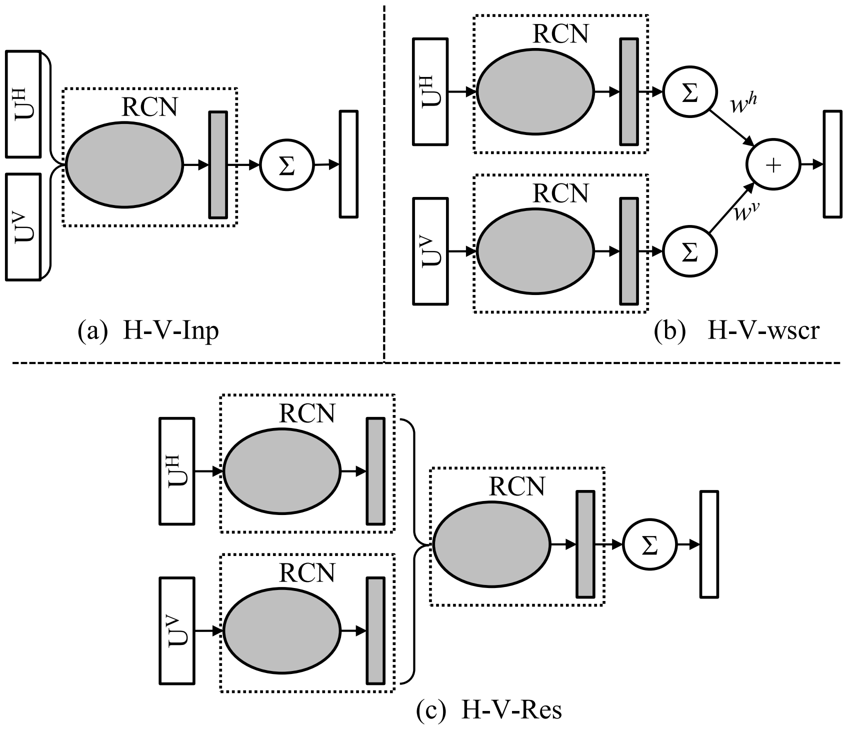


Figure 8.5: Different ways of combining horizontal (H) and vertical scanning (V) in a single system: (a) supply the RCN with one row and one column of the image, (b) compute a weighted sum of the digit scores (accumulations over time) emerging from an H-RCN and a V-RCN and (c) supply the H-RCN and V-RCN outputs to another RCN and accumulate the scores of the readouts of this RCN.

8.4.1 MNIST Corpus

The MNIST corpus [163] consists of clean handwritten isolated digit samples (0 to 9), grouped into two datasets: one that can be used for system development (60,000 images) and one that can be used for system evaluation (10,000 images). Each digit is represented by a 28×28 gray-scale encoded pixel array. The original pixel codes (between 0 and 255) are interpreted as real numbers between 0 and 1. Many studies sub-divide the development set into a training set of 50,000 images and a validation set of 10,000 images.

Some studies extend the training dataset by deforming the original training images and by considering the deformed images as extra training examples, but here we refrain from doing so because our main objective is to show that an RCN-based system has potential to become an alternative to other state-of-the-art systems and it is difficult to make a fair comparison of results obtained with an extended training set without knowing exactly which deformations were applied in the systems one wants to compare with.

In order to conduct experiments on noise robustness, we construct a multi-style dataset by dividing the dataset into six equally large parts. One part is left unaltered and serves as a clean dataset. The images of the other five parts are corrupted with noise, one noise type per part. The considered noise types are Gaussian noise, Salt & Pepper noise, Speckle noise, Block noise and Border noise, as in [166, 173]. A clean sample and five noisy versions created thereof are depicted in Figure 8.2. The division in five noise types and clean runs orthogonal to the division in train, development and test set, which means that the multi-style sets contain the same images, only now with noise added.

8.4.2 System Development and Evaluation

During system development, 5/6 of the development data is used for system training and 1/6 for system validation. The aim of the validation is to find good values for the hyperparameters such as the size of a reservoir and the number of layers. Once the hyperparameters are fixed, a final training is conducted to create the final system.

We distinguish between clean training, which is training on the clean training set, and multi-style training, which is training on the multi-style training set. We employ the DER (digit error rate) on the test set as the recognition performance measure.

8.4.3 Front-End

The front-end scans the image either horizontally (H) or vertically (V) and per scanning step t, the column vector (if H) or the row vector (if V) is a 28-dimensional vector X_t. However, it is common in neural networks to obtain U_t by extending X_t with its first and second derivatives in the scanning direction, or by stacking the vectors $X_{t-k}, .., X_{t+k}$. Both approaches have the advantage of providing the system with a glimpse of the future. In our experiments, we use frame stacking with $k = 2$.

8.4.4 RCN Hyperparameters

The creation of a suitable RCN was studied in detail in [3]. In summary, the theory presented there leads to the following conclusions:

1. The input and recurrent weight matrices ($\mathbf{W}^{in}$ and $\mathbf{W}^{rec}$) can be very sparse. In particular, 5 to 10 connections (non-zero weights) per node are enough, regardless of the reservoir size and the input feature vector size.

2. The spectral radius, ρ, must be tuned to the bandwidth F (normalized frequency) of the input activations of the reservoir (interpreted as time series):
$$\tau_\rho = \frac{-1}{\ln(\rho)} = \frac{0.35}{F} \tag{8.2}$$

3. The leak rate λ must be tuned to $T_{\min}$, the minimum time (in scan steps) the reservoir output is expected to remain constant:
$$\tau_\lambda = \frac{-1}{\ln(1-\lambda)} = T_{\min} \tag{8.3}$$

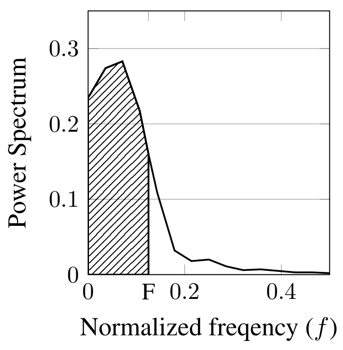
4. The square of α_U follows from a function of the other parameters, a function that is proportional to an auxiliary parameter V_{opt}, defined as the optimal reservoir output variance.

In [3], we argue that V_{opt} may be independent of the task, but since the objective of that work was only speech recognition, we did not present any experimental evidence for this argument yet. Here, we will show that $V_{opt} = 0.035$ which was found optimal for spoken digit recognition, is valid for handwritten digit recognition, as well.

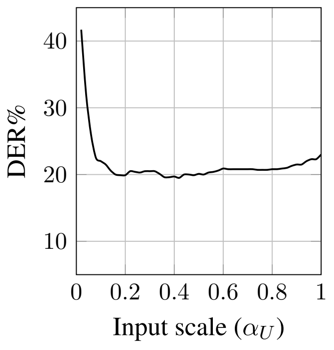
Design of reservoir in first layer

The first step consists of determining the bandwidth of the input activations. In order to do so, it suffices to consider an arbitrary small reservoir (500 nodes) with memory-less neurons (no leaky integration) and no recurrent connection, to record a few hundred input activation patterns of 28 samples long (which is the number of scan steps), to determine the periodogram (the square of the magnitude of the DFT) of each recorded pattern and to calculate the envisioned power spectrum as the mean of these periodograms. To facilitate our experiment, we fed the reservoir with normalized pixels with a unit variance.

Figure 8.6(a) shows the estimated power spectrum, $|B(f)|^2$, and its bandwidth, $F = 0.15$. For this value it follows from Equation 8.2 that $\rho = exp(-0.15/0.35) = 0.65$. Note that there is little difference in bandwidth between horizontal and vertical scanning.



(a) Power spectrum of the input activations



(b) Optimizing α_U

Figure 8.6: Power spectrum of the input activation (left) and DER as a function of the input scaling factor α_U for a system encompassing a single layer RCN with 500 nodes.

The next step consists of identifying the minimum number of scan steps a readout is supposed to remain constant. Given that white spaces are often not more than 4 pixels wide and that the core of a digit like '1' may be as narrow as that as well, we select $T_{\min} = 4$ as a realistic value. For this value, Equation 8.3 leads to a leak rate $\lambda = 0.22$.

The third step consists of finding the proper input scale factor α_U. The equation for solving α_U as a function of the other reservoir parameters can be found in [3].

Here we verify whether this function leads to a good result in the simple case of a recognizer built with the reservoir we used for measuring the power spectrum of the input activation. For this reservoir, the function reduces to

$$\alpha_U^2 \; K^{in} \; V_U \; \phi_b(F) = V_{opt} = 0.035 \tag{8.4}$$

with $K^{in} = 5$, $V_U = 1$ and

$$\phi_b(F) = \frac{\int_{-F}^{F} |B(f)|^2 \, df}{\int_{-0.5}^{0.5} |B(f)|^2 \, df} = 0.85 \tag{8.5}$$

The result is $\alpha_U = 0.28$. In order to verify whether this is a suitable value, we reused the same reservoir in combination with a large number of input scaling factors. Figure 8.6(b) shows the DER of the digit recognizer on a validation set as a function of this factor. It appears that 0.28 is in the middle of the optimal range from 0.2 to 0.5.

We also verified whether $\rho = 0.65$ and $\lambda = 0.22$ were appropriate values. Again, we considered a reservoir with 500 nodes. Since it was already shown in [3] that λ and ρ can be optimized independently of each other, we made two sweeps: one along the ρ and one along the λ axis. The results of these experiments are depicted in Figure 8.7. Apparently, the values originating from the theory are close to the actual optimum points.

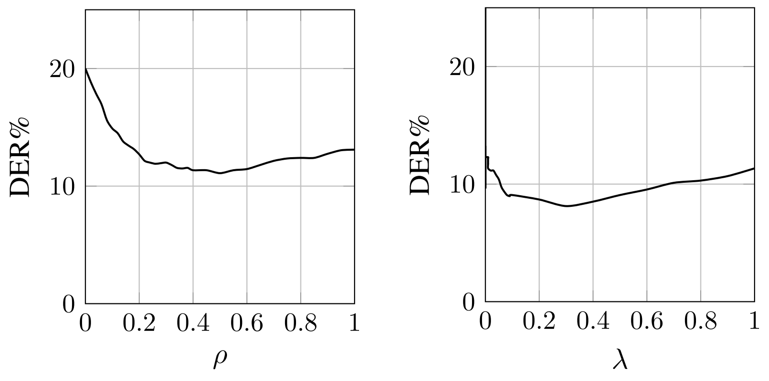


Figure 8.7: Control experiments to optimize reservoir dynamic parameters, ρ (left) and λ (right) when the normalized pixels are used as the inputs to a single layer memory-less RCN with 500 nodes.

Design of reservoirs in the higher layers

For the reservoirs in the higher layers, the inputs are always supposed to be close to the desired outputs already. Following the same recipe as before, we obtain $\rho = 0.4$. Since the target outputs are the same in all layers, the value of $\lambda = 0.22$ is the same as well. The input scaling factor is then computed according to Equation 5.22.

Summary

In this section, we have first of all gathered evidence that the formerly proposed parameter fixing strategy is trustworthy and that the value of V_{opt} that worked well for spoken digit recognition works well here. This saves a lot of time as the need for time consuming parameter sweeping experiments is eliminated completely.

By applying the strategy, we obtained the following reservoir settings: in the first layer, the reservoir parameters are fixed to $(K^{in}, K^{rec}, \rho, \alpha_U, \lambda)$

= (5, 5, 0.65, 0.28, 0.22); in all other layers, they are fixed to (5, 5, 0.4, 0.6, 0.22).

8.5 Experimental Results

In this section, we assess the performance of our systems as a function of the reservoir size (the number of neurons in the reservoir), the depth of the RCN (the number of layers) and the direction of scanning in the front-end. Unless stated otherwise, the RCNs are bi-directional and a bi-directional RCN with a reservoir of size N is actually encompassing two independent reservoirs of size $N/2$ working in parallel. The number of trainable parameters of such an RCN is $11 \times (N+1)$ (the extra 1 represents a bias for each readout node).

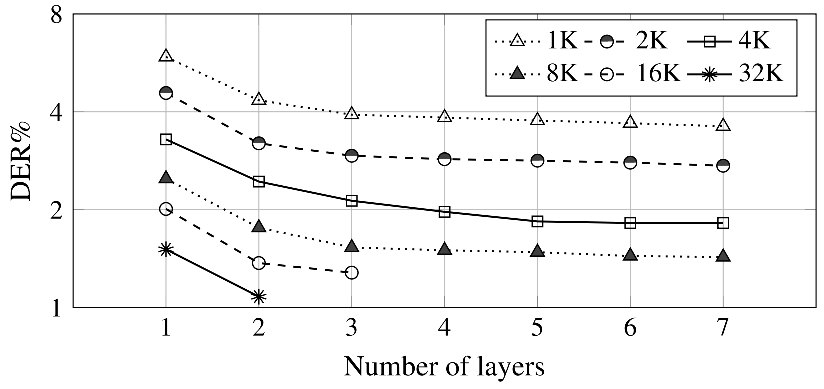
8.5.1 Deep versus Wide

First, we compare single-layer and multi-layer RCNs in combination with horizontal scanning. In the case of a multi-layer RCN, the reservoir size is kept the same in each layer. The results depicted in Figure 8.8 support the following conclusions:

- Any single-layer system can be improved by adding extra layers and the relative reduction of the DER due to adding a second layer is about 25%, irrespective of the reservoir size.
- The effectiveness of adding a layer decreases very quickly with the depth of the RCN. In general, there is no point in creating systems with more than three layers.
- Even though a multi-layer system does not improve DER significantly upon a single-layer system encompassing the same number of trainable parameters, the former is easier and faster to design. In fact, the memory load and the training time are roughly proportional to the square of the reservoir size, meaning that for the training of a one-layer RCN with a 32K reservoir, one needs four times more memory and two times more time than for the training of a two-layer RCN with a 16K reservoir in each layer.

8.5.2 Scanning Directions

In a second experiment, we assess the impact of the image scanning direction (Figure 8.9) and the scanning combination strategy (Table 8.2) on the system performance.



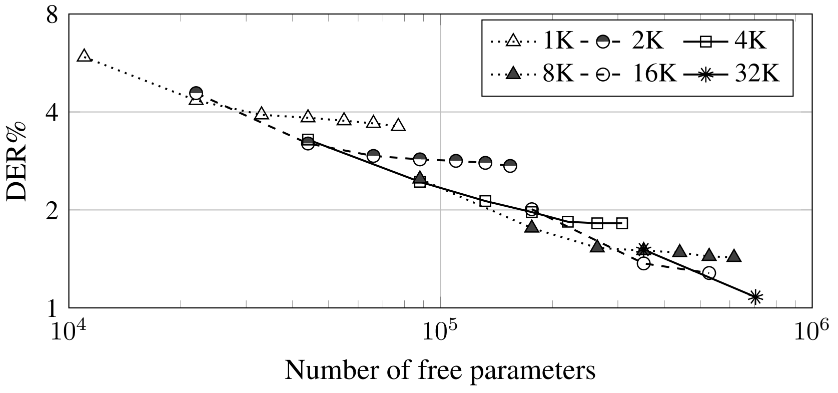


Figure 8.8: DER (in %) on the validation set as a function of the reservoir size and the number of layers (top) and the same results, but as a function of the number of trainable parameters (bottom).

Figure 8.9 clearly shows that in the case of a single-layer system, horizontal scanning outperforms vertical scanning; whereas for the deep systems, vertical scanning tends to produce slightly better results. The differences may be due to the fact that most digits occupy a smaller part of the image in horizontal than in vertical direction, as illustrated in Figure 8.10. This means on the one hand that a single column carries more information about the digit on average than a single row, which is beneficial for horizontal scanning. On the other hand, this means that the digit score in vertical scanning systems is based on more frames; what should normally favor this scanning direction. Apparently, the first phenomenon is more dominant than the second one in a one-layer system, whereas the second

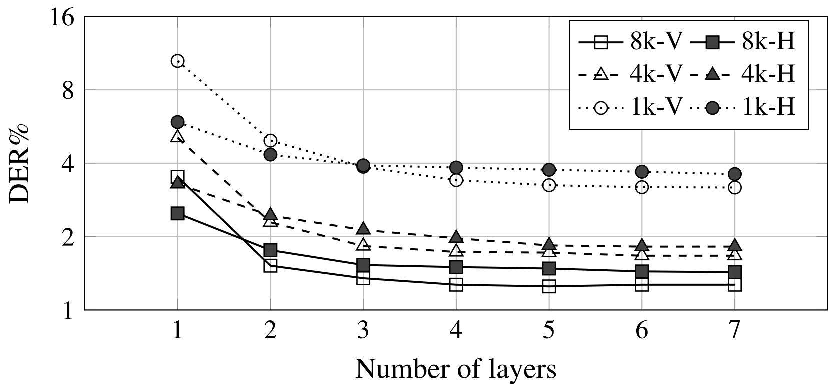


Figure 8.9: Comparing the effect of horizontal image scanning (H) with the vertical procedure (V) for three multi-layer RC-based systems having 1K, 4K, and 8K nodes per layer.

