Abstract Pattern Learning with Neural Networks for Improved Sequential Modeling

CITY
UNIVERSITY OF LONDON
EST 1894

Radha Manisha Kopparti

Department of Computer Science

City, University of London

A thesis submitted in partial fulfillment of the requirement for the degree of

Doctor of Philosophy

December 2020

I would like to dedicate this thesis to my loving parents. This work would be impossible without their unwavering support, care and blessings.

I owe everything to them.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains less than 65,000 words including appendices, bibliography, footnotes, tables and equations and has less than 150 figures. I grant powers of discretion to the City, University of London librarian to allow the dissertation to be copied in whole or in part without further reference to myself (the author). This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgment.

Radha Manisha Kopparti

December 2020

Acknowledgements

Firstly, my sincere and profound gratitude to my thesis supervisor Tillman Weyde for giving me the opportunity to work with him, for all the constant support, guidance and motivation to pursue this research work. I feel very blessed to be associated with him for the past three years and it has been a great learning experience for me. He was very kind and approachable, no matter what time of the day it was, he was there to help me, I feel very fortunate to have a supervisor like him.

I would like to thank my co-supervisor Artur Garcez for his great support and encouragement. I'm grateful for the financial support offered to me by City University London through a three-year studentship and the Data Science Institute for the travel grants and funding part of my fees in the course of the program. My sincere thanks to my examiners Brian McFee and Ernesto Jimenez Ruiz for their time, prompt response and making the online viva experience memorable. The feedback and discussion during the viva was very valuable. Special thanks to Esther Mondragón for chairing my viva.

I would also like to extend my warm regards to all the faculty in the Department of Computer Science at City for their help over the past three years. Thanks to Jacob Howe for reviewing my MPhil to PhD transfer report. Many thanks to the course officers especially Ann Marie De Here, Savita Afonso, David Mallo-Ferrer, Nathalie Chatelain, and academic staff members Paula Green, Asif Nawaz, Gabrielle for all their valuable help on numerous occasions.

I would like to thank all my friends and colleagues without whom the stay in London wouldn't have been awesome. Fatemeh, Atif, Abinaya, Alex, Ayesha, Eric, Nadine, Charitos, Ester, Sarah, Simon, Benedict, Warren, Nathan, Swetha, Bhagya and Naumana it has been great knowing you all. Thanks for being my best buddies in the past three years, I will always cherish the time spent with you people.

Abstract

Deep neural networks have been widely used for various applications and have produced state-of-the-art results in domains like speech recognition, machine translation, and image recognition. Despite the impressive successes achieved with deep neural networks, there has been an increasing awareness that there are tasks that still elude neural network learning, specifically the learning of abstract grammatical patterns and generalisation of the abstract patterns beyond the training data.

In this thesis, the problem of learning abstract patterns based on equality (also called identity) with neural networks is addressed. It was found in this study that feed-forward neural networks do not learn equality. This leads to feed-forward and recurrent neural networks' inability to learn abstract patterns. This problem is studied empirically and constructive solutions are developed in this thesis.

A solution is proposed, which is called *'Relation Based Patterns'* (RBP) models abstract relationships based on equality by using fixed weights and a special type of neuron. An extension of RBP called *'Embedded Relation Based Patterns'* (ERBP) is also proposed which models RBP as a Bayesian prior on network weights implemented as a regularisation term in otherwise standard neural network learning. Both RBP and particularly ERBP are very easy to integrate into standard neural network models. It is observed in experiments that integration of (E)RBP structures leads to almost perfect generalisation in abstract pattern learning tasks with synthetic data and to improvements also in neural language and music modeling. (E)RBP has been successfully applied on various neural network models like Feed-forward neural network (FFNN), RNN and their gated variants like GRUs and LSTMs, Transformers and Graph Neural Networks. It leads to improvements on real-word tasks like melody prediction, character and word prediction, abstract compositionality and graph edit distance.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

The ability to learn abstractions and generalise is seen as the essence of human intelligence (Brooks, 1991). Since 1950s, there have been efforts to build systems that learn and think like humans (Lake et al., 2016). The lack of systematicity in neural network learning has already been discussed over 30 years ago (Fodor and Pylyshyn, 1988). It was rekindled by Marcus et al. (1999) showing the failure of neural networks to recognise abstract patterns that infants learn in minutes. In the recent years, deep neural networks have had impressive successes based on computational advances and greatly increased amounts of data. Yet, it has become evident that there are still relevant limitations to systematicity of learning and generalisation in current neural network architectures (e.g., Baroni, 2019; Hupkes et al., 2019; Keresztury and Bruni, 2020; Lake and Baroni, 2018; Marcus, 2018). The emerging picture is that neural networks are effective at recognising and mapping numerical patterns seen in the training data to corresponding outputs, but very often do not extrapolate this mapping outside successfully on unseen data (Liska et al., 2018).

This problem also relates to recent discussions, sparked by Marcus (2018), about deep neural networks' need for very large amounts of training data, lack of robustness and lack of transparency as also expressed, e.g., by (Kansky et al., 2017; Sabour et al., 2017; Szegedy et al., 2013). These issues related to the lack of generalisation by neural networks beyond the space covered by the input data, i.e. extrapolation, is generally seen as requiring an inductive bias in the learning system, but there is no general agreement about the nature or implementation of inductive biases for neural networks, e.g. (Barrett et al., 2018; Feinman and Lake, 2018). Marcus et al. (1999) and (Marcus, 2018) see the failure to learn abstract patterns as evidence for general limitations of neural network learning. To our knowledge, this link has not been shown experimentally.