	H	V	H-V-Inp	H-V-wscr	H-V-Res
DER%	1.52	1.39	1.48	1.30	**1.18**

Table 8.2: Comparing different input scanning options.

phenomenon is more dominant in the multi-layer systems.

Since both scanning directions seem to have pros and cons, it makes sense to combine them in one system. Table 8.2 lists the results of five systems: (1) H: one 2-layer system with 5K reservoirs and horizontal scanning, (2) V: one 2-layer system with 5K reservoirs and vertical scanning, (3) H-V-Inp: one 2-layer system with 5K reservoirs driven by the concatenation of the two scanning directions, (4) H-V-wscr: two 2-layer systems with 2.5K reservoirs (one per scanning direction) whose digit scores are linearly combined, and (5) H-V-Res: two 2-layer systems with 2K reservoirs (one per scanning direction) followed by a single-layer system with a 2K reservoir.

Apparently, system H-V-Res is the best and it clearly outperforms both single scanning systems. This indicates that the H and the V readouts for a frame together form a richer feature space for the final classification of the frames.

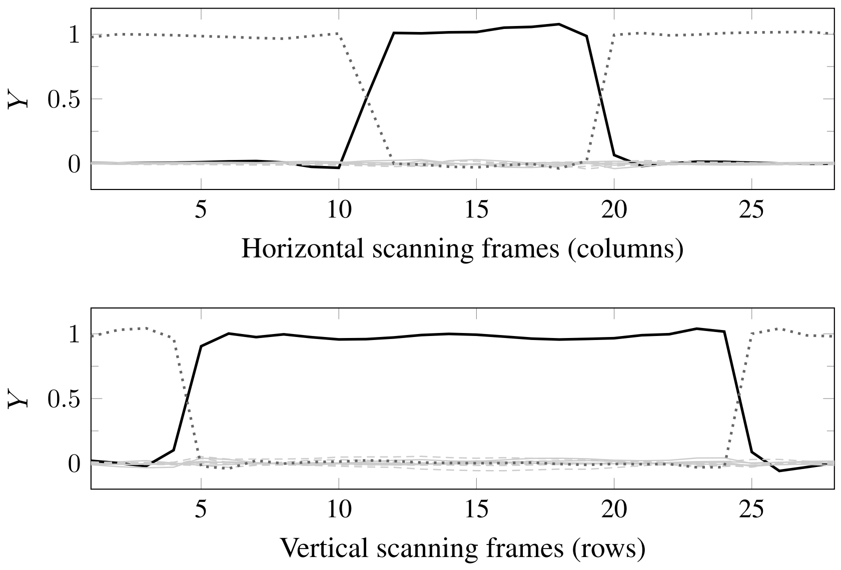


Figure 8.10: The readouts of reservoirs working with horizontal (top) and vertical (bottom) scanning for a sample of digit 1. The strong black line is the readout of digit 1, the dashed line is the readout of white space class.

8.5.3 Final Result

Based on the above findings, we designed a system of type H-V-Res comprising two 2-layer systems comprising a 16K reservoir in each layer, followed by a single layer RCN encompassing a 16K reservoir. This system has 880K trainable parameters and it achieves a DER of 0.81% on the MNIST test set (see Table 8.3), showing that it is competitive with formerly reported systems working with the same inputs (grey-scale pixels) and being trained on the same training samples.

8.6 Noise Robustness

In this section, we study the noise robustness of our RCN systems. First, we consider systems that recognize digits from raw noisy images and later, we consider systems that recognize digits from denoised (cleaned) images.

8.6.1 Recognizing Raw Noisy Images

In this case, we distinguish two experimental settings: one in which the system is trained on clean images only (clean training) and one in which

System	Year	DER%
DNN committee [175]	2011	1.70
SVM [176]	2002	1.40
DBN [166]	2010	1.03
DBM [167]	2009	0.95
Deep Convex Network [168]	2011	0.83
Deep RCN	[This work]	**0.81**
Large Convex Network [170]	2006	0.60

Table 8.3: Results obtained with several state-of-the-art systems that used the raw training samples, without applying any image deformation to extend the training material.

the system is trained on a mix of clean samples and samples corrupted by the five noise types that are also present in the test set (multi-style training). We present DERs for each of the six subsets of the multi-style test set. Note that multi-style training is bound to yield an optimistic result as all noise types encountered in the test data were also present during system training. Nevertheless, we followed this recipe to have comparable results with other references.

The results of our experiments are listed in Table 8.4. For comparison with the state-of-the-art, the table also includes the results for DBN systems we could find in the literature. In the clean training case, the presence of noise induces a dramatic increase of the DER in all systems. None of the systems stands out on all conditions. The DBN system wins in three of the six conditions, the RCN in the other three, be it that on average the DBN system yields the lowest DER. It is fair to say that RCNs degenerate at more or less the same pace as DBNs when the mismatch between the training and the test conditions increases. We interpret this as a positive result because deep neural networks are acknowledged for their good noise robustness and because the research on RCNs is still in its initial phase.

In the multi-style training case, the effect of the noise is much more moderate. The H-V-Res system now yields an average error rate of only 3.54% and it outperforms the DBN systems in all conditions for which a comparison is possible. Combining two scanning directions seems to help significantly as long as there is no big mismatch between the training and the test conditions (that means clean test for clean training and all tests for multi-style training). More research is needed to establish why the advantage of the combination disappears in mismatched conditions.

None of the tested systems stands out on all noise types, but on average, the H-V-Inp architecture is the winner because there is no noise type

	System	Clean	Gaussian	S & P	Speckle	Block	Border	Average
Clean	DBN-2010	1.03	-	-	-	33.78	66.14	-
	DBN-2013	1.09	**29.17**	**18.63**	**8.11**	25.72	90.05	28.80
	V	1.11	57.04	56.27	72.96	24.97	85.49	49.64
	H	1.28	31.43	40.91	45.91	25.41	60.99	34.32
	H-V-Inp	1.18	29.46	40.94	30.70	22.12	**16.97**	**23.56**
	H-V-wscr	0.89	32.12	38.47	48.50	**21.79**	64.94	34.45
	H-V-Res	**0.81**	32.10	38.91	49.32	21.85	79.34	37.06
Multi	DBN-2010	1.68	-	-	-	8.72	1.95	-
	V	1.88	4.73	6.06	7.38	9.50	2.45	5.33
	H	2.28	4.12	5.17	5.65	9.10	2.42	4.79
	H-V-Inp	2.28	4.20	5.22	5.23	8.96	2.63	4.75
	H-V-wscr	1.65	3.12	3.90	4.47	7.20	1.93	3.71
	H-V-Res	**1.50**	**3.08**	**3.75**	**4.32**	**6.82**	**1.75**	**3.54**
	MC-RCN	2.82	4.54	6.07	6.22	9.82	3.23	5.45

Table 8.4: DER (in %) per noise type for the cases of clean and multi-style training. The last row shows the DER of a multi-column RCN-based recognizer comprising twelve sub-systems each trained on one noise condition and one direction. The systems DBN-2010 and DBN-2013 refer to [166] and [173], respectively.

for which it completely breaks down. This system performs exceptionally well for the Border noise. The DBN-based system on the other hand is the winner for three of the five noise types.

In order to further investigate the difference between the performances of H-V-Inp, H and V, we should take look at Figure 8.11 which shows the readouts of the three systems for a noisy sample of digit 7 surrounded by a border. This figure shows that in the case of H and V, the winner is mainly determined by the digit readout that reaches the highest value at the beginning and the end of the scan, where only the black border is observed. In the case of horizontal scanning, this seems to be '1' which often comprise a more or less vertical line that bares a lot of resemblance with the black border. For the H-V-Inp system, none of the digit readouts seems to match the black border and the correct winner is more likely to pop-up.

To confirm our hypothesis we investigated in more detail the confusions between the recognized and the correct digit in the case border noise is present. Table 8.5 shows the digits that were recognized in case of a wrong decision. For instance, it is indicated that with the H-RCN system, 8501 of the validation samples have been wrongly classified as digit 1.

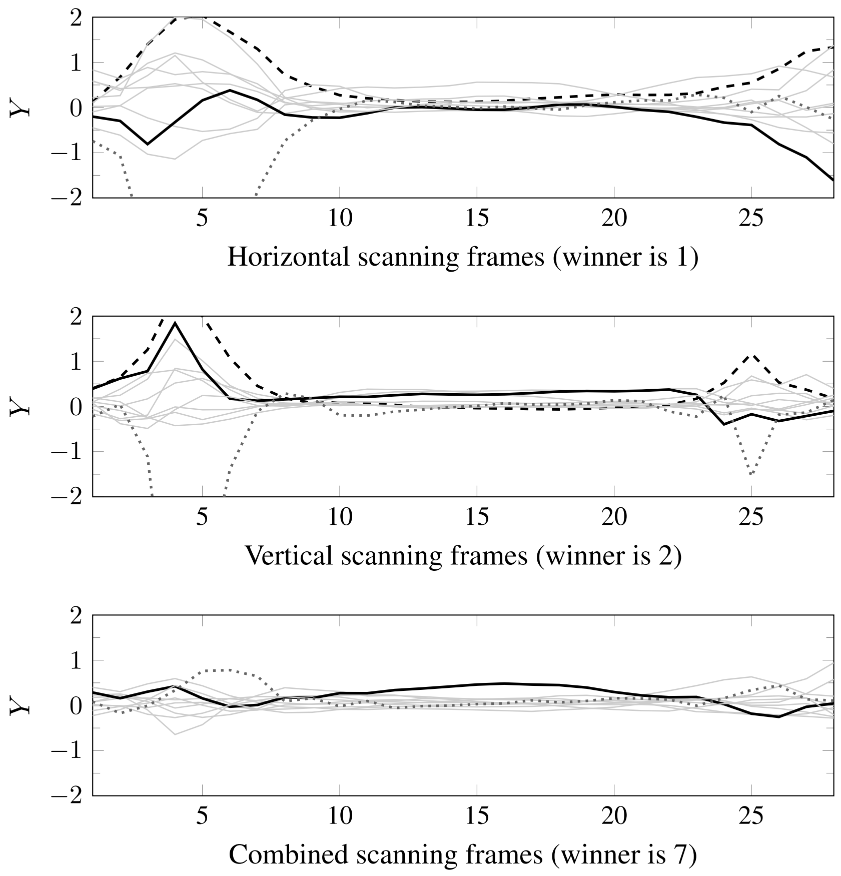


Figure 8.11: The readouts of the H, V, and H-V-Inp recognizers trained on the clean dataset and fed with an image of digit 7 corrupted with border noise. The solid black, dashed black, and dotted gray lines belong to the readouts of digit 7, the winner digit and white space, respectively.

	0	1	2	3	4	5	6	7	8	9	Sum
H	0	8501	4	0	33	0	0	0	11	1	8549
V	0	0	5180	13	69	788	0	50	0	0	6099
H-V-Inp	2	1533	2	1	98	17	0	41	2	1	1697

Table 8.5: Distribution of the wrong decisions in case of border noise.

Apparently, the H-system is strongly biased towards the digits exhibiting a strong vertical line (‘1’ and ‘4’) whilst system V is biased towards digits with horizontal lines on top, bottom or center (‘2’, ‘4’,‘5’ and ‘7’). A simple solution to reduce the effect of the false positive reaction to a border, without seriously degrading the performance on the other noise types, is to bound the readouts to an acceptable interval such as (-0.05, 0.25). By doing so, the DER for Border noise can be decreased to 64.93%, 34.79%, and 15.42% for the V, H, and H-V-Inp systems, respectively. This, in its turn, leads to average DERs of 44.26%, 27.61%, and 22.09%, respectively.

In Section 8.5.2 we also showed that in the case of clean test samples, H-V-Res performs best, followed by H-V-Inp. Combining this with the now obtained clean training results, we conjecture that H-V-Res can indeed learn very well what it has seen during training, but that this knowledge does not generalize so well to unseen conditions.

For multi-style training, only two results are reported for the DBN system, but the figures in Table 8.5 do not contradict the conclusion that RCN-based systems are even more robust to noise than DBN systems which are generally acknowledged for their good robustness in comparison to other techniques.

For completeness, we also trained a two-stage multi-column RCN (MC-RCN) recognizer. The first stage encompasses twelve 2-layer RCNs with a 3K reservoir per layer, one RCN per noise type (6 noise types) and per scanning direction (2 directions). The outputs of these twelve RCNs drive a 2-layer RCN with a 4K reservoir per layer. The reservoir sizes were chosen such that the system has the same number of trainable parameters as the H-V-Res system. Apparently, the MC-RCN does not even outperform the much simpler H and V systems (see Table 8.4). Our hypothesis is that the reservoirs in the first stage are too small to permit an accurate classification. This was confirmed by an experiment in which the reservoir size was increased to 8K (leading to 2M trainable parameters) and in which the error rate dropped to 3.41%. Increasing the size of the H-V-Res system on the other hand did not cause any significant improvement (error rate of 3.49%). This latter results proves that the MC-RCN, in spite of its much larger complexity, will in the end not significantly outperform the much simpler H-V-Res architecture.

8.6.2 Removing the Noise in the Front-End

Multi-style training is one approach to reduce the mismatch between training and testing. Another options is to apply noise reduction in the front-end. We propose an RCN-based denoising Auto-Encoder (DAE) to accomplish this.

For fixing the hyper-parameters of the DAE reservoirs, we follow the same strategy as before, but this time under the assumption that the dynamics of the targeted outputs are identical to the dynamics of the inputs. Moreover, we established that bi-directional processing is helpful for this task but that it suffices to stack three (instead of five) successive frames in the DAE input. Since the output of the DAE is a denoised version of the input feature vector, the number of trainable parameters of such an RCN-based DAE of the size N is $28 \times (N + 1)$, with 28 being the number of pixels per column/row.

First, we consider a single noise-independent DAE and call it a mixed DAE (MixDAE). Then, we set-up a committee of five noise-specific DAEs which together drive a noise-independent DAE (see Figure 8.12). We call this system a combined DAE (ComDAE).

In order to assess the effectiveness of a DAE, we have to define a good measure of the amount of noise that is present in a noisy image. Therefore, we used the Noise Fraction (NF) as the ratio between the variance of the noise values and the variance of the clean pixel values. The values of NF are between 0 and 1, with NF = 0 meaning a clean image.

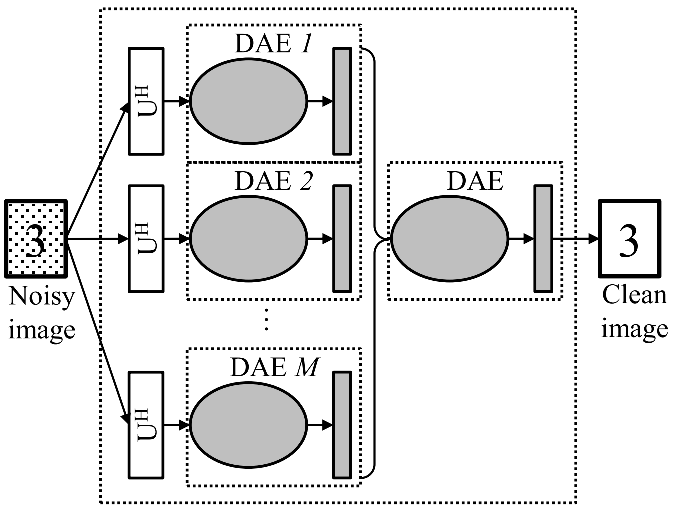


Figure 8.12: Architecture of the combined DAE: the outputs of a committee of noise-specific DAEs (M different noise types) drive a noise-independent DAE.

Figure 8.13(a) shows the mean NF obtained on the validation set in function of the reservoir size and the noise type after denoising the image by means of a single-layer MixDAE using horizontal scanning.

- With a reservoir of size 4K, the mean NF is already smaller than

0.2 for all noise types. The mean NF is in all cases significantly smaller than the mean NF of the raw noisy images. This mean ranged between 0.4 and 0.83 depending on the noise type (see header of Figure 8.13(a)).

- The noise reduction improves very gradually as the reservoir size increases. There is no clear bend in the curve for any of the noise types.

- Border noise, the most problematic noise type in our previous experiments, can be removed almost completely. This follows from the fact that it is very easy to establish where it occurs and which clean pixel value the DAE has to predict there. It is, therefore, not surprising to find that the NF after denoising of an image corrupted by border noise is even lower than that of a clean image that is being denoised (there, the output of the DAE depends on the location of the digit in the image).

- Speckle noise is the only noise type for which the NF is almost independent of the size of the DAE.