1.2 Problem Overview and Research Objectives

This thesis starts with the abstract pattern learning problem posed by (Marcus et al., 1999). The task was to recognize patterns defined by simple grammatical rules, specifically sequences of the structure ABA or ABB. The patterns are simple and defined in terms of equality between sequence elements.[1] Seven month old infants showed evidence of recognizing the patterns after just two minutes of familiarization while recurrent neural neural failed at the task. In the exchange of arguments that followed over several years in several new studies no general consensus was reached on whether and to what extent standard feed-forward and recurrent networks learn abstract patterns. In experiments it is here confirmed that standard feed-forward and recurrent neural network models are not able to learn abstract grammar-like rules of Marcus et al. (1999). The underlying problem turns out to be that standard neural networks do not seem to learn to recognise equality, which is fundamental for these patterns and many higher level tasks requiring systematic or compositional learning. While the generalising solutions to equality and abstract pattern tasks are clearly in the hypothesis space of neural networks, gradient descent methods fail to find these solutions. This prompted the interest in an inductive bias that enables neural networks to find these solutions.

[1]The term identity is also often used for this kind of rule. Identity and equality are used interchangeably in this thesis.

It is identified that both the comparison of related input neurons and of input tokens needs to be predefined in the network to learn general rules from data. The goal is to design such a bias and evaluate its effectiveness on equality and abstract pattern learning as well as practical tasks. In recent years, there was a trend to remove human designed features from neural networks, and leave everything to be learned from the data (Bengio et al., 2013). In this work the inverse approach is followed, which is to add a designed internal representation in standard neural networks. The developed methods are evaluated both on the synthetic data and real world data.

1.3 Research Questions

The primary focus of the research work addresses the following research questions :

- Can feed-forward neural networks **learn equality**? If not, why not?
- How do we make **standard neural networks** learn equality?
- How does equality learning relate to **abstract pattern learning**?
- What kind of **inductive biases** are required for abstract pattern learning, and how do we add / implement them in the network?
- How do the inductive bias methods affect **real world tasks** and neural network learning in general?

1.4 Main Contributions

In this research work, the problem of learning abstract patterns based on equality using neural networks is studied. Equality learning has been addressed as a standard problem, and evaluated with empirical methods. The solutions for learning equality has been proposed in the thesis, which also further helps in understanding and realising why standard neural networks cannot learn abstract grammatical patterns. Hence, the solutions for solving equality has been applied on tasks like abstract grammar/pattern learning. This shows that modeling lower level equality and abstract pattern learning in neural networks helps in learning real world problems. Thus, the main contribution lies in terms of realising equality learning as a problem that standard neural networks cannot solve, developing solutions for this to solve abstract pattern learning, and applying this to real-world tasks like on natural language, music and graphs.

To solve this problem, simple ways of adding inductive bias into standard neural network architectures has been proposed. Firstly, a solution called *'Relation Based Patterns'* (RBP) is proposed. RBP creates an inductive bias in the neural networks that leads to the learning of generalisable solutions to equality and abstract patterns and improved language and music modelling. Second, *'Embedded Relation Based Patterns'* (ERBP) are also proposed, which is based on Relation Based Patterns (RBP), but modeled as a Bayesian prior on network weights and implemented as a regularisation term in otherwise standard network learning.

RBP and especially ERBP are easy to integrate into standard neural networks and do not affect their general learning capacity. In our experiments, (E)RBP lead to almost perfect generalisation when learning abstract patterns from synthetic data. (E)RBP also improves natural language models on the word and character level, pitch prediction in melodies, linguistic compositionality tasks and graph distance estimation. While most current approaches focus on the structure of the network and adding different operators, this is to our knowledge the first approach that addresses abstract pattern recognition with a prior on the network weights. The main contributions of the thesis are to:

- experiment with several common NN architectures: feed-forward networks, RNN, GRU, and LSTM, to find that they fail to learn abstract patterns
- identify reasons that prevent the learning process from being successful in the context of learning abstractions in data

- formulate equality learning as a problem and link to how it helps in generalizing abstract grammar patterns better
- propose Relation Based Patterns (RBP) and Embedded Relation Based Patterns (ERBP) as new methods to enable the learning of equality and abstract patterns with standard network structures
- show in experiments that abstract patterns can be learnt with (E)RBP structure on artificial data, including mixed abstract and concrete patterns
- show (E)RBP improves performance in real word tasks on language, music and graphs applied to state of the art neural network models like Transformers and Graph neural networks.

1.5 Publications

1.5.1 Journal Papers

- Radha Kopparti and Tillman Weyde (2020). **'Relational Weight Priors in Neural Networks for Abstract Pattern Learning and Language Modeling'** (under review for Machine Learning)
- Tillman Weyde and Radha Kopparti (2019). **'Modelling Identity Rules with Neural Networks'**, Volume 6, No 4, 745–769, Journal of Applied Logic (JAL), ISSN 2631-9829.

1.5.2 Conference Papers

1. Work on Rule Learning / Abstract Relations :
    - Radha Kopparti and Tillman Weyde (2019). **'Weight Priors for Learning Identity Relations'.** KR2ML @ NeurIPS, Vancouver, Canada, Dec 8-14.

- Radha Kopparti and Tillman Weyde (2019). **'Factors for the Generalisation of Identity Relations by Neural Networks'.** Workshop on Understanding and Improving Generalization in Deep Learning, ICML, Long Beach, June 9-15.
- Tillman Weyde and Radha Kopparti (2018). **'Feed-Forward Neural Networks need Inductive Bias to Learn Equality Relations'.** Workshop on Relational Representation Learning at NIPS, Montreal, Canada, December 3-8.

2. Work on Music Modeling :

   - Radha Kopparti and Tillman Weyde (2020). **'Monophonic Pitch Prediction using Transformers with Relational Representations'** (submitted)
   - Radha Kopparti and Tillman Weyde (2019). **'Modeling Interval Relations for Neural Music Language models'.** Workshop on Machine Learning for Music Discovery, ICML, Long Beach, June 9-15.
   - Radha Kopparti and Tillman Weyde (2018). **'Evaluating repetition based melody prediction over different context lengths'.** ICML Joint Workshop on Machine Learning and Music, Stockholm, Sweden, July 14.
   - Radha Kopparti and Tillman Weyde (2018). **'Repetition Based Melody Prediction Model'**, at WiMiR workshop as part the 19th International Society of Music Information Retrieval (ISMIR), Paris, France, Sep 22-28.

1.5.3 Poster presentations

- Poster/Talk on **'Abstract Rule-based Learning with Neural Networks'**, AAAI-2020 Doctoral Consortium, New York, Feb 6-12, 2020.

- Poster/Talk on **'Abstract Rule learning with Neural Networks'**, MI21-HLC 2019, Windsor, Berkshire, UK, June 30- July 3.
- Poster/Talk on **'Repetition Based Melody Modelling'** at Again and Again Workshop, City University of London, April 24-25, 2019.
- Poster on **'Learning Abstraction Relationships using Neural Networks'**, at WIML co-located with NIPS , Montreal, Canada, December 3-8, 2018.
- Poster on **'Sequential Rule Learning with Neural Networks'**, at Machine Learning Summer School, at Universidad Autónoma de Madrid, Madrid, Spain, Aug 27 - Sep 8, 2018 and Spring School on Language, Music, and Cognition: Organizing Events in Time. University of Cologne, Cologne, Germany, February 2-8, 2018.

1.6 Code

Code for the experiments performed in the thesis are available here :

- **https://github.com/radhamanisha1/RBP-architecture** and
- **https://github.com/radhamanisha1/ERBP-models**

1.7 Organisation of the thesis

The thesis is organised as follows :

- *Chapter 1 : Introduction* provides an overview of the thesis, covering motivation behind the work, problems and research objectives,and the main contributions of the thesis. Publications and Code details are also provided.
- *Chapter 2 : Literature Review* in the thesis is divided into two parts. The first section of the literature review covers the abstract rule learning work which is

the main problem solved in the thesis. The later part focuses on sequence to sequence modeling tasks, compositionality and music language modeling works.

- *Chapter 3 : Relation Based Patterns Approach* introduces the Relation Based Patterns (RBP) approach, the design and ways of integrating them into neural network models. Further the application of RBP on the rule learning task, vector equality task and numerical tasks like digit reversal, parity and sum of digits are shown. Different experiments and variants of the models are explained in this chapter.

- *Chapter 4 : Case study : Applying RBP for Melody Modeling* shows the application of RBP on Melody Modeling task. The overview of the process of how RBP has been applied on pitch prediction, dataset used and the experimental results are summarized. Different experiments ranging from Markov Models to Neural Networks are performed as shown in this chapter.

- *Chapter 5 : Embedded Relation Based Patterns Approach* introduces the Embedded Relation based Patterns ((E)RBP) approach which is a bayesian approach to the RBP method introduced in Chapter 5. Several Experiments with (E)RBP are performed ranging from abstract pattern learning, mixed abstract and concrete pattern learning, character prediction, melody prediction and word prediction. The results are compared with the RBP model and summarized in the chapter.

- *Chapter 6 : Experimenting (E)RBP with different architectures and tasks* extends the ERBP approach further on the recently popular neural network architectures like Transformers and Graph Neural Networks. It also covers new tasks like abstract compositionality and shows how adding (E)RBP structures in the neural network models help in improving the performance.

- *Chapter 7 : Conclusions* has the final remarks and summarizes about the problems addressed in the thesis, approaches proposed followed by the future scope of work. References are provided after the Conclusions.

Chapter 2

Literature Review

The literature review is divided into the following parts. Firstly, the abstract pattern learning is discussed, describing the earlier works and methods proposed. In the second part, music melody modeling works are summarized. The later part of this chapter focuses on sequence to sequence modeling tasks, compositionality and inductive bias methods.

2.1 Abstract Pattern Learning

In abstract pattern learning, specifically as in the studies by Marcus (2001); Marcus et al. (1999), the task is the learning of abstract patterns from sequential data. Abstract patterns are defined by the relations between elements of a sequence, rather than the specific values, and can described as grammar-like rules. The goal is to learn to apply these rules based on sequences of example data like words in a language or notes in music. This generalization of structured combinatorial relations from language-like data is also known as rule learning or grammar learning (Alhama and Zuidema, 2019b). In grammar learning, the key idea is to understand the rules of a language without prior knowledge, just from the data. This is not only of interest in machine learning but also as a model for human language acquisition by comparing the ability of humans and artificial models in learning artificial and natural grammatical patterns.