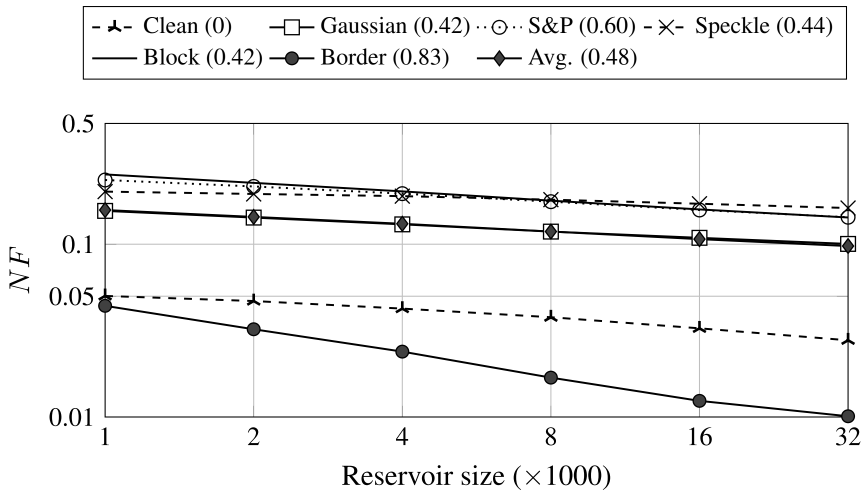
The effect of stacking multiple RCNs with 32K reservoir nodes on the average NF is depicted in Figure 8.13(b). Adding a second layer clearly induces an additional gain whilst further layers are not beneficial anymore.

Without reporting the results in detail, we mention that neither changing the scanning direction nor combining two scanning directions in an H-V-Res like system leads to any significant improvement. This is not so surprising as the aim of denoising is to find and remove the noise patterns and the noise types encountered in this work are direction-independent.

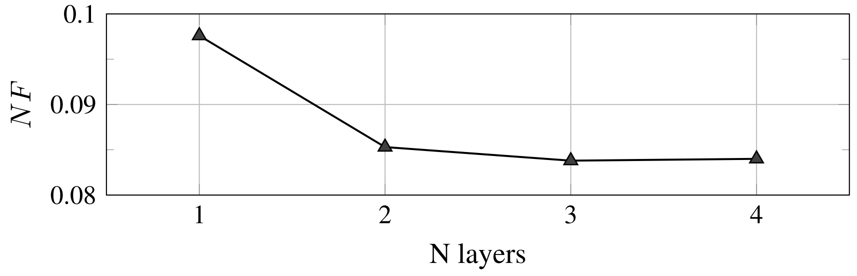
Based on the above findings, we also considered a 2-layer MixDAE with 32K reservoirs in each layer as the reference against which we will compare the ComDAE. To make ComDAE equally large as the MixDAE (in terms of trainable parameters), the former is composed of five 2-layer noise-specific DAEs with 6K reservoirs per layer and a single-layer noise-independent DAE with a 4K reservoir.

Finally, in a control experiment we also consider the first stage of the ComDAE and select from that stage the output of the DAE that was designed for the type of noise that is known to be in the test sample. This so-called ideal DAE is denoted as IdlDAE.

Figure 8.14 summarizes the results obtained with the three DAEs. It supports the following conclusions:



(a) Optimizing the reservoir size for a single layer DAE. The figures between brackets in the legend represent the NF of the original noisy images.



(b) Optimizing the number of layers for a DAE with 32K neurons per layer.

Figure 8.13: Optimizing the reservoir size and the number of layers for an RCN-based DAE.

- For two noise types (Gaussian and S&P), the ComDAE achieves a noise reduction that is nearly identical to that of the IdlDAE, but on three other types, the MixDAE is better than the ComDAE.
- On average, there is little difference between the simple MixDAE and the much more complex ComDAE.

Figure 8.15 shows some examples of what the MixDAE can achieve.

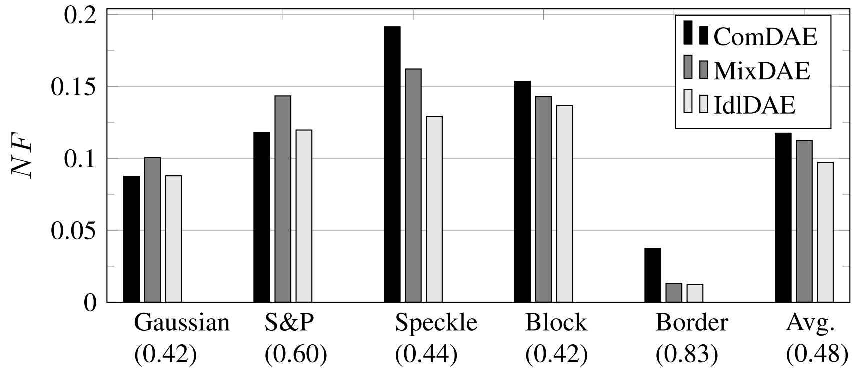


Figure 8.14: The noise fraction (NF) for the output of the mixed, the combined and an 'ideal' DAE that has prior knowledge of the used noise type. The NF of the raw noisy images are mentioned between brackets.

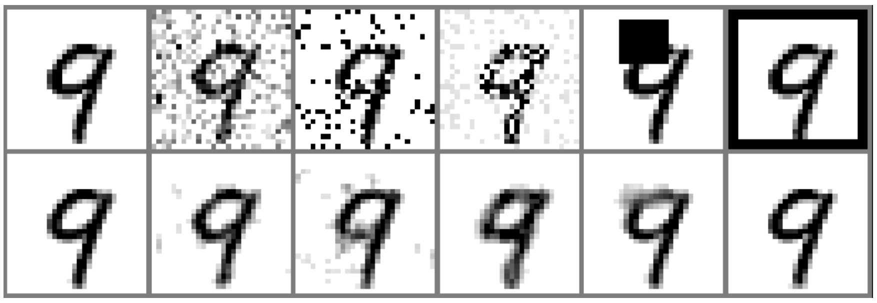


Figure 8.15: One clean and five noise corrupted samples of digit 9 (top) and the corresponding outputs of the MixDAE.

8.6.3 Recognition of Denoised Images

The aim of the DAE in this work is not just to reduce the NF but to permit good recognition on the basis of the denoised images. First, we test the cascade of the MixDAE and the H-V-Res system we formerly developed on clean images (that is clean training). The results obtained with this cascade are listed in Table 8.6. The table also includes the performances of the adaptive multi-column stacked sparse denoising auto-encoder (AMC-SSDA) reported in [173] and the RBM-based denoiser reported in [166]. It is clear that the MixDAE introduces a dramatic gain in noise robustness of the H-V-Res system at the cost of only a minor loss of accuracy in the case of clean images. Furthermore, the H-V-Res system with MixDAE outperforms the AMC-SSDA system in five of the six conditions.

Classifier DAE	DBN-2010 RBM-based	DBN-2013 AMC-SSDA	H-V-Res -	H-V-Res MixDAE	H-V-Res MixDAE
Clean	1.24	1.5	**0.81**	1.03	1.22
Gaussian	-	1.47	32.1	**1.33**	1.57
S & P	-	2.22	38.91	**1.86**	2.17
Speckle	-	**2.09**	49.32	2.41	2.19
Block	19.09	5.18	21.85	4.95	**3.94**
Border	1.29	1.15	79.34	**0.89**	1.25
Average	-	2.27	37.06	2.08	**2.06**

Table 8.6: The influence of adding an RCN-based DAE in front of the classifier on the performance of the RCN-based recognizer (as DER%) on the noisy version of MNIST dataset. The systems DBN-2010 and DBN-2013 refer to [166] and [173], respectively.

In theory, the just tested configuration is sub-optimal because it implies a mismatch between training and testing. Therefore, we also trained an H-V-Res system on denoised training images. However, to our surprise, the results in Table 8.6 show no significant improvement over the sub-optimal system. Apparently, there is no need to retrain the recognizer every time the DAE is improved, for instance, by taking a new noise type into account.

The results we obtain for the H-V-Res system embedding a mixed DAE show that image denoising in combination with clean training is more effective than multi-style training, even though the latter is over optimistic because it is tested on noises that were present during training. This is a remarkable result since a limited study in [166] involving border noise and block noise came to the opposite conclusion for a system encompassing sparse DBNs. In that study, a clean trained DBN, a multi-style trained DBN, and a clean trained DBN supplied with the denoised images led to the DERs of 22.7%, 4.6% and 6.4%, respectively.

8.7 Conclusion and Future Work

The aim of this work was to investigate the potential of reservoir computing networks (RCNs) in the context of image processing, with a particular focus on handwritten digit recognition (HDR) and image denoising. An RCN is peculiar in the sense that it consists of a hidden layer of recurrently connected non-linear neurons with fixed (that means non-trained) coefficients – called a reservoir – and an output layer of linear neurons with trained coefficients which 'read out' the outputs of the reservoir. The two key properties of an RCN are that it is easy to train and that it generalizes well to unseen conditions. We showed that a large enough RCN recognizer can surpass conventional neural network-based recognizers in matched conditions. Moreover, we established that an RCN can be very effective in denoising an image and that the combination of the denoiser and the recognizer outperforms a similar combination created by means of a conventional deep neural network technology.

Now that good HDR and image denoising have been demonstrated, time has come to think about using RCNs for large vocabulary continuous handwriting recognition and for other image processing applications as well.

Part IV

Conclusive Discussion

9 Conclusive Discussion

I conducted my research as part of the European ORGANIC[1] project. The aim of the project was to investigate whether complex systems like systems for handwriting and speech recognition can benefit from employing neuro-dynamical components as alternatives to the more conventional generative statistical components.

The claims made by researchers in machine learning were that neuro-dynamical systems are more powerful as they can analyze long-term relationships in a natural way. Therefore, they can make a distinction between the dynamics of the signal and those of distortions that have corrupted the signal before the signal reaches the recognition system.

In order to verify these claims for the case of speech recognition, we devised two research paths. One path should demonstrate the ability of neuro-dynamical systems to extract long-term dynamical properties of the speech and as such raise the recognition performance. Since long-term dynamics are expected to be most important in running speech with a rich vocabulary, we opted for large vocabulary continuous speech recognition (LVCSR) in this path. The second path was to demonstrate that neuro-dynamical systems better suppress the effects of ambient noise and channel distortion than conventional generative model-based systems that only look at local information. As noise robustness is commonly achieved in the acoustic

[1]ORGANIC stands for stands for Self-Organized Recurrent Neural Learning for Language Processing (see also http://organic.elis.ugent.be)

model, and since the performance of a small vocabulary continuous speech recognition (SVCSR) system is almost entirely determined by the quality of the acoustic model it encompasses, we decided to study the robustness in the context of SVCSR. In both paths, we focused on a specific category of neuro-dynamical systems, namely reservoir computing systems.

Although the main body of my work concerned noise robustness in SVCSR (path 2), I did contribute to the development of reservoir-based LVCSR (path 1) as well. This chapter starts with a brief review of my contributions to LVCSR and the role they played in my work in path 2.

9.1 LVCSR with Reservoir Systems

As explained in Chapters 1 and 2, an LVCSR decoder relies on three major components: a language model, an acoustic model and a pronunciation dictionary. The language model and the pronunciation dictionary make it possible to model each speech utterance as a sequence of basic linguistic units, called phonemes. The acoustic models are then responsible for assessing the probabilities that selected speech fragments correspond to particular phonemes.

9.1.1 Reservoir Systems

A reservoir system is a particular type of neural networks. It is composed of one or more Reservoir Computing Networks (RCN). Each RCN consists of two parts: (1) a pool of recurrently connected non-linear neurons (a reservoir) with fixed weights that project the low-dimensional input vector into a high-dimensional reservoir state space and (2) a set of linear readouts that perform a linear regression in that space. As such, the RCN acts like a Support Vector Machine (SVM) which similarly projects the inputs into a high-dimensional inner space and performs a linear regression in that space.

Due to its recurrent connections, the reservoir can be interpreted as a nonlinear dynamical system: the output of each reservoir neuron can be envisioned as the output of a nonlinear recursive filter that is stimulated by a multi-dimensional acoustic input pattern.

Since the weights of the reservoir are fixed, the training of an RCN reduces to finding the best linear regressor in the reservoir state space. By using a mean squared error (MSE) criterion, one achieves that the envisioned regressor is the solution of an over determined set of linear equations. Important is that the training of an RCN is straightforward because it does not require an iterative process that can get stuck in a bad local optimum.

9.1.2 Phone Recognition

The first step in assessing the effectiveness of a new technology such as RCN for creating acoustic models is usually to create a phoneme recognizer encompassing these models. The phoneme recognizer converts the speech into a sequence of phonemes. This sequence is then compared to the sequence written down by phoneticians after having listened to the speech. In practice, some phonemes break down into smaller atoms and some phonemes have well established variants (called allophones) depending on the context in which they appear. All these atoms and variants together constitute a set of units, called phones. The quality of the acoustic model is quantified by the Phone Error Rate (PER) which measures the minimum number of edit operations (delete/insert/substitute) one has to perform to convert the correct phone sequence into a recognized phone sequence. Most work on phone recognition worldwide is conducted on the internationally renowned TIMIT benchmark [177]. That is why I also worked on that benchmark.

In the beginning of the ORGANIC project, I collaborated with my colleague Fabian Triefenbach to design RCNs that adequately model phones in running speech. After all, a good baseline system for a well-known condition is needed before one can investigate the robustness against changes in the condition, which was the main objective of my research.

Our first RCN-based phone recognizer that was competitive with the state-of-the-art at the time was reported in [8]. We learned in this phase that good recognition can only be achieved with very big reservoirs containing for example 20k neurons. We also learned that cascading RCNs significantly improves performance for a given reservoir size. The latter result is particularly important because both the required memory and the computational cost during system development are roughly proportional to the square of the reservoir size.

Next to our collaboration in the initial phase of the ORGANIC project, my work on a good reservoir design methodology helped to speed up the research towards a fully competitive RCN-based phone recognizer [9]. Table 9.1 lists the PER obtained with that recognizer together with some published PERs of the best state-of-the-art systems till 2013. What I learned from this research was that non-causal bi-directional systems outperform the causal uni-directional systems (PER of 23.1% versus 24.2%).

Another LVCSR result that was relevant to my research towards robust ASR is that RCN-based systems can benefit from front-end techniques such as vocal tract length normalization (VTLN) and utterance-wise mean and variance normalization of the acoustic features. In fact, these methods were able to reduce the PER to 20.5% [10].

System Description	PER
Bayesian Tri-phone HMM [178]	25.6
Bi-directional LSTM-NN [90, 179]	24.6
Causal RCN-HMM	**24.2**
Bi-directional RCN-HMM	**23.1**
Deep Belief Networks [84]	22.4
CD-MLP-Bottleneck Hierarchy [75]	21.2

Table 9.1: Phone recognition error rates (in %) on core test set of TIMIT obtained with state-of-the-art systems.

Acoustic Model	**Bi-gram**	**Tri-gram**
GMM-HMM (MMI training + VTLN) [182]	-	3.0
MLP-HMM hybrid [114]	8.5	6.5
DET-WFSM bottom-up decoding [183]	-	6.0
DNN-HMM hybrid (MFCCs) [184]	5.7	-
RCN-HMM hybrid	**6.2**	**3.9**

Table 9.2: WER in % on Wall-Street Journal (Nov-92 evaluation set, 5k) using different acoustic models in combination with a bi-gram and tri-gram language model.

9.1.3 Large Vocabulary Recognition

Although we achieved good phone recognition on TIMIT, we need to provide good LVCSR on an international benchmark that was specifically created for testing LVCSR, to convince people that RCNs have potential for acoustic modeling. In [180], we presented an RCN-HMM hybrid that offers competitive LVCSR results on the Wall-Street Journal (WSJ0) benchmark [181]. Table 9.2 lists the performance of our best system along with some recently reported performances of the competitors. Again, my fundamental work on the fast design of good reservoirs allowed us to achieve such a promising result in a very limited time.

9.2 Noise Robust RCN-Based Speech Recognition

The main focus of this dissertation was of course on the noise-robustness of Reservoir Computing. In this respect, I conducted a large number of experiments on continuously spoken digit recognition and a small number of experiments on isolated handwritten digit recognition. In particular, I addressed the following two questions: (1) Is it possible to derive universally applicable reservoir design principles that yield appropriate reservoir parameters for any task in any domain? (2) To what extent are RCN-based models robust to noise and what is the main reason for this robustness?

However, before formulating answers to these questions, I want to recapitulate the most important results and achievements of my research into noise-robustness.

9.2.1 Whole Word Acoustic Models

In order to investigate the importance of the dynamics of the reservoir, I started with using RCNs for the acoustic modeling of whole words (isolated digits) rather than phones. This work showed that recurrent connections are indispensable and that additional dynamic modeling provided by Leaky Integration Neurons (LINs) improves the recognition as well. The combination of recurrent connections and LINs gives the user more control over the short-term and long-term memory of the reservoir. Using that combination I achieved state-of-the-art performance for clean isolated digit recognition (WER = 0.12%) on Aurora-2 [29]. My next goal was to extend the task to continuous digit recognition in the presence of noise. By adopting a hybrid RCN-HMM framework and by supplying the reservoir with noise robust acoustic features (MSVA) I created a competitive noise robust RCN-based continuous digit recognizer. However, more research was needed to pinpoint where exactly this noise robustness came from. Furthermore, the RCN-based recognizer was not yet competitive with the state-of-the-art in clean conditions.