There are two phases in the grammar learning task as addressed in this thesis, one is the training stage where the patterns are made familiar to the subjects (i.e. humans or machines) and the ability to identify and generalise this new knowledge is tested in the second stage, the testing phase. The testing phase typically comprises the recognition of patterns used in the training phase but, at least partly, transferred to another set of symbols or sounds or in a new sequential context. This corresponds to the ability of humans to use words they have just learnt in a grammatically correct way, or to arrange known words into a grammatically sentence they have not seen before (Chomsky, 1986).

There have been several studies on grammar learning in psychology. Gordon and Holyoak (1983) made an early contribution on implicit learning and generalisation. Subsequently, Knowlton and Squire (1994, 1996) studied specifically the knowledge acquired during artificial grammar learning tasks. An influential contribution to abstract pattern learning was by Marcus et al. (1999) and Marcus (2001) where the abstract grammar like rules are shown to seven-month-old infants and their reactions showed that they could distinguish the sequences according to the underlying grammatical rules. The infants were first exposed to sequences of one of the forms ABA or ABB, e.g. ‘le di le’ or ‘le di di’, for a few minutes in the familiarisation phase. In the test phase the infants were exposed to sequences with a different vocabulary (e.g. ‘ba po ba’ and ‘ba po po’) and they reacted by approaching significantly more often the loudspeaker that played sequences with a new structure, thus showing that the experiments was that the infants were able to learn the grammatical rules within a few minutes of habituation. The amount of time taken by the infants was noted using a head-turn preference procedure (HTPP) where the looking times of both the grammar conditions are measured.

A total of three experiments was performed. The stimuli used in Experiment 1 had mostly the combination *‘voiced-unvoiced-voiced’*. There is a possibility that the infants learned the voicing pattern rather than abstract rule, as there are studies which showed that infants seem to be sensitive to voicing distinctions in the early months

(Eimas et al., 1971). Experiment 2 was performed with grammar patterns highlighting lexical features instead of phonetic features and the results of the experiments were consistent with Experiment 1, showing that infants do learn abstract grammar rules effectively. In another Experiment 3, the infants were tested on the grammars of the form *'ABB'* and *'AAB'* where both the grammars contain immediate repetitions. In Experiments 1 and 2 above, one of the grammar patterns had an immediate repetition (eg: *'ABB'* and the other didn't have an immediate repetition *'ABA'*. Results from Experiment 3 also suggest that infants showed a significant preference for the novel stimuli, ruling out the possibility of infants distinguishing the grammars, was based on having or lack of immediate repetition alone.

The type of pattern used in these experiments depends only on the equality between elements of the sequence and after successful learning it should be recognisable independently of the vocabulary used. Therefore underlying rules of these abstract patterns are often called identity or equality rules. This suggests that the standard networks have a problem in learning equality although the solution lies within the hypothesis space of the network. When a recurrent neural network, often called an Elman network, was applied to an equivalent task, it failed to distinguish the grammar patterns, showing no significant difference in its outputs.

The finding by Marcus et al. (1999) sparked an exchange about whether human speech acquisition is based on rules or statistics and the proposal of several neural networks models that claimed to match the experimental results. Seidenberg and Elman (1999) and Elman (1999) proposed a solution based on a distributed representation of the input and on pre-training where the network is first trained to recognise repeated items in a sequence. The network is subsequently trained on classifying ABA vs ABB patterns. Only Elman (1999) reports specific results and has only 4 test data points, but 100% accuracy. However, Vilcu and Hadley (2001) reported that they could not reproduce these results. Shultz and Bale (2001) and Altmann and Dienes (1999) suggested solutions which are based on modified network architectures and training methods. Vilcu and Hadley (2005) could not replicate the results by Altmann and

Dienes (1999) and found that the models by Shultz and Bale (2001) do not generalise. The claims by Vilcu and Hadley (2005) were again contested by Shultz et al. (2005). A number of other methods were suggested that used specifically designed network architectures, data representations, and training methods, such as (Christiansen et al., 2000; Dominey and Ramus, 2000; Gasser and Colunga, 1999; Shastri and Chang, 1999). Further discussions of these results focused more on psychological aspects of this experiment. More recent work by Alhama and Zuidema (2016) suggests that prior experience or pre-defined context representation ("pre-training" or "pre-wiring") is necessary for the network to learn general identity rules when using echo state networks.

The discussion of this problem should be seen as part of a wider debate on the systematicity of language learning models, which started in the 1980s and 1990s (Fodor and Pylyshyn, 1988; Hadley, 1994a). This debate, like the more specific one on identity rules, has been characterised by claims and counter-claims (Chalmers, 1990; Christiansen and Chater, 1994; Fodor and McLaughlin, 1990; Hadley, 1994b; Niklasson and van Gelder, 1994; Smolensky, 1987), which, as stated by (Frank, 2014), often suffer from a lack of empirical grounding. Recently, the work in Lake and Baroni (2018) has defined a test of systematicity in a framework of translation, applied it to standard *seq2seq* neural network models (Sutskever et al., 2014). They found that generalisation occurs in this setting, but it depends largely on the amount and type of data shown, and does not exhibit the extraction and systematic application of rules in the way a human learner would.

While the works listed above are interesting and relevant, they do not answer one important question: whether commonly used network architectures can learn abstract patterns. In addition: most of the studies above, the evaluation has mostly been conducted by testing whether the output of the network shows a statistically significant difference between inputs that conform to a trained abstract pattern and those that do not. From a machine learning perspective, this criterion is not satisfactory as we, like (Lake and Baroni, 2018), would expect that an identity/equality rule should always be applied correctly once if it has been learned from examples, at least in cases of noise-free synthetic data. Therefore it is interesting to explore the question of whether and how this general rule learning can be achieved with common neural network types for sequence classification.

2.2 Music Language Modeling

One of the real-world examples of abstract patterns are musical melodies. Musical melodies have culture-specific abstract repetition patterns as will be shown in this thesis. A useful way of testing for abstract patterns is by modelling melodic repetitions to gain a better understanding and better models of melodies. Musical melodies have been stated as the first examples of a Gestalt already in 1890s by (von Ehrenfels, 1890). In Gestalt psychology, it was observed that people can distinguish between melodies, even though they have been transposed to a new key or tuning using different set of notes (Ellis, 1938; von Ehrenfels, 1890).

One of the initial works focusing on the abstract mathematical interpretations of information theory in the context of music was done by (Coons and Kraehenbuehl, 1958; Kraehenbuehl and Coons, 1959). In that study, various sequences of symbols such as (AAAB, AABA, ABAC, AABB etc.) are ordered rank-wise according to the indices of hierarchy and articulatedness. The authors assumed that these symbol sequences reflected song structures and explained how their positions in the rank-ordering might reflect their use in composition.

In mid 1990s, there have been series of works on statistical modeling of musical structure (Conklin, 1990; Conklin and Cleary, 1988; Conklin and Witten, 1995). Few other works on melody prediction using statistical modeling are (Conklin and Witten, 2003b; Paiement et al., 2009a,b; Pearce and Wiggins, 2004, 2012). There are also works on implicit learning and acquisition of music (Rohrmeier and Rebuschat, 2012) and statistical learning of musical tones by infants and humans (Saffran, 1999). Other works explored the use of connectionist models for music prediction tasks (Baffioni et al., 1984; Bharucha, 1987; Cherla et al., 2013, 2015).

A wide range of statistical methods using n-gram models (Paiement et al., 2009a,b; Pearce and Wiggins, 2004) have been used for music prediction tasks. N-gram models are often criticised as unable to capture long-term dependencies. Techniques like smoothing in combination with n-gram models of unbounded context length have been proposed by Pearce and Wiggins (2004) to avoid problems of overfitting and sparsity in music. As an extension to n-gram models, Conklin and Witten (2003a) proposed multiple viewpoint systems for melody modelling, which take into account various combinations of features and have shown to improve melody prediction tasks. Pearce (2005) developed the IDyOM framework for constructing multiple-viewpoint variable-order Markov models for predictive statistical modelling of musical structure.

Markov models like the above normally calculate the probabilities as the relative frequencies of concrete events or features, hence they cannot account for representing the abstract repetition patterns unless the relation that pattern is based on is encoded in a feature and the feature is applied to all relevant notes. Thus, generalising identity patterns from one pitch to another is limited using this approach.

An alternative to statistical models are recurrent neural networks which are popularly used for modelling sequential data. The simplest form of RNN is the Elman network proposed in 1990 (Elman, 1990a). It differs from a feed-forward neural network by having a temporal context layer that acts as input to the hidden layer. There are two popular gated variants of RNNs called Long Short Term Memory (Hochreiter and Schmidhuber, 1997b) and Gated Recurrent Units (Chung et al., 2014). RNNs, especially LSTMs, have been used for music generation for about 20 years and seen a surge in interest recently. LSTMs have also been very successful in other sequence modelling tasks, such as speech recognition and Natural Language Modelling Franklin (2004).

Connectionist models with multiple viewpoint modelling have been shown to outperform n-gram models (Cherla et al., 2013, 2015). Recent improvements to melody modeling include, a feature discovery approach Langabhel (et.al., 2017) using the *PULSE* learning framework, and a predictive model based on recurrent gated autoencoders for learning structured interval representations (Lattner et al., 2018). More recently, Transformer based models Vaswani et al. (2017) have been used for generating structured music compositions Huang et al. (2018) across longer polyphonic sequences.

Transformers are recently popular sequence-to-sequence models, which are currently the state of the art in language modelling tasks, having beating LSTMs and other RNN variant models (Vaswani et al., 2017). In Transformers, there is a Encoder-Decoder model coupled with multi-head attention blocks that serve as the basis for deciding which parts of the input sequence are important (Vaswani et al., 2017).