My collaborative work with Fabian Triefenbach on LVCSR showed that cascading reservoirs can increase the complexity of the system without raising the hardware requirements. Moreover, the higher layers of a cascade are able to correct error patterns that were learned by the previous layers. More precisely, more layers always lead to better performance until saturation occurs. This behavior is far more appealing than that of DNNs which often exhibit a very fluctuating performance and no clear saturation when more and more layers are added.

By working with mean and variance normalized MFCCs, by layer-wise optimizing the RCN using a combination of user-controlled reservoir parameter fixing and automatic supervised training of the readout weights, and by introducing multi-state whole-word models, I developed a continuous digit recognizer (CDR) with an average WER of 13.7% on different noise types and signal-to-noise ratios (SNR), whereas a state-of-the-art GMM-HMM system achieved an average WER of 16.9% (see Table 4.1).

9.2.2 Fast and Optimal Reservoir Design

A major hindrance to the initial experiments was the fact that any change in one parameter of the reservoir (e.g., the spectral radius, or the input features) required redoing the time-consuming search for the optimal combination of the remaining parameters. This motivated me to exploit the relations among reservoir inputs, reservoir parameters and reservoir outputs in order to reduce the search for a good (near-optimal) reservoir setup. The reservoir parameters to consider in this respect are: (1) α_U, the maximal absolute eigenvalue of the input weight matrix $\mathbf{W}^{in}$, (2) ρ, the maximal absolute eigenvalue of the recurrent weight matrix $\mathbf{W}^{rec}$, (3) K^{in}, the number of inputs driving each reservoir neuron, (4) K^{rec}, the number of delayed reservoir outputs driving each reservoir neuron and (5) λ, the leak rate of the leaky integrator neurons of the reservoir. The first two parameters control the strength of the input and the recurrent stimulation of a reservoir neuron, the next two control the sparsity of the input and the recurrent weight matrices and the last parameter controls the bandwidth of the reservoir outputs.

I first introduced a number of simple principles which should be met for a reservoir system to be maximally effective:

1. a reservoir should have enough memory to capture the relevant frequencies in the dynamics of the input patterns;
2. the leaky integration should permit the reservoir outputs to clearly convey the desired frequencies in the target output class patterns;
3. to obtain a robust solution, the variances of the reservoir neuron activations should be comparable to each other, hence avoiding that a solution is dominated by a few neurons only;
4. the variances of the reservoir neuron activations should be large enough to let the reservoir benefit from the non-linear characteristics of its neurons, but not too large to prevent that too many neurons are frozen by saturation at any given time.

Based on some reasonable assumptions and approximations, these principles could be translated into comprehensive relations between reservoir parameters. Finally, a number of experiments were conducted to prove the validity of these relations. The main outcomes of this research can be summarized as follows:

1. The input and recurrent weight matrices ($\mathbf{W}^{in}$ and $\mathbf{W}^{rec}$) can be very sparse (5 to 10 elements per row regardless of the reservoir size

and the input feature vector size). This can be understood as follows: if the reservoir is large enough, even a sparse connection scheme guarantees that all inputs and all neuron activations computed in the previous time step contribute to the current activation of the reservoir. Moreover, the sparse connection matrices improves the independency of the neurons and speeds up the operation of the reservoir. The latter is easily achieved since nowadays good libraries for sparse matrices exist for most high-level programming languages.

2. The spectral radius ρ must be chosen so that the corresponding time constant characterizing the decay of the reservoir memory emerging from the recurrent connections is long enough to capture the dynamics arising from the information bearing changes in the input, which translated to 50 ms in the case of speech signals.

3. The constant λ must be chosen so that the corresponding integration time is equal to the minimal duration of the time intervals during which the output class remains constant.

4. Given an appropriate ρ and λ, there exists a formula for computing a value for α_U which leads to a proper activation level and a proper balance between the contributions of the present inputs and the past reservoir outputs to the present activations.

5. The same optimal reservoir parameters can be re-used with success for different sizes of the reservoir.

9.2.3 Combining Technologies

The previous experiments showed that changing the statistical model employed to relate speech and acoustic units from a GMM to an RCN provides some measure of noise robustness. However, the decades of research on noise robustness preceding this work have lead to a wide array of alternative techniques. In the subsequent work, we re-investigated some of these techniques in the framework of an RCN-based system.

Employing Noise Resistant Features

First, I investigated whether noise robustness could be further improved by employing noise-resistant acoustic features such as the noise robust advanced front-end features (AFEs) that were especially engineered for improving GMM-based systems on Aurora-2. Introducing AFEs in combination with bi-directional RCNs further lowered the average WER from 11.3% to 9.9%.

Combining GMM with RCN

Since I noticed that the errors made by an RCN-based and a GMM-based system were different to some extent, I also investigated whether further improvements could be obtained by combining the two technologies. In particular, I investigated RCN-GMM and GMM-RCN tandems as well as a fusion approach to combine the likelihoods retrieved from the RCN and the GMM. Whereas the system combination did improve the clean speech results (matched condition), a substantial degradation was observed in the more realistic mismatched condition (training on clean speech, testing on noisy data).

Adaptation to Noise

Another technique frequently employed with GMM-based systems is fast unsupervised adaptation of the system to new conditions. Adaptation to a new condition with minimal effort is an appealing technique since even systems trained on many diverse noise types may still encounter a condition which has not been learned before. To import this functionality in an RCN-based system, I introduced an additional RCN to adapt the scaled likelihoods of the RCN-HMM to a new condition in an unsupervised way. The latter means that no manual transcriptions are needed for the adaptation. I showed that three minutes of adaptation data suffice to train a single layer RCN with a reservoir of 250 neurons that can achieve an average relative improvement of 13% (see Table 4.2).

Denoising the Inputs

Since a denoising feature extracting scheme such as the AFE features improved the result, one could try to further improve this system by means of a possibly better denoising scheme based on RCNs. A neural network that removes the effects noise has on the acoustic features is called a Denoising Auto-Encoder (DAE). I inserted an RCN-based DAE at different points in the MFCC extractor, namely after applying DFT, after the Mel-filter bank, and after the DCT. However, none of these approaches resulted in a significant improvement of the noise-robustness with respect to the case of AFEs.

Summary on the Speech-Related Experiments

My experiments indicate that:

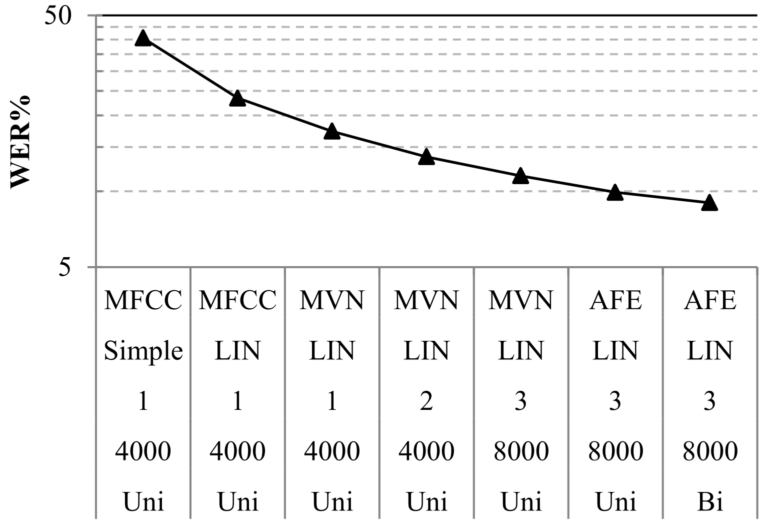
- RCNs focus more on the salient relation between the input features and the acoustic units and less on the fine details, hence they lose some performance in perfectly matched conditions with sufficient training data (clean speech training and clean speech testing), but gain robustness in the other conditions.
- The big multi-layer RCNs have, at least for a small vocabulary task, enough modeling power to do feature denoising and recognition in one step.
- Although noise robust features help, their effect is less pronounced than in the case of GMM-based systems. This may be a direct correlate of the previous observations.
- Despite the added robustness, adaptation to new (unseen) conditions is still needed for optimal performance.

In what follows, I briefly recall the most important steps that lead to the good results I obtained on speech recognition (see also Figure 9.1).

- Replace the simple memory-less neurons with LINs
- Introduce utterance-wise feature normalization (MVNs)
- Employ very large as well as multi-layer reservoirs
- Replace the simple MVN acoustic features by AFE features
- Exploit both past and future observations by introducing bi-directional reservoirs

9.3 Image Classification

Although my research in speech recognition confirmed that an RCN is more robust to the presence of noise than most other acoustic models, it did not prove that this property also holds in general. In the same sense, there was no proof yet that the strategy I introduced to optimize the reservoir parameters works for other tasks as well. To find such proof, I considered the task of handwritten isolated digit recognition. Although many techniques have



MFCC	MFCC	MVN	MVN	MVN	AFE	AFE
Simple	LIN	LIN	LIN	LIN	LIN	LIN
1	1	1	2	3	3	3
4000	4000	4000	4000	8000	8000	8000
Uni	Uni	Uni	Uni	Uni	Uni	Bi

Figure 9.1: The most important results towards noise robust speech recognition with RCNs. The table below the plot describes the type of the acoustic features, type of the reservoir nodes, number of layers, size of each layer, and uni- or bi-directional processing. WER denotes the average word error rates on the SNR 0–20 dB conditions of Test A-C of the Aurora-2 dataset. All systems are trained on the clean conditions.

been introduced to extract good features from the raw image (e.g., Gabor filters), I opted for recognition on the basis of the raw pixels as inputs. This strategy is defensible since this concerned a quick transgression to another domain and since many published systems do start from the raw pixels.

While most systems take the complete image as their input, an RCN is designed to process sequential (temporal) data. Therefore, the RCN-based classifier will scan the image in a sequential way. Hence, a first question that arose was the scanning direction: horizontal, vertical or a combination of the two. I proposed three methods for combining horizontal and vertical scanning, namely, (1) a concatenation of a column and a row in the input vector, (2) a separate classifier per scanning direction followed by a weighted summation of the outputs of both, and (3) a concatenation of the two classifier outputs as the inputs to a third RCN-based classifier (see Figure 8.5). Following the reservoir setup instructions I formulated for speech recognition, I could quickly setup an appropriate RCN-based handwritten digit recognizer and get confirmation that it was indeed quasi-optimally designed. Similar to the speech recognition task, cascading RCNs and using bi-directional RCNs led to better results. In a short time, I created a system trained on clean samples that achieved a WER of 0.81% on the renowned MNIST benchmark. This is fairly close to the 0.60% achieved by a state-of-the-art system using the same input features and convolutional networks (see Figure 8.3).

I also tested the recognition of noisy samples containing five different noise types: Gaussian, Salt and Pepper, Speckle, Block and Border noise. While a deep belief network (DBN) attains an average error rate of 28.8% for these noisy data, the best RCN classifier achieves an error rate of 23.6%. Adding noise to the training set (multi-style training) further reduces this error rate to 3.5%.

Like in speech recognition, I also studied the ability of an RCN-based DAE to denoise the images. In a first experiment I supplied the denoised images to a classifier trained on clean images. In a second experiment I re-trained the classifier on denoised images. The outcome of both experiments is very similar, WERs of 2.08% and 2.06%, respectively. These WERs are somewhat lower than the 2.27% achieved by a DBN-based competitor. The results obtained with RCNs are appealing because first of all they show that an RCN can advance the state-of-the-art and secondly, they show that only the DAE needs to be retrained when a new noise type comes into play and the classifier can be left untouched.

9.4 Answers to the Questions

In this section I contemplate on how far I have come to answering the two research questions I formulated before.

Q1: Is it possible to derive universally applicable reservoir design principles that yields appropriate reservoir parameters for any task in any domain?

The reservoir design strategy I conceived in Chapter 5 which was tuned to the task of clean spoken digit recognition led to quasi-optimal reservoirs for noisy spoken digit recognition, phone recognition (the same input features but another task) and handwritten digit recognition (different input features but a related task). Moreover, the same strategy also led to very good reservoirs for a simple video processing task using the output of a very low quality camera to detect when doors are opened and closed (different input features and a totally different task). Although not discussed in this thesis, the latter experiment is described in [7].

Given the above results and given the fact that the principles underlying the conceived design method are fairly generic and comprehensible, there is sufficient evidence to argue that a universal reservoir design recipe probably exists. However, more experiments on various benchmarks are needed to eliminate all remaining doubts.

Q2: To what extent are RCN-based acoustic models robust to noise and what is the main reason for this robustness?

RCNs benefit from the recurrent connections that add the ability of processing temporal information in a non-linear way. Hence, they are better suited for tasks such as speech recognition for which dynamics are a main property.

Figure 9.2 shows the performances of digit recognizers based on RCNs, ELMs, MLPs [142], RNNs [143] and GMMs [113], all trained on clean Aurora-2 dataset. First of all, the recurrent connections of a reservoir do cause a substantial improvement compared to an ELM, both in clean and in noisy conditions.

Next to looking at the absolute performances of the different system for a given SNR level, one could also try to quantify the robustness of each system. For example, one could compare the relative increase in errors compared to the noise free base-line results across the different systems. Since Figure 9.2 uses a logarithmic error scale, this relative increase in errors corresponds to the slope of each curve. With this respect, the figure shows that the two relatively 'special' approaches (RCN and ELM) have a very comparable noise robustness. On the other hand, they are more noise robust than the more conventional approaches, GMM, MLP, and RNN. Therefore, the robustness of RCN is caused by a property shared with ELM but not with the other systems.

The fundamental difference between the 'special' and the conventional approaches lies in the fact that the first layer in RCNs and ELMs consist of non-trained random values, whereas GMMs, MLPs and RNNs train all parameters across all layers. This, in combination with the MSE optimization criterion, prevent RCNs and ELMs from overfitting the training data and hence let the system generalize better to unseen data than a system with a fully trained parameters [101].

In addition, RCNs have a very simple and fast training procedure which allows the user to enlarge the size of the network to thousands of nodes compared to a few hundred nodes of an RNN. Projecting to such a very high-dimensional space facilitates the discrimination between the classes, both in clean and noisy conditions. Moreover, increasing the network size

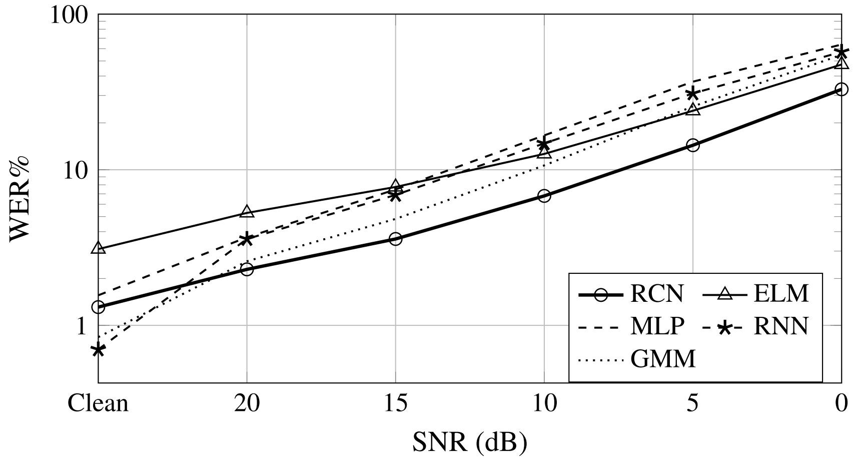


Figure 9.2: Comparing the robustness of state-of-the-art techniques, trained on clean set and tested on Test A-C of Aurora-2 dataset. Both RCN and ELM consist of a single-layer uni-directional hidden layer of 8K nodes.

allows the RCN to focus more on the less conspicuous relations between features and classes, and hence is expected to help more for clean speech where such fine differences are still meaningful. This effect is visible in Figure 9.3. Note that this figure also shows that the MSE training of an RCN is very effective in preventing over-fitting and in maintaining a part of the gain that was achieved in matched conditions in the severely non-matched conditions: even for the noisiest test conditions, the larger reservoirs continue to perform better than the smaller reservoirs.

9.5 Future Work

Although the idea behind reservoir computing is rather simple, it is not easy to fully understand the behavior of a non-linear dynamical system. Therefore, a lot of questions are still unanswered, even in spite of the amount of theoretical research in reservoir computing (e.g. [185]) that has been conducted over the last decade. In what follows I recall some of the questions I consider as still being open:

- Thus far, RCN hierarchies consider the same classes at all levels, but could the idea of deep learning perhaps better be exploited by considering different types of acoustic units, such as phonemes, syllables and words at different levels?

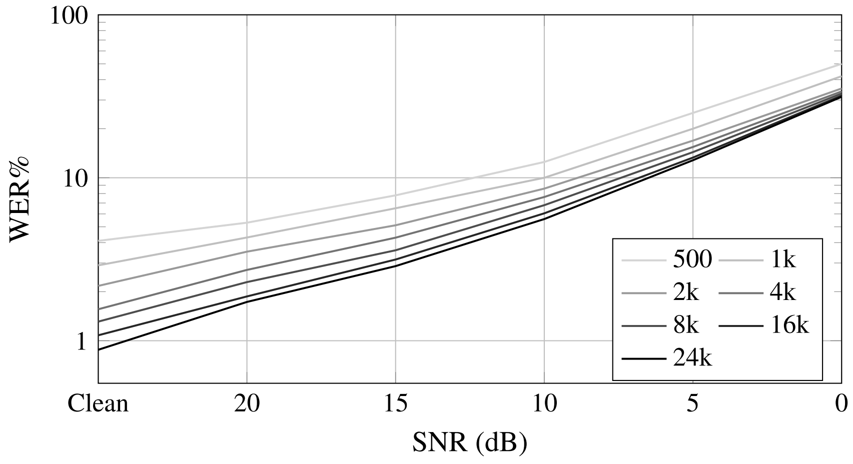


Figure 9.3: The effect of increasing the size of the reservoir on the performance of a single-layer uni-directional RCN-based digit recognizer, trained on clean and tested on Test A-C of Aurora-2 dataset.

- Is there a better but still comprehensible way to initialize the random reservoir weights than the one we developed in this thesis?

- Can a clever grouping of reservoir neurons into blocks which are supplied with different inputs provide a more powerful architecture?

- In a conventional RCN all the readout weights are trained, giving rise to $(N^{res} + 1) \times N^{out}$ trainable parameters. Due to the fact that all the reservoir weights are set randomly, many reservoir neurons may not turn out contribute to the solution of the targeted task. Can one conceive a smart way to prune these dead neurons, globally or per readout node, in order to reduce complexity without loosing accuracy?

- Is an RCN-based system helped by the use of more elaborate input features or by the use of multiple input feature sets?

- Can reservoirs be successfully applied to more challenging image processing tasks, such as medical image recognition?

To conclude this book I would like to emphasize that, compared to other techniques, reservoir computing is still a relatively young domain. I therefore argue that there is still a lot to be done in this domain. Given the opportunity, I would myself be enthusiastic to continue the research, but even if this turns out to be impossible, I hope that the research I presented here will motivate others to continue the search for better neuro-inspired recognition systems.

– The end –

A
Hybrid RCN-HMM Performance on Aurora-2

This appendix contains WER (%) in detail for hybrid RCN-HMM with single- and 3-layer, uni- and bi-directional reservoirs using MFCC, MelFB and AFE features.

The reservoir of each layer contains 8K nodes (2$\times$4K for bi-directional reservoirs). Each reservoir node is supplied with 10 randomly selected input features along with 10 randomly selected recurrent connections (i.e., $K^{in} = K^{rec} = 10$). The HMM consists of 7-state models per digit and a single state for silence.

	SNR	TESTA					TESTB					TESTC			All
		Subway	Babble	Car	Exhib.	Avg.	Restaurant	Street	Airport	Station	Avg.	Subway	Street	Avg.	Avg.
Clean	Clean	1.11	1.45	1.34	1.23	1.28	1.11	1.45	1.34	1.23	1.28	1.26	1.66	1.46	1.32
	SNR20	2.09	2.27	2.24	2.44	2.26	2.21	2.54	2.15	1.73	2.16	2.55	2.84	2.70	2.31
	SNR15	3.81	3.17	3.22	4.35	3.63	2.98	3.05	3.49	3.21	3.18	4.64	4.02	4.33	3.59
	SNR10	6.94	5.71	6.20	8.95	6.94	5.93	6.08	5.43	5.89	5.83	8.93	7.98	8.45	6.80
	SNR5	13.08	12.55	12.05	19.07	14.16	14.06	13.18	11.90	13.33	13.11	18.42	15.96	17.18	14.34
	SNR0	29.72	32.22	28.87	38.01	32.18	36.87	30.08	28.18	30.15	31.29	40.10	34.64	37.35	32.86
	SNR-5	57.81	65.63	55.35	64.21	60.73	70.10	57.80	58.57	58.69	61.26	67.49	62.73	65.09	61.81
	0-20dB	11.13	11.18	10.52	14.56	11.83	12.41	10.99	10.23	10.86	11.11	14.93	13.09	14.00	11.98
Multi	Clean	1.90	1.96	1.88	1.82	1.89	1.90	1.96	1.88	1.82	1.89	1.96	2.18	2.07	1.93
	SNR20	1.72	2.12	1.85	1.76	1.86	1.90	2.21	1.88	1.64	1.91	1.93	2.51	2.22	1.95
	SNR15	2.36	2.57	2.62	2.56	2.53	2.52	2.57	2.65	2.53	2.57	2.79	3.39	3.09	2.66
	SNR10	4.08	3.87	3.73	4.50	4.04	4.11	4.29	4.03	4.10	4.13	4.30	5.53	4.92	4.25
	SNR5	7.40	7.98	7.69	10.27	8.33	9.18	9.70	8.02	9.56	9.11	9.67	10.76	10.22	9.02
	SNR0	19.50	23.67	22.37	22.68	22.06	25.88	23.97	20.82	23.60	23.55	26.07	27.63	26.85	23.61
	SNR-5	46.98	57.07	48.88	50.11	50.77	59.72	52.03	50.19	52.48	53.58	57.78	58.25	58.02	53.34
	0-20dB	7.01	8.04	7.65	8.35	7.76	8.72	8.55	7.48	8.29	8.25	8.95	9.96	9.46	8.30

Table 1: WERs (in %) for test sets A - C obtained with a 1-layer uni-directional hybrid RCN-HMM and MFCC as the input features.

	SNR	TESTA					TESTB					TESTC			All
		Subway	Babble	Car	Exhib.	Avg.	Restaurant	Street	Airport	Station	Avg.	Subway	Street	Avg.	Avg.
Clean	Clean	0.71	0.88	0.75	0.52	0.71	0.71	0.88	0.75	0.52	0.71	0.74	0.85	0.79	0.73
	SNR20	1.60	1.63	1.94	1.60	1.69	1.44	1.69	1.52	1.17	1.46	1.90	2.00	1.95	1.65
	SNR15	3.16	2.48	2.83	3.42	2.97	2.30	2.72	2.74	2.68	2.61	4.02	3.51	3.76	2.98
	SNR10	6.60	4.87	5.64	7.68	6.19	5.04	5.80	4.74	5.18	5.19	8.81	7.22	8.01	6.15
	SNR5	12.56	12.12	12.17	18.17	13.73	13.17	13.00	11.54	12.93	12.65	18.24	15.33	16.77	13.90
	SNR0	29.87	31.74	29.26	38.26	32.25	35.83	29.90	28.96	31.16	31.44	40.31	34.28	37.27	32.93
	SNR-5	59.07	65.66	56.61	65.04	61.57	69.70	59.55	60.45	59.70	62.33	70.00	64.51	67.24	63.01
	0-20dB	10.76	10.57	10.37	13.83	11.37	11.56	10.62	9.90	10.62	10.67	14.66	12.47	13.55	11.52
Multi	Clean	1.17	1.27	1.22	1.08	1.19	1.17	1.27	1.22	1.08	1.19	1.14	1.33	1.23	1.20
	SNR20	0.98	1.30	1.16	0.71	1.04	1.35	1.48	1.25	0.74	1.21	1.11	1.75	1.43	1.19
	SNR15	1.60	1.42	1.61	1.27	1.47	1.54	2.00	1.64	1.64	1.70	1.84	2.36	2.10	1.69
	SNR10	2.67	2.63	2.62	2.93	2.71	3.07	3.20	2.71	2.44	2.86	3.16	4.29	3.73	2.97
	SNR5	5.31	6.35	5.64	7.90	6.29	7.58	7.71	6.26	7.90	7.36	7.80	9.19	8.50	7.16
	SNR0	16.00	21.70	18.82	19.59	19.04	24.62	20.95	19.36	21.75	21.65	22.90	24.85	23.88	21.05
	SNR-5	43.69	56.86	44.86	46.71	48.04	60.55	49.91	50.01	49.74	52.53	55.88	56.53	56.21	51.47
	0-20dB	5.31	6.68	5.97	6.48	6.11	7.63	7.07	6.24	6.89	6.96	7.36	8.49	7.93	6.81

Table 2: WERs (in %) for test sets A - C obtained with a 3-layer uni-directional hybrid RCN-HMM and MFCC as the input features.

	SNR	TESTA					TESTB					TESTC			All
		Subway	Babble	Car	Exhib.	Avg.	Restaurant	Street	Airport	Station	Avg.	Subway	Street	Avg.	Avg.
Clean	Clean	1.60	1.66	1.52	1.23	1.50	1.60	1.66	1.52	1.23	1.50	1.50	1.69	1.60	1.52
	SNR20	2.49	2.36	2.36	2.62	2.45	2.21	2.75	2.33	2.07	2.34	2.76	2.99	2.88	2.49
	SNR15	3.93	3.33	3.34	4.81	3.85	3.19	3.33	3.46	3.24	3.31	4.45	4.29	4.37	3.74
	SNR10	7.34	6.41	5.82	9.53	7.26	5.68	6.14	5.25	5.71	5.69	8.47	7.47	7.97	6.77
	SNR5	13.48	12.30	11.39	19.78	14.20	13.29	12.67	11.57	12.77	12.57	17.68	14.75	16.21	13.95
	SNR0	28.52	30.96	26.13	38.35	30.94	33.07	28.93	26.48	28.69	29.27	37.46	31.74	34.58	31.00
	SNR-5	56.89	64.60	53.77	63.71	59.72	66.66	55.44	56.58	55.11	58.42	64.63	59.52	62.06	59.67
	0-20dB	11.15	11.07	9.81	15.02	11.74	11.49	10.76	9.82	10.50	10.64	14.16	12.25	13.20	11.59
Multi	Clean	2.21	2.33	2.09	1.91	2.14	2.21	2.33	2.09	1.91	2.14	2.33	2.36	2.35	2.18
	SNR20	1.96	2.21	2.09	1.73	2.00	2.21	2.36	2.27	1.54	2.10	2.06	2.75	2.41	2.12
	SNR15	2.39	2.66	2.71	2.34	2.53	2.52	2.75	2.86	2.62	2.69	2.67	3.63	3.15	2.72
	SNR10	3.59	3.66	3.88	4.75	3.97	3.87	4.17	4.06	3.83	3.98	3.99	5.05	4.52	4.08
	SNR5	7.22	7.44	7.01	10.06	7.92	8.32	9.67	7.96	8.64	8.65	8.87	10.34	9.61	8.55
	SNR0	16.70	21.19	18.31	21.81	19.50	23.03	21.67	19.68	21.51	21.46	22.47	24.61	23.55	21.09
	SNR-5	42.00	54.08	43.75	46.07	46.49	56.09	48.46	46.88	47.24	49.65	51.52	52.24	51.88	48.83
	0-20dB	6.37	7.43	6.80	8.14	7.18	7.99	8.12	7.37	7.63	7.78	8.01	9.28	8.65	7.71

Table 3: WERs (in %) for test sets A - C obtained with a 1-layer bi-directional hybrid RCN-HMM and MFCC as the input features.

	SNR	TESTA					TESTB					TESTC			All
		Subway	Babble	Car	Exhib.	Avg.	Restaurant	Street	Airport	Station	Avg.	Subway	Street	Avg.	Avg.
Clean	Clean	0.89	1.09	0.98	0.56	0.88	0.89	1.09	0.98	0.56	0.88	0.98	1.24	1.11	0.93
	SNR20	1.50	1.75	1.76	1.67	1.67	1.57	1.78	1.79	1.08	1.56	2.09	2.33	2.21	1.73
	SNR15	3.29	2.60	2.92	3.61	3.10	2.27	2.60	2.80	2.62	2.58	3.81	3.63	3.72	3.02
	SNR10	6.05	4.99	4.77	8.11	5.97	4.88	5.17	4.35	4.75	4.79	7.89	6.80	7.34	5.77
	SNR5	12.16	10.52	11.36	17.86	12.95	11.51	11.67	10.23	11.88	11.32	17.22	13.78	15.49	12.80
	SNR0	27.20	29.96	25.86	37.30	30.04	31.75	27.75	26.07	28.42	28.47	37.61	31.59	34.58	30.32
	SNR-5	56.65	64.15	53.89	61.93	59.13	65.80	55.89	56.16	54.52	58.07	67.12	61.43	64.25	59.73
	0-20dB	10.04	9.96	9.33	13.71	10.75	10.40	9.79	9.05	9.75	9.74	13.72	11.63	12.67	10.73
Multi	Clean	1.41	1.60	1.52	1.23	1.44	1.41	1.60	1.52	1.23	1.44	1.23	1.81	1.52	1.46
	SNR20	1.11	1.51	1.19	0.77	1.15	1.29	1.60	1.40	0.83	1.28	1.17	1.75	1.46	1.26
	SNR15	1.20	1.72	1.70	1.39	1.50	1.50	2.06	1.88	1.85	1.82	1.44	2.30	1.87	1.70
	SNR10	2.39	2.78	2.83	2.96	2.74	2.58	3.23	2.54	2.47	2.71	2.67	4.02	3.35	2.85
	SNR5	5.07	5.93	5.82	7.44	6.06	6.48	7.19	6.35	7.25	6.82	6.54	8.28	7.42	6.64
	SNR0	14.15	19.32	16.02	18.14	16.91	21.28	18.20	17.51	18.85	18.95	19.47	22.16	20.82	18.51
	SNR-5	38.93	52.51	40.11	42.83	43.61	54.74	45.77	46.23	43.97	47.66	49.31	50.70	50.01	46.51
	0-20dB	4.78	6.25	5.51	6.14	5.67	6.63	6.46	5.94	6.25	6.32	6.26	7.70	6.98	6.19

Table 4: WERs (in %) for test sets A - C obtained with a 3-layer bi-directional hybrid RCN-HMM and MFCC as the input features.

	SNR	TESTA					TESTB					TESTC			All
		Subway	Babble	Car	Exhib.	Avg.	Restaurant	Street	Airport	Station	Avg.	Subway	Street	Avg.	Avg.
Clean	Clean	1.26	1.45	1.10	0.86	1.17	1.26	1.45	1.10	0.86	1.17	1.29	1.42	1.36	1.21
	SNR20	2.24	1.45	1.61	2.31	1.90	1.96	1.75	1.46	1.76	1.73	2.18	1.87	2.03	1.86
	SNR15	3.56	2.21	2.48	3.89	3.02	2.92	2.60	2.24	2.47	2.55	4.08	2.90	3.49	2.93
	SNR10	7.22	4.53	5.01	8.76	6.36	5.80	5.83	4.15	5.00	5.19	8.44	6.71	7.57	6.13
	SNR5	19.13	13.51	10.92	21.20	16.13	15.72	16.26	10.53	13.27	13.93	21.15	15.93	18.52	15.73
	SNR0	42.28	44.14	26.69	45.23	39.50	45.23	36.91	32.27	31.84	36.54	45.23	38.69	41.93	38.80
	SNR-5	72.55	86.91	59.17	74.58	73.25	84.92	69.50	69.46	63.68	71.87	73.35	70.16	71.74	72.40
	0-20dB	14.89	13.17	9.34	16.28	13.38	14.33	12.67	10.13	10.87	11.99	16.22	13.22	14.71	13.09
Multi	Clean	1.54	1.69	1.70	1.20	1.54	1.54	1.69	1.70	1.20	1.54	1.63	1.69	1.66	1.56
	SNR20	2.12	1.66	1.67	1.73	1.79	2.06	1.75	1.85	1.27	1.73	2.27	1.84	2.06	1.82
	SNR15	2.64	1.96	2.15	2.50	2.31	2.30	2.15	2.15	2.04	2.16	3.13	2.12	2.62	2.31
	SNR10	4.64	2.87	3.22	3.76	3.62	3.44	3.72	2.68	3.27	3.28	4.61	3.84	4.22	3.60
	SNR5	8.38	6.92	7.10	8.73	7.77	7.15	9.13	5.91	7.50	7.42	9.67	8.74	9.20	7.92
	SNR0	21.68	21.98	21.00	20.55	21.30	21.86	24.18	18.22	20.70	21.23	23.49	25.42	24.46	21.90
	SNR-5	51.86	54.56	55.89	48.84	52.82	56.03	57.29	49.60	53.63	54.12	53.55	59.10	56.34	54.04
	0-20dB	7.89	7.08	7.03	7.45	7.36	7.36	8.19	6.16	6.96	7.16	8.63	8.39	8.51	7.51

Table 5: WERs (in %) for test sets A - C obtained with a 1-layer uni-directional hybrid RCN-HMM and MelFB as the input features.