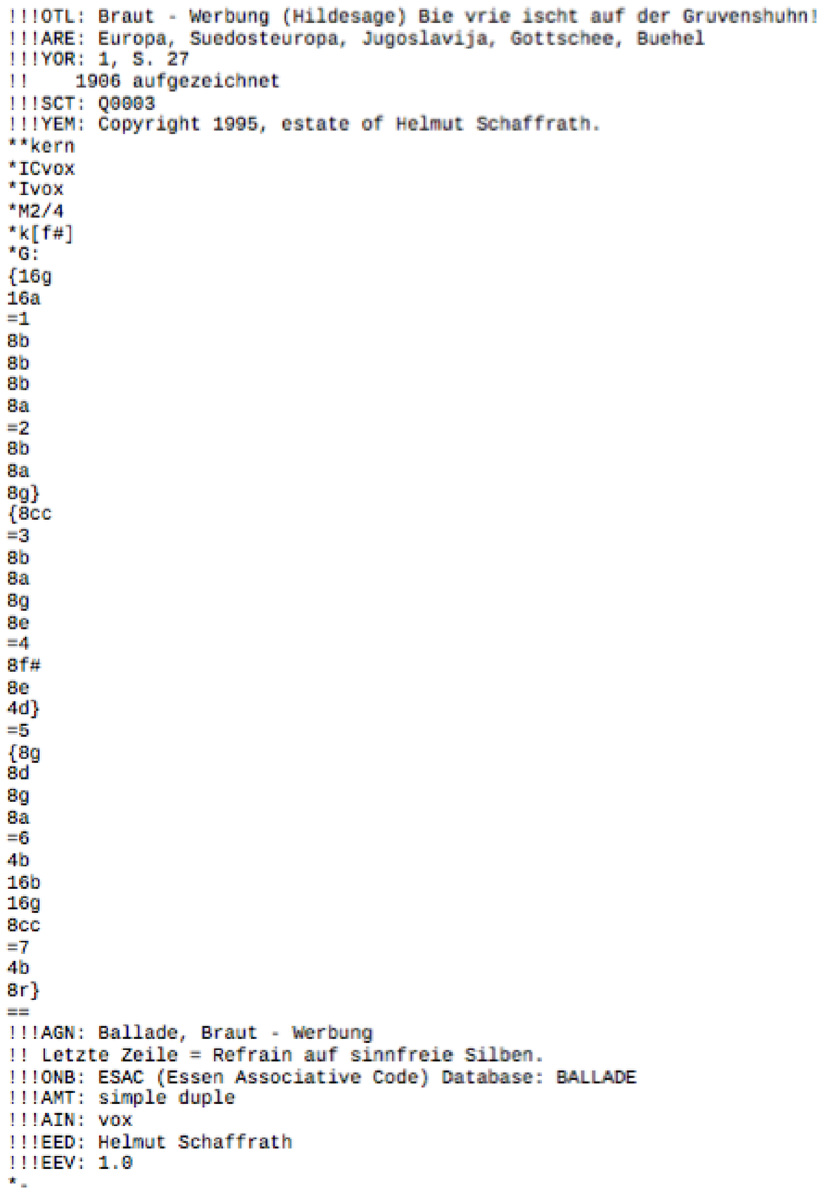
2.2.1 ***Music21*** **library and** *****kern*** **representation**

For music modeling experiments, the data is encoded in ***kern* format using the *music21* python library [1]. This library allows encoding of the pitch, duration, time signatures, timbre, ornaments, articulations, phrase markings, bar lines and other syntactic and structural components of music. The relevant pitch information required for the music pitch prediction experiments are extracted using the *music21* toolkit. The *music21* library Cuthbert and Ariza (2010b) provides suitable functions for extracting and parsing symbolic music scores. Example of the **kern score from the Essen Folk Song Collection are shown in Figures 2.1 and 2.2.