	SNR	TESTA					TESTB					TESTC			All
		Subway	Babble	Car	Exhib.	Avg.	Restaurant	Street	Airport	Station	Avg.	Subway	Street	Avg.	Avg.
Clean	Clean	0.58	0.79	0.72	0.37	0.62	0.58	0.79	0.72	0.37	0.62	0.55	0.88	0.72	0.64
	SNR20	1.81	1.30	1.13	1.39	1.41	1.20	1.66	1.22	1.17	1.31	1.69	1.69	1.69	1.43
	SNR15	2.79	1.75	1.61	2.75	2.22	2.12	2.69	1.58	2.38	2.19	3.19	2.42	2.80	2.32
	SNR10	5.71	4.29	4.15	7.00	5.27	4.45	5.17	3.22	3.67	4.13	6.97	5.11	6.03	4.97
	SNR5	16.12	11.52	10.29	18.82	14.14	13.60	14.18	9.22	11.79	12.18	17.90	13.91	15.89	13.71
	SNR0	38.50	40.24	26.87	40.70	36.51	43.75	37.79	30.39	31.16	35.75	41.69	37.70	39.68	36.84
	SNR-5	71.48	85.55	59.02	75.22	72.76	83.91	71.52	70.24	64.30	72.48	71.48	71.58	71.53	72.40
	0-20dB	12.99	11.82	8.81	14.13	11.91	13.02	12.30	9.13	10.03	11.11	14.29	12.17	13.22	11.85
Multi	Clean	1.14	1.12	1.10	0.56	0.98	1.14	1.12	1.10	0.56	0.98	1.14	1.15	1.14	1.01
	SNR20	1.38	1.09	0.95	0.71	1.03	1.07	1.15	1.10	0.77	1.03	1.38	1.24	1.31	1.09
	SNR15	1.78	1.30	1.28	1.17	1.38	1.20	1.69	1.22	1.14	1.31	1.90	1.57	1.74	1.42
	SNR10	2.95	1.87	2.54	2.38	2.43	2.21	2.45	1.67	1.85	2.04	3.13	2.75	2.94	2.38
	SNR5	6.36	4.87	4.83	6.60	5.65	5.74	7.16	4.35	6.02	5.81	6.85	6.74	6.79	5.94
	SNR0	18.88	19.32	16.52	16.85	17.89	19.90	20.95	15.81	18.42	18.76	19.44	21.58	20.52	18.76
	SNR-5	48.17	52.72	49.99	43.94	48.73	54.13	53.30	48.67	50.54	51.65	48.73	55.41	52.09	50.57
	0-20dB	6.27	5.69	5.22	5.54	5.68	6.02	6.68	4.83	5.64	5.79	6.54	6.78	6.66	5.92

Table 6: WERs (in %) for test sets A - C obtained with a 3-layer uni-directional hybrid RCN-HMM and MelFB as the input features.

	SNR	TESTA					TESTB					TESTC			All
		Subway	Babble	Car	Exhib.	Avg.	Restaurant	Street	Airport	Station	Avg.	Subway	Street	Avg.	Avg.
Clean	Clean	1.29	1.57	1.19	0.99	1.26	1.29	1.57	1.19	0.99	1.26	1.29	1.51	1.40	1.29
	SNR20	2.49	1.72	1.91	2.59	2.17	1.96	2.36	1.85	2.07	2.06	2.33	2.24	2.28	2.15
	SNR15	4.08	2.78	2.65	4.44	3.48	2.52	3.02	2.39	3.02	2.74	4.94	2.93	3.93	3.27
	SNR10	7.89	4.78	5.49	9.60	6.92	5.50	6.29	4.24	5.09	5.27	9.43	6.77	8.09	6.49
	SNR5	19.71	12.70	12.08	22.15	16.60	14.43	14.90	9.69	12.03	12.75	20.94	16.05	18.48	15.43
	SNR0	39.79	37.73	27.89	44.34	37.36	38.87	34.25	27.97	29.03	32.51	43.11	35.58	39.31	35.81
	SNR-5	69.30	77.24	59.17	75.16	70.16	76.97	64.78	63.41	58.81	65.98	71.26	66.17	68.70	68.20
	0-20dB	14.79	11.94	10.00	16.62	13.31	12.66	12.16	9.23	10.25	11.07	16.15	12.71	14.42	12.63
Multi	Clean	1.93	1.69	1.70	1.33	1.66	1.93	1.69	1.70	1.33	1.66	1.78	1.81	1.80	1.69
	SNR20	2.12	1.75	1.91	1.64	1.85	1.81	2.06	1.82	1.36	1.76	2.39	2.00	2.19	1.88
	SNR15	2.55	2.21	2.30	2.28	2.33	2.09	2.33	2.09	1.94	2.11	2.89	2.30	2.59	2.29
	SNR10	4.27	3.11	2.77	3.86	3.50	3.35	3.75	2.80	3.09	3.24	4.11	3.60	3.85	3.47
	SNR5	8.90	6.59	6.41	8.58	7.61	6.82	8.80	5.43	7.10	7.03	9.15	8.19	8.67	7.59
	SNR0	20.42	19.89	18.37	19.38	19.51	19.65	22.52	16.94	18.14	19.31	21.65	23.04	22.35	20.00
	SNR-5	47.47	51.27	47.75	45.48	48.01	52.35	53.42	45.63	47.18	49.63	49.06	54.26	51.68	49.39
	0-20dB	7.65	6.71	6.35	7.15	6.96	6.74	7.89	5.82	6.33	6.69	8.04	7.83	7.93	7.05

Table 7: WERs (in %) for test sets A - C obtained with a 1-layer bi-directional hybrid RCN-HMM and MelFB as the input features.

	SNR	TESTA					TESTB					TESTC			All
		Subway	Babble	Car	Exhib.	Avg.	Restaurant	Street	Airport	Station	Avg.	Subway	Street	Avg.	Avg.
Clean	Clean	0.68	0.88	0.69	0.49	0.68	0.68	0.88	0.69	0.49	0.68	0.61	0.88	0.75	0.69
	SNR20	1.57	1.30	1.37	1.79	1.50	1.07	1.63	1.28	1.08	1.27	1.57	1.66	1.61	1.43
	SNR15	2.27	2.21	2.56	3.18	2.55	2.03	2.30	2.06	1.76	2.04	3.16	2.27	2.71	2.38
	SNR10	5.68	4.02	4.32	7.37	5.33	4.11	5.11	3.43	3.39	4.01	6.72	5.38	6.05	4.95
	SNR5	15.20	11.15	10.05	18.88	13.78	12.04	12.88	8.08	10.15	10.78	17.72	13.15	15.42	12.91
	SNR0	35.55	35.34	26.13	41.65	34.60	36.41	32.53	26.27	27.71	30.71	39.24	33.01	36.10	33.34
	SNR-5	66.56	75.76	57.02	73.50	68.15	73.41	63.24	62.54	57.70	64.21	67.70	64.36	66.02	66.15
	0-20dB	12.05	10.80	8.89	14.57	11.55	11.13	10.89	8.22	8.82	9.76	13.68	11.09	12.38	11.00
Multi	Clean	1.04	1.18	1.01	0.77	1.00	1.04	1.18	1.01	0.77	1.00	1.04	1.15	1.10	1.02
	SNR20	1.20	1.15	1.13	0.77	1.06	0.92	1.12	1.37	0.62	1.01	1.20	1.27	1.23	1.07
	SNR15	1.47	1.18	1.43	1.02	1.28	1.26	1.39	1.58	1.08	1.33	1.72	1.51	1.61	1.37
	SNR10	2.49	2.00	2.27	2.41	2.29	2.24	2.36	2.09	1.76	2.11	2.86	2.63	2.74	2.31
	SNR5	5.65	4.84	4.47	6.60	5.38	5.28	6.89	4.56	4.94	5.42	6.26	6.77	6.52	5.62
	SNR0	16.67	16.72	13.75	15.21	15.58	18.21	18.74	14.94	15.43	16.82	16.83	19.38	18.11	16.58
	SNR-5	42.89	48.25	41.90	38.78	42.97	49.34	48.76	42.80	42.49	45.84	44.52	49.61	47.08	44.94
	0-20dB	5.50	5.18	4.61	5.20	5.12	5.58	6.10	4.91	4.77	5.34	5.77	6.31	6.04	5.39

Table 8: WERs (in %) for test sets A - C obtained with a 3-layer bi-directional hybrid RCN-HMM and MelFB as the input features.

	SNR	TESTA					TESTB					TESTC			All
		Subway	Babble	Car	Exhib.	Avg.	Restaurant	Street	Airport	Station	Avg.	Subway	Street	Avg.	Avg.
Clean	Clean	1.14	1.45	1.28	1.27	1.28	1.14	1.45	1.28	1.27	1.28	1.17	1.36	1.26	1.28
	SNR20	2.18	2.06	1.76	2.01	2.00	2.58	2.51	2.12	1.76	2.24	2.06	2.24	2.15	2.13
	SNR15	3.68	3.17	2.30	3.49	3.15	3.41	3.11	2.83	2.75	3.02	3.62	3.54	3.58	3.18
	SNR10	7.00	5.59	4.38	5.74	5.67	6.17	5.14	4.59	4.91	5.20	6.75	5.23	5.99	5.55
	SNR5	13.29	13.06	8.17	13.73	12.04	14.18	11.97	9.72	10.77	11.65	13.97	11.28	12.61	12.00
	SNR0	30.15	33.89	21.23	30.61	28.93	33.16	26.63	24.63	25.08	27.36	33.01	27.84	30.40	28.60
	SNR-5	59.23	66.66	47.90	60.97	58.64	67.45	54.14	56.28	50.66	57.12	58.80	54.41	56.59	57.62
	0-20dB	11.26	11.55	7.57	11.12	10.36	11.90	9.87	8.78	9.05	9.89	11.88	10.03	10.95	10.29
Multi	Clean	1.60	1.90	2.12	1.82	1.86	1.60	1.90	2.12	1.82	1.86	1.81	1.93	1.87	1.86
	SNR20	2.06	2.00	1.76	1.73	1.88	2.06	2.33	2.09	1.76	2.06	1.81	2.33	2.07	1.99
	SNR15	2.55	2.54	2.45	2.53	2.52	2.95	3.08	2.33	2.87	2.80	2.98	3.17	3.08	2.74
	SNR10	5.34	3.63	3.91	4.38	4.31	4.27	4.50	3.82	3.86	4.11	4.42	4.69	4.55	4.28
	SNR5	8.20	7.92	5.49	8.39	7.49	8.93	9.01	6.71	7.59	8.06	8.44	8.89	8.67	7.95
	SNR0	19.44	23.31	15.57	19.35	19.40	23.76	21.37	18.16	19.19	20.61	20.85	22.97	21.92	20.39
	SNR-5	48.33	55.53	42.38	45.48	47.92	55.76	49.09	45.66	44.99	48.86	48.94	49.12	49.03	48.52
	0-20dB	7.52	7.88	5.84	7.28	7.12	8.39	8.06	6.62	7.05	7.53	7.70	8.41	8.06	7.47

Table 9: WERs (in %) for test sets A - C obtained with a 1-layer uni-directional hybrid RCN-HMM and AFE as the input features.

	SNR	TESTA					TESTB					TESTC			All
		Subway	Babble	Car	Exhib.	Avg.	Restaurant	Street	Airport	Station	Avg.	Subway	Street	Avg.	Avg.
Clean	Clean	0.95	0.91	0.66	0.62	0.78	0.95	0.91	0.66	0.62	0.78	0.74	1.09	0.91	0.81
	SNR20	1.41	1.36	1.19	1.23	1.30	1.72	1.66	1.49	0.96	1.46	1.44	1.63	1.54	1.41
	SNR15	2.76	2.54	1.70	2.47	2.36	2.89	2.51	2.48	2.07	2.48	3.13	2.69	2.91	2.52
	SNR10	6.88	4.63	3.73	4.60	4.95	5.53	4.53	4.15	3.89	4.52	6.42	5.05	5.73	4.93
	SNR5	12.50	12.88	8.35	12.47	11.53	13.45	11.15	9.63	10.00	11.05	13.60	11.37	12.48	11.53
	SNR0	30.12	34.58	21.89	30.11	29.14	34.48	26.81	26.10	25.61	28.23	32.58	27.36	29.95	28.94
	SNR-5	59.56	67.65	49.84	60.57	59.37	67.24	55.53	58.25	52.36	58.34	59.69	55.56	57.61	58.61
	0-20dB	10.73	11.20	7.37	10.18	9.86	11.61	9.33	8.77	8.51	9.55	11.43	9.62	10.52	9.87
Multi	Clean	1.20	1.45	1.10	1.23	1.25	1.20	1.45	1.10	1.23	1.25	1.23	1.63	1.43	1.29
	SNR20	1.14	1.39	1.13	0.83	1.12	1.32	1.69	1.31	0.93	1.31	0.98	1.54	1.26	1.22
	SNR15	1.63	1.66	1.49	1.27	1.51	2.06	1.81	1.94	1.85	1.92	1.50	2.03	1.77	1.73
	SNR10	3.32	2.75	2.86	3.02	2.99	3.01	3.45	2.86	2.44	2.94	2.98	3.72	3.35	3.04
	SNR5	6.14	6.35	4.77	7.00	6.06	7.46	7.71	5.64	6.48	6.82	6.63	7.74	7.19	6.59
	SNR0	16.24	20.89	13.75	17.71	17.14	21.61	19.95	16.88	17.68	19.02	18.15	19.98	19.07	18.28
	SNR-5	46.30	54.75	41.37	44.37	46.69	55.73	47.64	45.30	43.97	48.14	47.01	48.07	47.54	47.44
	0-20dB	5.69	6.61	4.80	5.97	5.76	7.09	6.92	5.73	5.88	6.40	6.05	7.00	6.53	6.17

Table 10: WERs (in %) for test sets A - C obtained with a 3-layer uni-directional hybrid RCN-HMM and AFE as the input features.

	SNR	TESTA					TESTB					TESTC			All
		Subway	Babble	Car	Exhib.	Avg.	Restaurant	Street	Airport	Station	Avg.	Subway	Street	Avg.	Avg.
Clean	Clean	1.44	1.75	1.70	1.42	1.58	1.44	1.75	1.70	1.42	1.58	1.60	1.66	1.63	1.59
	SNR20	2.21	2.24	2.21	2.07	2.18	2.49	2.60	2.03	2.01	2.28	2.09	2.66	2.38	2.26
	SNR15	3.35	3.14	2.68	3.39	3.14	3.47	3.14	3.01	3.27	3.22	3.56	3.42	3.49	3.24
	SNR10	6.69	5.14	4.41	5.99	5.55	5.89	5.53	4.86	4.63	5.23	6.05	4.96	5.50	5.41
	SNR5	12.53	12.18	8.05	13.42	11.52	12.40	10.79	8.98	10.15	10.57	12.07	10.97	11.52	11.14
	SNR0	26.65	30.96	19.77	28.82	26.51	29.32	23.40	23.41	22.52	24.65	28.68	25.45	27.05	25.87
	SNR-5	55.05	61.64	45.78	58.22	55.13	61.01	50.15	53.15	47.98	53.07	55.30	51.42	53.34	53.95
	0-20dB	10.29	10.73	7.42	10.74	9.78	10.71	9.09	8.46	8.52	9.19	10.49	9.49	9.99	9.59
Multi	Clean	1.90	2.63	1.97	2.25	2.19	1.90	2.63	1.97	2.25	2.19	1.96	2.60	2.28	2.21
	SNR20	2.00	2.30	2.12	1.91	2.08	2.55	2.72	2.27	1.82	2.34	1.87	2.78	2.33	2.23
	SNR15	2.39	2.24	2.42	2.62	2.42	3.10	2.93	2.59	2.65	2.82	2.61	3.33	2.97	2.69
	SNR10	4.11	3.93	3.79	4.07	3.97	4.05	4.38	3.52	3.30	3.81	3.84	4.75	4.30	3.97
	SNR5	7.34	7.41	5.99	8.42	7.28	8.01	8.86	6.41	7.44	7.68	7.71	8.65	8.18	7.62
	SNR0	17.01	20.77	13.81	18.73	17.56	20.57	18.68	16.43	17.37	18.25	18.36	20.28	19.33	18.19
	SNR-5	42.37	52.06	38.29	42.80	43.87	51.27	45.04	42.47	40.91	44.91	44.30	46.13	45.22	44.56
	0-20dB	6.57	7.33	5.63	7.15	6.66	7.66	7.51	6.24	6.52	6.98	6.88	7.96	7.42	6.94

Table 11: WERs (in %) for test sets A - C obtained with a 1-layer bi-directional hybrid RCN-HMM and AFE as the input features.

	SNR	TESTA					TESTB					TESTC			All
		Subway	Babble	Car	Exhib.	Avg.	Restaurant	Street	Airport	Station	Avg.	Subway	Street	Avg.	Avg.
Clean	Clean	0.77	0.97	0.81	0.74	0.82	0.77	0.97	0.81	0.74	0.82	0.89	0.97	0.93	0.84
	SNR20	1.78	1.75	1.52	1.51	1.64	1.72	2.12	1.61	1.17	1.66	1.57	1.93	1.75	1.67
	SNR15	2.61	2.21	2.09	2.16	2.26	2.82	2.18	2.51	2.07	2.39	2.79	2.66	2.73	2.41
	SNR10	5.68	4.47	3.76	5.25	4.78	4.79	4.56	3.88	3.55	4.19	5.56	4.69	5.12	4.61
	SNR5	11.45	10.97	7.72	12.28	10.59	11.33	9.61	8.56	8.49	9.49	11.51	10.40	10.95	10.22
	SNR0	26.28	30.99	19.42	27.65	26.05	28.65	23.04	23.65	22.12	24.36	28.31	25.63	26.96	25.56
	SNR-5	55.51	61.94	46.97	57.76	55.51	60.67	51.33	53.18	48.16	53.33	56.13	52.36	54.23	54.38
	0-20dB	9.56	10.08	6.90	9.77	9.06	9.86	8.30	8.04	7.48	8.42	9.95	9.06	9.50	8.89
Multi	Clean	1.38	1.72	1.22	1.14	1.37	1.38	1.72	1.22	1.14	1.37	1.44	1.63	1.54	1.40
	SNR20	1.20	1.63	1.31	1.05	1.30	1.63	1.93	1.43	1.11	1.53	1.14	1.81	1.48	1.43
	SNR15	1.66	1.81	1.82	1.45	1.69	2.18	2.30	1.97	1.57	2.01	1.90	2.36	2.13	1.91
	SNR10	2.92	2.60	2.86	2.75	2.78	3.01	3.69	2.83	2.44	2.99	2.76	3.69	3.23	2.95
	SNR5	5.56	5.62	5.01	6.88	5.76	6.48	7.07	5.40	5.71	6.16	5.80	7.62	6.72	6.11
	SNR0	14.43	18.65	12.68	16.48	15.55	18.94	17.74	15.21	15.12	16.75	15.23	18.14	16.69	16.26
	SNR-5	40.53	50.63	37.40	41.78	42.58	49.92	43.62	41.72	39.90	43.78	42.28	44.71	43.50	43.24
	0-20dB	5.15	6.06	4.74	5.72	5.42	6.45	6.55	5.37	5.19	5.89	5.37	6.72	6.05	5.73

Table 12: WERs (in %) for test sets A - C obtained with a 3-layer bi-directional hybrid RCN-HMM and AFE as the input features.

Bibliography

[1] A. Jalalvand, F. Triefenbach, D. Verstraeten, and J.-P. Martens, "Connected digit recognition by means of reservoir computing," in *Proc. Interspeech*, pp. 1725–1728, 2011.

[2] A. Jalalvand, F. Triefenbach, and J.-P. Martens, "Continuous digit recognition in noise: Reservoirs can do an excellent job!," in *Proc. Interspeech*, p. ID:644, 2012.

[3] A. Jalalvand, F. Triefenbach, K. Demuynck, and J.-P. Martens, "Robust continuous digit recognition using reservoir computing," *Computer Speech and Language*, vol. 30, no. 1, pp. 135 – 158, 2015.

[4] A. Jalalvand, K. Demuynck, and J.-P. Martens, "Noise robust continuous digit recognition with reservoir-based acoustic models," in *Proc. ISPACS*, p. ID:99, 2013.

[5] A. Jalalvand, K. Demuynck, and J.-P. Martens, "Feature enhancement with a reservoir-based denoising auto encoder," in *International Symposium on Signal Processing and Information Technology, Proceedings*, p. 6, Proc. ISSPIT, 2013.

[6] A. Jalalvand, W. De Neve, R. Van de Walle, and J.-P. Martens, "Noise robust handwritten digit recognition with reservoir computing networks," *Journal of Machine Learning Research*, submitted, 2014.

[7] A. Jalalvand, G. Van Wallendael, and R. Van de Walle, "Reservoir computing networks for detection tasks on (motion) pictures," *IAPR Intl. Conference on Machine Vision Applications*, submitted, 2015.

[8] F. Triefenbach, A. Jalalvand, B. Schrauwen, and J.-P. Martens, "Phoneme recognition with large hierarchical reservoirs," in *Proc. NIPS*, pp. 2307–2315, 2010.

[9] F. Triefenbach, A. Jalalvand, K. Demuynck, and J.-P. Martens, "Acoustic modeling with hierarchical reservoirs," *IEEE Trans. Audio, Speech and Language Processing*, vol. 21, no. 11, pp. 2439–2450, 2013.

[10] F. Triefenbach, A. Jalalvand, K. Demuynck, and J.-P. Martens, "Context-dependent modeling and speaker normalization applied to reservoir-based phone recognition," in *Proc. Interspeech*, pp. 3342–3346, 2013.

[11] C. Lee, F. Soong, and K. Paliwal, *Automatic Speech and Speaker Recognition: Advanced Topics*. Kluwer International series in engineering and computer science: VLSI, computer architecture and digital signal processing, Springer, 1996.

[12] R. Pieraccini, *The Voice in the Machine: Building Computers That Understand Speech*. The MIT Press, Mar. 2012.

[13] E. Reiter, "Natural language generation," *The Handbook of Computational Linguistics and Natural Language Processing*, pp. 574–598, 2010.

[14] C. Henton, *The Encyclopedia of Applied Linguistics*, ch. Text-to-Speech Synthesis Development. Blackwell Publishing Ltd, 2012.

[15] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," *Pattern recognition and artificial intelligence*, vol. 116, pp. 374–388, 1976.

[16] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 28, pp. 357–366, aug 1980.

[17] J. H. Hansen, "Analysis and compensation of speech under stress and noise for environmental robustness in speech recognition," *Speech Communication*, vol. 20, pp. 151–173, 1996.

[18] H. Hermansky and N. Morgan, "RASTA processing of speech.," *IEEE Trans. Speech and Audio Processing*, vol. 2, no. 4, pp. 578–589, 1994.

[19] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *Journal of the Acoustical Society of America*, vol. 57, pp. 1738–1752, Apr. 1990.

[20] S. Ikbal, H. Misra, H. Hermansky, and M. Magimai-Doss, "Phase autocorrelation (PAC) features for noise robust speech recognition," *Speech Communication*, vol. 54, pp. 867–880, Sept. 2012.

[21] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 27, no. 2, pp. 113–120, 1979.

[22] M. Berouti, R. Schwartz, and J. Makhoul, "Enhancement of speech corrupted by acoustic noise," in *Proc. ICASSP*, vol. 4, pp. 208–211, 1979.

[23] C. Ris and S. Dupont, "Assessing local noise level estimation methods: Application to noise robust ASR," *Speech Communication*, vol. 34, no. 12, pp. 141–158, 2001.

[24] G. Lathoud, M. Magimai-Doss, and H. Bourlard, "Unsupervised spectral subtraction for noise-robust asr," in *Proc. ASRU*, pp. 189–194, 2005.

[25] P. Jain and H. Hermansky, "Improved mean and variance normalization for robust speech recognition," in *Proc. ICASSP*, 2001.

[26] X. Xiao, J. Li, E.-S. Chng, H. Li, and C.-H. Lee, "A study on the generalization capability of acoustic models for robust speech recognition," *IEEE Trans. Audio, Speech and Language Processing*, vol. 18, no. 6, pp. 1158–1169, 2010.

[27] S. Molau, M. Pitz, and H. Ney, "Histogram based normalization in the acoustic feature space," in *Proc. ASRU*, pp. 21–24, 2001.

[28] T. Kai, M. Suzuki, and K. Chijiiwa, "Combination of SPLICE and feature normalization for noise robust speech recognition," *Intl. Workshop on Nonlinear Circuits, Communications and Signal Processing*, vol. 16, no. 4, pp. 323–326, 2012.

[29] H.-G. Hirsch and D. Pearce, "The AURORA experimental framework for the performance evaluation of speech recognition systems under noise conditions," in *Automatic Speech Recognition: Challenges for the Next Millennium*, pp. 181–188, ISCA ITRW, 2000.

[30] R. Lippmann, "An introduction to computing with neural nets," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 4, no. 2, pp. 4–22, 1987.

[31] M. Cooke, P. Green, L. Josifovski, and A. Vizinho, "Robust automatic speech recognition with missing and unreliable acoustic data," *Speech Communication*, vol. 34, no. 3, pp. 267–285, 2001.

[32] C. Leggetter and P. Woodland", "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," *Computer Speech and Language*, vol. 9, no. 2, pp. 171–185, 1995.

[33] M. Gales and P. Woodland, "Mean and variance adaptation within the MLLR framework," *Computer Speech and Language*, vol. 10, no. 4, pp. 249–264, 1996.

[34] V. Digalakis, D. Rtischev, and L. Neumeyer, "Speaker adaptation using constrained estimation of gaussian mixtures," *IEEE Trans. Speech and Audio Processing*, vol. 3, pp. 357–366, Sep 1995.

[35] P. Woodland, "Speaker adaptation for continuous density HMMs: A review," in *Proc. ISCA Workshop on Adaptation Methods for Speech Recognition*, pp. 11–19, 2001.

[36] H. Liao and M. J. F. Gales, "Issues with uncertainty decoding for noise robust automatic speech recognition," in *Speech Communication*, vol. 50, pp. 265–277, 2008.

[37] N. Cressie, C. A. Calder, J. Clark, J. Ver Hoef, and C. Wikle, "Accounting for uncertainty in ecological analysis: The strengths and limitations of hierarchical statistical modeling - with discussion," *Ecological Applications*, vol. 19(3), pp. 553–570, 2009.

[38] ITU recommendation G.712, "Transmission performance characteristics of pulse code modulation channels," tech. rep., ITU, Nov. 1996.

[39] L. Deng and X. Li, "Machine learning paradigms for speech recognition: An overview," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 21, no. 5, pp. 1060–1089, 2013.

[40] H. Wit and R. Ritsma, "Stimulated acoustic emissions from the human ear," *Journal of the Acoustical Society of America*, vol. 66, no. 3, pp. 911–913, 1979.

[41] A. V. Oppenheim and S. R. W., *Digital signal processing*. Prentice-Hall, 1975.

[42] J. C. Brown, "Calculation of a constant Q spectral transform," *Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[43] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 28, pp. 357–366, Aug 1980.

[44] S. Furui, “Cepstral analysis technique for automatic speaker verification,” *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 29, no. 2, pp. 254–272, 1981.

[45] S. Furui, “Speaker-independent isolated word recognition using dynamic features of speech spectrum,” *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 34, no. 1, pp. 52–59, 1986.

[46] H.-L. Lou, “Implementing the viterbi algorithm,” *IEEE Signal Processing Magazine*, vol. 12, pp. 42–52, Sep 1995.

[47] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[48] L. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proc. IEEE*, vol. 77, pp. 257–286, Feb 1989.

[49] L. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*. Prentice-Hall, 1993.

[50] M. Gales and S. Young, *The Application of Hidden Markov Models in Speech Recognition*. Foundations and trends in signal processing, Now Publishers, 2008.

[51] L. Liporace, “Maximum likelihood estimation for multivariate observations of markov sources,” *Information Theory, IEEE Transactions on*, vol. 28, no. 5, pp. 729–734, 1982.

[52] I. J. Myung, “Tutorial on maximum likelihood estimation,” *Journal of Mathematical Psychology*, vol. 47, no. 1, pp. 90–100, 2003.

[53] T. K. Moon, “The Expectation-Maximization Algorithm,” *IEEE Signal Processing Magazine*, vol. 13, pp. 47–60, Nov. 1996.

[54] H. Jiang, “Discriminative training of hmms for automatic speech recognition: A survey,” *Computer Speech and Language*, vol. 24, pp. 589–608, Oct 2010.

[55] G. Heigold, H. Ney, R. Schluter, and S. Wiesler, “Discriminative training for automatic speech recognition: Modeling, criteria, optimization, implementation, and performance,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 58–69, 2012.

[56] M. J. F. Gales, S. Watanabe, and E. Fosler-Lussier, “Structured discriminative models for speech recognition: An overview,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 70–81, 2012.

[57] L. Bahl, P. Brown, P. De Souza, and R. Mercer, “Maximum mutual information estimation of hidden markov model parameters for speech recognition,” in *Proc. ICASSP*, vol. 11, pp. 49–52, 1986.

[58] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland, *The HTK Book, version 3.4*. Cambridge, UK: Cambridge University Engineering Department, 2006.

[59] KU Leuven, “Speech Processing, Recognition and Automatic Annotation Kit SPRAAK v1.0,” Dec. 2010. `http://www.spraak.org`.

[60] K. J. Lang, A. H. Waibel, and G. E. Hinton, “A time-delay neural network architecture for isolated word recognition,” *Neural Networks*, vol. 3, no. 1, pp. 23–43, 1990.

[61] E. Trentin and M. Gori, “A survey of hybrid ann/hmm models for automatic speech recognition,” *Neurocomputing*, vol. 37, pp. 91–126, 2001.

[62] A. K. Jain, J. Mao, and K. M. Mohiuddin, “Artificial neural networks: A tutorial,” *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[63] B. Prasad and S. Prasanna, *Speech, Audio, Image and Biomedical Signal Processing Using Neural Networks*. Studies in Computational Intelligence, Springer, 2008.

[64] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, Nov 2012.

[65] M. Richard and R. Lippmann, “Neural network classifiers estimate posterior probabilities,” *Neural Computation*, vol. 3, pp. 461–483, Winter 1991.

[66] H. Bourlard and C. J. Wellekens, “Links between markov models and multilayer perceptrons,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, pp. 1167–1178, Dec 1990.

[67] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain.,” *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[68] S. I. Gallant, “Perceptron-based learning algorithms,” *Neural Networks, IEEE Transactions on*, vol. 1, no. 2, pp. 179–191, 1990.

[69] D. E. Rumelhart, G. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, pp. 533–536, Oct 1986.

[70] M. Schumacher, R. Rossner, and W. Vach, “Neural networks and logistic regression: Part i,” *Computational Statistics and Data Analysis*, vol. 21, no. 6, pp. 661–682, 1996.

[71] C. Serrano-Cinca, W. Vach, R. Rossner, and M. Schumacher, “Neural networks and logistic regression: Part ii,” *Computational Statistics and Data Analysis*, vol. 21, no. 6, pp. 683–701, 1996.

[72] G. Sivaram and H. Hermansky, “Sparse multilayer perceptron for phoneme recognition,” *IEEE Trans. Audio, Speech and Language Processing*, vol. 20, pp. 23–29, Jan 2012.

[73] P. Schwarz, P. Matejka, and J. Cernocky, “Hierarchical structures of neural networks for phoneme recognition,” in *Proc. ICASSP*, pp. 325–328, 2006.

[74] J. Pinto, G. Sivaram, M. Magimai-Doss, H. Hermansky, and H. Bourlard, “Analysis of MLP-based hierarchical phoneme posterior probability estimator,” *IEEE Trans. Audio, Speech and Language Processing*, vol. 2, pp. 225–241, Feb 2011.

[75] L. Toth, “A hierarchical, context-dependent neural network architecture for improved phone recognition,” in *Proc. ICASSP*, pp. 5040–5043, 2011.

[76] A. Waibel, H. Sawai, and K. Shikano, “Modularity and scaling in large phonemic neural networks,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, pp. 1888 –1898, dec 1989.

[77] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, *Readings in speech recognition*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990.

[78] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1.* Cambridge, MA, USA: MIT Press, 1986.

[79] T. Robinson and F. Fallside, "A recurrent error propagation network speech recognition system," *Computer Speech and Language*, vol. 5, pp. 259–274, Jul 1991.

[80] H. Jaeger, "Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the echo state network approach," tech. rep., GMD Report 159, German National Research Center for Information Technology, 2002.

[81] P. Werbos, "Backpropagation through time: what it does and how to do it," *Proc. IEEE*, vol. 78, pp. 1550–1560, Oct 1990.

[82] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, Summer 1989.

[83] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Processing*, vol. 45, pp. 2673–2681, Nov 1997.

[84] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio, Speech and Language Processing*, vol. 20, pp. 14–22, Jan 2012.

[85] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, July 2006.

[86] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proc. ICML*, pp. 473–480, 2007.

[87] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, vol. 9, pp. 249–256, May 2010.

[88] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Computation*, vol. 22, pp. 3207–3220, Dec. 2010.

[89] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov 1997.

[90] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. PhD thesis, Technische Universität München, 2008.

[91] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Interspeech*, pp. 338–342, 2014.