2.3 Inductive Biases, Compositionality and Systematicity

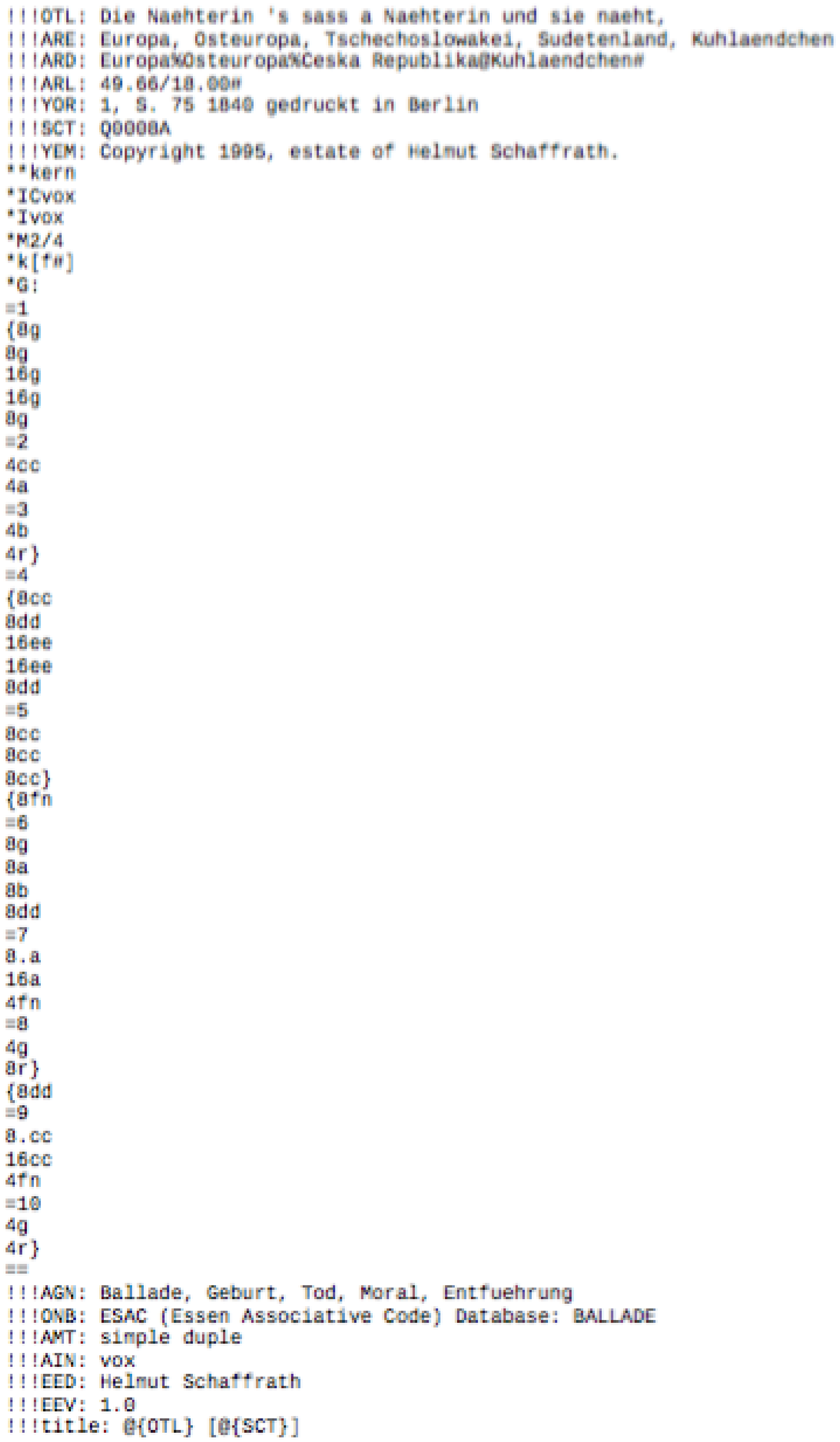
There has recently been an increased interest in inductive biases to improve generalisation in machine learning and specifically neural networks with potential benefits in various applications like relational reasoning (Battaglia et al., 2018), spatial reasoning (Hamrick et al., 2018), learning from few examples (Snell et al., 2017; Wang and Yao, 2019), cognitive modeling (Griffiths et al., 2010), natural language processing (Mitchell

[1] http://web.mit.edu/music21/



```
!!!OTL: Braut - Werbung (Hildesage) Bie vrie ischt auf der Gruvenshuhn!
!!!ARE: Europa, Suedosteuropa, Jugoslavija, Gottschee, Buehel
!!!YOR: 1, S. 27
!!    1906 aufgezeichnet
!!!SCT: Q0003
!!!YEM: Copyright 1995, estate of Helmut Schaffrath.
**kern
*ICvox
*Ivox
*M2/4
*k[f#]
*G:
{16g
16a
=1
8b
8b
8b
8a
=2
8b
8a
8g}
{8cc
=3
8b
8a
8g
8e
=4
8f#
8e
4d}
=5
{8g
8d
8g
8a
=6
4b
16b
16g
8cc
=7
4b
8r}
==
!!!AGN: Ballade, Braut - Werbung
!! Letzte Zeile = Refrain auf sinnfreie Silben.
!!!ONB: ESAC (Essen Associative Code) Database: BALLADE
!!!AMT: simple duple
!!!AIN: vox
!!!EED: Helmut Schaffrath
!!!EEV: 1.0
*-
```

Fig. 2.1 **Kern format of the Jugoslav melody (jugos001.krn) from the Essen Folk Song Collection.



```
!!!OTL: Die Naehterin 's sass a Naehterin und sie naeht,
!!!ARE: Europa, Osteuropa, Tschechoslowakei, Sudetenland, Kuhlaendchen
!!!ARD: Europa%Osteuropa%Ceska Republika@Kuhlaendchen#
!!!ARL: 49.66/18.00#
!!!YOR: 1, S. 75 1840 gedruckt in Berlin
!!!SCT: Q0008A
!!!YEM: Copyright 1995, estate of Helmut Schaffrath.
**kern
*ICvox
*Ivox
*M2/4
*k[f#]
*G:
=1
{8g
8g
16g
16g
8g
=2
4cc
4a
=3
4b
4r}
=4
{8cc
8dd
16ee
16ee
8dd
=5
8cc
8cc
8cc}
{8fn
=6
8g
8a
8b
8dd
=7
8.a
16a
4fn
=8
4g
8r}
{8dd
=9
8.cc
16cc
4fn
=10
4g
4r}
==
!!!AGN: Ballade, Geburt, Tod, Moral, Entfuehrung
!!!ONB: ESAC (Essen Associative Code) Database: BALLADE
!!!AMT: simple duple
!!!AIN: vox
!!!EED: Helmut Schaffrath
!!!EEV: 1.0
!!!title: @{OTL} [@{SCT}]
```

Fig. 2.2 **Kern format of the Czech melody (czech01.krn) from the Essen Folk Song Collection.

et al., 2018), machine translation (Lake and Baroni, 2018; Sutskever et al., 2014), and numeric reasoning (Trask et al., 2018).