[92] Y. Bengio, R. D. Mori, G. Flammia, and R. Kompe, "Global optimization of a neural network - hidden Markov model hybrid," *IEEE Trans. Neural Networks*, vol. 3, pp. 252–259, 1991.

[93] Y. Bengio, "A connectionist approach to speech recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 7, no. 4, pp. 647–668, 1993.

[94] F. Johansen, "A comparison of hybrid HMM architecture using global discriminating training," in *Proc. ICSLP*, vol. 1, pp. 498–501, 1996.

[95] P. Haffner, M. Franzini, and A. Waibel, "Integrating time alignment and neural networks for high performance continuous speech recognition," in *Proc. ICASSP*, pp. 105–108, 1991.

[96] C. Dugast, L. Devillers, and X. L. Aubert, "Combining TDNN and HMM in a hybrid system for improved continuous-speech recognition.," *IEEE Trans. Speech and Audio Processing*, vol. 2, no. 1, pp. 217–223, 1994.

[97] S. Austin, G. Zavaliagkos, J. Makhoul, and R. Schwartz, "A hybrid continuous speech recognition system using segmental neural nets with hidden Markov models," in *Proc. NNSP*, pp. 347–356, 1991.

[98] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *IEEE Trans. Neural Networks*, vol. 20, pp. 391–403, 2007.

[99] H. Jaeger, "The ''echo state'' approach to analysing and training recurrent neural networks - with an erratum note," tech. rep., GMD Report 148, German National Research Center for Information Technology, 2001.

[100] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky-integrator neurons," *Neural Networks*, vol. 20, pp. 335–352, Apr 2007.

[101] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, 2006.

[102] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, “Extreme learning machine for regression and multiclass classification,” *IEEE Trans. Syst., Man, Cybern. B*, vol. 42, pp. 513–529, April 2012.

[103] E. Cambria, G.-B. Huang, L. L. C. Kasun, H. Zhou, C. M. Vong, J. Lin, J. Yin, Z. Cai, Q. Liu, K. Li, V. C. Leung, L. Feng, Y.-S. Ong, M.-H. Lim, A. Akusok, A. Lendasse, F. Corona, R. Nian, Y. Miche, P. Gastaldo, R. Zunino, S. Decherchi, X. Yang, K. Mao, B.-S. Oh, J. Jeon, K.-A. Toh, A. B. J. Teoh, J. Kim, H. Yu, Y. Chen, and J. Liu, “Extreme learning machines [trends controversies],” *IEEE Intelligent Systems*, vol. 28, pp. 30–59, Nov 2013.

[104] D. Serre, *Matrices: theory and applications*. Springer, 2002.

[105] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, pp. 273–297, 1995.

[106] A. Robinson, “An application of recurrent neural nets to phone probability estimation,” *IEEE Trans. Neural Networks*, vol. 5, pp. 298–305, 1994.

[107] Q. Zhi-yi, L. Yu, Z. Li-hong, and S. Ming-xin, “A speech recognition system based on a hybrid HMM/SVM architecture,” in *Proc. International Conference on Innovative Computing, Information and Control*, pp. 100–104, 2006.

[108] D. Verstraeten, B. Schrauwen, and D. Stroobandt, “Isolated word recognition using a Liquid State Machine,” in *Proc. ESANN*, pp. 435–440, 2005.

[109] M. Skowronski and J. Harris, “Automatic speech recognition using a predictive echo state network classifier,” *IEEE Trans. Neural Networks*, vol. 20, no. 3, pp. 414–423, 2007.

[110] D. W. Marquardt and R. D. Snee, “Ridge regression in practice,” *The American Statistician*, vol. 29, no. 1, pp. 3–20, 1975.

[111] E. H. C. Choi, “A generalized framework for compensation of mel-filterbank outputs in feature extraction for robust ASR,” in *Proc. Interspeech*, pp. 933–936, 2005.

[112] J. F. Gemmeke, T. Virtanen, and A. Hurmalainen, “Exemplar-based sparse representations for noise robust automatic speech recognition,” *IEEE Trans. Audio, Speech and Language Processing*, vol. 19, no. 7, pp. 2067–2080, 2011.

[113] C.-P. Chen and J. A. Bilmes, “MVA processing of speech features,” *IEEE Trans. Audio, Speech and Language Processing*, vol. 15, pp. 257–270, jan. 2007.

[114] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. De Mori, “Linear hidden transformation for adaptation of hybrid ann/hmm models,” *Speech Communication*, vol. 49, pp. 827–835, 2007.

[115] P. J. Moreno, *Speech Recognition in Noisy Environments*. PhD thesis, ECE Department, CMU, 1996.

[116] J. Droppo, L. Deng, and A. Acero, “Evaluation of SPLICE on the Aurora 2 and 3 tasks,” in *Proc. ICSLP*, pp. 29–32, 2002.

[117] S. Tamura and A. Waibel, “Noise reduction using connectionist models,” in *Proc. ICASSP*, pp. 553–556, apr 1988.

[118] B. Raj and R. Stern, “Missing-feature approaches in speech recognition,” *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 101–116, 2005.

[119] M. J. F. Gales, *Model-based techniques for noise robust speech recognition*. PhD thesis, Cambridge University, 1995.

[120] M. J. F. Gales and S. Young, “Robust continuous speech recognition using parallel model combination,” *IEEE Trans. Speech and Audio Processing*, vol. 4, no. 5, pp. 352–359, 1996.

[121] S. Sagayama, Y. Yamaguchi, S. Takahashi, and J. Takahashi, “Jacobian approach to fast acoustic model adaptation,” in *Proc. ICASSP*, vol. 2, pp. 835–838, apr 1997.

[122] A. Acero, L. Deng, T. Kristjansson, and J. Zhang, “HMM adaptation using vector Taylor series for noisy speech recognition.,” in *Proc. Interspeech*, pp. 869–872, 2000.

[123] J. Droppo, A. Acero, and L. Deng, “Uncertainty decoding with SPLICE for noise robust speech recognition,” in *Proc. ICASSP*, pp. 57–60, 2002.

[124] H. Xu, M. J. F. Gales, and K. Chin, “Joint uncertainty decoding with predictive methods for noise robust speech recognition,” *IEEE Trans. Audio, Speech and Language Processing*, vol. 19, pp. 1665–1676, Aaugust 2011.

[125] J. Li, L. Deng, D. Yu, Y. Gong, and A. Acero, "High performance HMM adaptation with joint compensation of additive and convolutive distortions via vector Taylor series," in *Proc. ASRU*, 2007.

[126] J. Li, L. Deng, D. Yu, Y. Gong, and A. Acero, "A unified framework of HMM adaptation with joint compensation of additive and convolutive distortions," in *Computer Speech and Language*, vol. 23, pp. 389–405, 2009.

[127] X. Xiao, J. Li, E. S. Chng, and H. Li, "Lasso environment model combination for robust speech recognition," in *Proc. ICASSP*, pp. 4305–4308, March 2012.

[128] M. L. Seltzer and A. Acero, "Factored adaptation for separable compensation of speaker and environmental variability," in *ASRU*, pp. 146–151, 2011.

[129] E. Trentin and M. Gori, "A survey of hybrid ANN/HMM models for automatic speech recognition," *Neurocomputing*, vol. 37, pp. 91–126, Apr 2001.

[130] ETSI-SMG, "European digital cellular telecommunication system (Phase 1); Transmission planning aspects for the speech service in GSM PLMN system," tech. rep., ETSI-SMG, July 1994.

[131] ETSI, "Speech processing, transmission and quality aspects STQ; distributed speech recognition; advanced front-end feature extraction algorithm; compression algorithms," tech. rep., ES 202 050, 2002.

[132] P. Moreno, B. Raj, and R. Stern, "A vector Taylor series approach for environment-independent speech recognition," in *Proc. ICASSP*, vol. 2, pp. 733–736, 1996.

[133] D. Y. Kim, C. K. Un, and N. S. Kim, "Speech recognition in noisy environments using first-order vector Taylor series," *IEEE Trans. Signal Processing*, vol. 24, no. 1, pp. 39 – 49, 1998.

[134] M. Van Segbroeck and H. Van hamme, "Advances in missing feature techniques for robust large vocabulary continuous speech recognition," *IEEE Trans. Audio, Speech and Language Processing*, vol. 19, no. 1, pp. 123–137, 2011.

[135] M. Van Segbroeck, *Robust large vocabulary continuous speech recognition using missing data techniques*. PhD thesis, ESAT,K.U. Leuven, January 2010.

[136] J. F. Gemmeke and B. Cranen, “Noise robust digit recognition using sparse representations,” in *Proc. ISCA Tutorial and Research Workshop on Speech Analysis and Processing for Knowledge Discovery*, pp. 1–4, 2008.

[137] J. F. Gemmeke, A. Hurmalainen, T. Virtanen, and Y. Sun, “Toward a practical implementation of exemplar-based noise robust ASR,” in *Proc. EUSIPCO*, pp. 1490–1494, 2011.

[138] J. F. Gemmeke and H. Van hamme, “Advances in noise robust digit recognition using hybrid exemplar-based techniques,” in *Proc. Interspeech*, 2012.

[139] L. Bahl, P. Brown, P. de Souza, and R. Mercer, “Maximum mutual information estimation of hidden Markov model parameters for speech recognition,” in *Proc. ICASSP*, vol. 11, pp. 49–52, apr 1986.

[140] P. Woodland and D. Povey, “Large scale discriminative training of hidden Markov models for speech recognition,” *Computer Speech and Language*, vol. 16, no. 1, pp. 25–47, 2002.

[141] J. Du, P. Liu, F. Soong, J.-L. Zhou, and R.-H. Wang, “Noisy speech recognition performance of discriminative HMMs,” in *Proc. ISCSLP*, pp. 358–369, 2006.

[142] O. Vinyals and S. Ravuri, “Comparing multilayer perceptron to deep belief network tandem features for robust ASR,” in *Proc. ICASSP*, pp. 4596–4599, 2011.

[143] O. Vinyals, S. Ravuri, and D. Povey, “Revisiting recurrent neural networks for robust ASR,” in *Proc. ICASSP*, pp. 4085–4088, 2012.

[144] C. M. Bishop, “Training with noise is equivalent to tikhonov regularization,” *Neural Computation*, vol. 7, pp. 108–116, Jan 1994.

[145] R. Penrose, “A generalized inverse for matrices,” *Proc. The Cambridge Philosophical Society*, vol. 51, pp. 406–413, July 1955.

[146] G.-B. Huang, X. Ding, and H. Zhou, “Optimization method based extreme learning machine for classification,” *Neural Computation*, vol. 74, no. 13, pp. 155–163, 2010.

[147] F. Wilcoxon, “Individual Comparisons by Ranking Methods,” *Biometrics Bulletin*, vol. 1, pp. 80–83, Dec. 1945.

[148] B. Schuller, M. Wöllmer, T. Moosmayr, and G. Rigoll, "Recognition of noisy speech: A comparative survey of robust model architecture and feature enhancement," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2009, no. 1, p. ID942617, 2009.

[149] S.-X. Zhang and M. Gales, "Structured support vector machines for noise robust continuous speech recognition," in *Proc. Interspeech*, pp. 989–992, 2011.

[150] H. Bourlard and N. Morgan, *Connectionist speech recognition: a hybrid approach*, vol. 247 of *The Kluwer International Series in Engineering and Computer Science*. Springer US, 1994.

[151] H. Hermansky, D. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional hmm systems," in *Proc. ICASSP*, pp. 1635–1638, 2000.

[152] K. Demuynck, J. Duchateau, and D. Van Compernolle, "Optimal feature sub-space selection based on discriminant analysis," in *Proc. Eurospeech*, pp. 1311–1314, 1999.

[153] H. G. Hirsch and D. Pearce, "Applying the advanced ETSI frontend to the Aurora-2 task," tech. rep., version 1.1, 2006.

[154] L. Deng, A. Acero, L. Jiang, J. Droppo, and X. Huang, "High-performance robust speech recognition using stereo training data," in *Proc. ICASSP*, vol. 1, pp. 301–304, 2001.

[155] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. ICML*, pp. 1096–1103, 2008.

[156] Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 32, no. 6, pp. 1109–1121, 1984.

[157] P. J. Wolfe and S. J. Godsill, "Efficient alternatives to the ephraim and malah suppression rule for audio signal enhancement," in *special issue) EURASIP JASP on Digital Audio for Multimedia Communications*, pp. 1043–1051, 2003.

[158] D. Yu, L. Deng, J. Droppo, J. Wu, Y. Gong, and A. Acero, "Robust speech recognition using a cepstral minimum-mean-square-error-motivated noise suppressor," *IEEE Trans. Audio, Speech and Language Processing*, vol. 16, no. 5, pp. 1061–1070, 2008.

[159] K. Indrebo, R. Povinelli, and M. Johnson, "Minimum mean-squared error estimation of mel-frequency cepstral coefficients using a novel distortion model," *IEEE Trans. Audio, Speech and Language Processing*, vol. 16, no. 8, pp. 1654–1661, 2008.

[160] L. Deng, J. Droppo, and A. Acero, "Estimating cepstrum of speech under the presence of noise using a joint prior of static and dynamic features.," *IEEE Trans. Speech and Audio Processing*, vol. 12, no. 3, pp. 218–233, 2004.

[161] R. Rotili, E. Principi, S. Cifani, F. Piazza, and S. Squartini, "Multichannel feature enhancement for robust speech recognition," *Speech technologies. InTech*, pp. 978–953, 2011.

[162] M. Lukoševičius, "A practical guide to applying echo state networks," in *Neural Networks: Tricks of the Trade*, vol. 7700 of *Lecture Notes in Computer Science*, pp. 659–686, Springer Berlin Heidelberg, 2012.

[163] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[164] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, pp. 318–362, 1986.

[165] P. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. DAR*, pp. 958–963, Aug 2003.

[166] Y. Tang and C. Eliasmith, "Deep networks for robust visual recognition," in *Proc. ICML*, pp. 1055–1062, 2010.

[167] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *AISTATS*, pp. 448–455, 2009.

[168] D. Yu and L. Deng, "Deep convex net: A scalable architecture for speech pattern classification.," in *Proc. Interspeech*, pp. 2285–2288, 2011.

[169] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Proc. IJCAI*, pp. 1237–1242, 2011.

[170] M. Ranzato, C. S. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model.," in *Proc. NIPS*, pp. 1137–1144, MIT Press, 2006.

[171] D. C. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. CVPR*, pp. 3642–3649, 2012.

[172] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Proc. NIPS*, pp. 350–358, 2012.

[173] F. Agostinelli, M. Anderson, and H. Lee, "Adaptive multi-column deep neural networks with application to robust image denoising," *Proc. NIPS*, vol. 26, pp. 1–9, 2013.

[174] M. Lukoševičius, H. Jaeger, and B. Schrauwen, "Reservoir computing trends," *Künstliche Intelligenz*, vol. 26, no. 4, pp. 365–371, 2012.

[175] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Handwritten digit recognition with a committee of deep neural nets on GPUs," *Neural Networks Tricks of the Trade*, 2011.

[176] D. Decoste and B. Schölkopf, "Training invariant Support Vector Machines," *Machine Learning*, vol. 46, pp. 161–190, Mar. 2002.

[177] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, and N. Dahlgren, "The DARPA TIMIT acoustic-phonetic continuous speech corpus cd-rom," tech. rep., NIST, 1993.

[178] J. Ming and F. Smith, "Improved phone recognition using bayesian triphone models," in *Proc. ICASSP*, pp. 409–412, 1998.

[179] S. Fernández, A. Graves, and J. Schmidhuber, "Phoneme recognition in TIMIT with BLSTM-CTC," tech. rep., IDSIA-04-08, 2008.

[180] F. Triefenbach, K. Demuynck, and J.-P. Martens, "Large vocabulary continuous speech recognition with reservoir-based acoustic models," *IEEE Signal Processing Letter*, vol. 21, pp. 311–315, March 2014.

[181] D. B. Paul and J. M. Baker, "The design for the wall street journal-based csr corpus," in *Proc. of the workshop on Speech and Natural Language*, HLT '91, pp. 357–362, 1992.

[182] G. Heigold, S. Wiesler, M. Nussbaum-Thom, P. Lehnen, R. Schluter, and H. Ney, "Discriminative hmms, log-linear models, and crfs: what is the difference?," in *Proc. ICASSP*, pp. 5546–5549, 2010.

[183] S. Siniscalchi, T. Svendsen, and C.-H. Lee, “A bottom-up modular search approach to large vocabulary continuous speech recognition,” *IEEE Trans. Audio, Speech and Language Processing*, vol. 21, no. 4, pp. 786–797, 2013.

[184] S. Siniscalchi, D. Yu, L. Deng, and C.-H. Lee, “Speech recognition using long-span temporal patterns in a deep network model,” *IEEE Signal Processing Letter*, vol. 20, no. 3, pp. 201–204, 2013.

[185] M. Hermans, *Expanding the theoretical framework of reservoir computing*. PhD thesis, Ghent University, 2012.