Earlier work on systematicity of human language learning and connectionism started with Fodor and Pylyshyn (1988). There have been wider debates on systematicity of neural network learning with claims and counter-claims on their abilities (Christiansen and Chater, 1994; Fodor and McLaughlin, 1990; Frank, 2014; Niklasson and van Gelder, 1994). In the context of abstract pattern learning, there was series of studies triggered by Marcus et al. (1999) with different extended network architectures and evaluation methods (Alhama and Zuidema, 2016, 2019a; Altmann, 1999; Elman, 1999; Shultz and Bale, 2006; Vilcu and Hadley, 2005).

Traditional rule-based approaches do not suffer from the lack of systematicity, but they lack the flexibility of neural networks. Some approaches aim to improve rule based learning with more flexibility, e.g. through probabilistic logic (De Raedt and Thon, 2010; De Raedt et al., 2019). Other studies aim at combining symbolic rule based knowledge with neural networks (Besold et al., 2017; Garcez et al., 2015; Raedt et al., 2019; Vankov and Bowers, 2020). However, this approach has not yet been adopted in the mainstream of machine learning research and applications.

There has been work on different ways of modeling inductive biases in neural networks in recent years, like using matrix factorisation (Neyshabur et al., 2014) and convolutional arithmetic circuits (Cohen and Shashua, 2016a) for computer vision tasks, relational inductive bias models and Bayesian inference in the context of deep reinforcement learning (Gershman and Niv, 2015; Zambaldi et al., 2019) and inductive bias for integrating tree structures in LSTMs by ordering the neurons (Shen et al., 2019).

It has also been identified for a long time that weight initialisation can improve the speed and quality of neural network learning (Le et al., 2015; Nye and Saxe, 2018; Sutskever et al., 2013). Various regularisation methods have also been proposed to improve generalisation (Pachitariu and Sahani, 2013; Sussillo, 2014; Zaremba et al., 2014). Mikolov et al. (2015) use two weight matrices with different learning dynamics to encourage long term memory. Recently, spatial weight priors have been proposed for Bayesian convolutional neural networks by (Atanov et al., 2019). There are works by Demeester et al. (2016) for injecting inductive bias in the form of implication first order rules through an additional regularization term for learning relations between entities in WordNet.

Recent studies have confirmed that state of the art neural networks lack systematic generalisation, specifically recurrent neural networks for a sequence to sequence learning task (Bastings et al., 2018; Lake and Baroni, 2018; Liska et al., 2018), but have stressed the lack of compositionality and generalisation in neural networks (Andreas, 2019; Baroni, 2019; Hupkes et al., 2019; Keresztury and Bruni, 2020; Lake, 2019; Nye et al., 2020; Vankov and Bowers, 2019, 2020).

Other works on incorporating symbolic prior knowledge into neural networks include (Xu et al., 2018) where a loss function based on constraints on the output has been developed as a regularization term and evaluated on structured prediction and multi-label classification tasks. (Minervini et al., 2017) and **?** came up with a method for regularizing multi-layer graph based neural networks using logic based prior knowledge on link prediction tasks. There are other works like Neural Theorem Provers (NTPs) (Rocktäschel and Riedel, 2017) which are proposed to solve large scale knowledge reasoning tasks (Minervini et al., 2019) and systematic generalization (Minervini et al., 2020). In the context of relational reasoning and natural language understanding there are works like (Sinha et al., 2019, 2020) evaluating systematic out of order logical generalization in graph neural networks.

Mitchell et al. (2018) studied among other problems the learning equality of numbers in binary representation, which does not generalise from even to odd numbers as pointed out already by Marcus (2001). They investigate several potential solutions, with a preference for a convolutional approach. Trask et al. (2018) studied the numerical extrapolation behaviour using logic gates based on arithmetic logic units for fundamental numerical operations.

In this research work, we have shown that equality relations and abstract patterns based on equality/identity relations are not generalised by standard feed-forward and recurrent neural network architectures. This failure is independent of various parameters like the vector dimension, amount of training data, train/test split, type of activation function, data representation, vector coverage, and aspects of the network architecture. The proposed models, create an inductive bias that leads to almost perfect generalisation on a number of tasks that relate to equality detection and learning abstract patterns. More details are given in the next chapters.

Chapter 3

Relation Based Patterns Approach

In this chapter, the abstract pattern learning task is formally defined and different network architectures are evaluated to see to what extent recurrent and feed-forward neural networks can learn and generalise abstract patterns based on identity rules. A new approach called Relation Based Patterns (RBP) is proposed as a solution for this task.

3.1 Abstract Pattern Learning Task

The task is to learn rules from sequential data also known as grammar learning task. In the well-known experiments by (Marcus et al., 1999), infants were exposed to sequences of one of the forms ABA or ABB and in the test phase they showed significantly different behaviour depending on whether the sequences exhibited the form they were familiarised with or not. (Marcus et al., 1999) also showed that simple recurrent Elman networks were not able to perform this learning task.

This raises the question of whether and how this general rule learning can be achieved with common neural network types for sequence classification. The rest of the chapter covers the formalisation of this problem, the design of the experiments, and the approaches proposed to solve the rule learning problem which are further extended and applied to other tasks in the next chapters.

3.2 Supervised learning of identity rules

The task in the experiment by (Marcus et al., 1999) is an unsupervised learning task, as the infants in the experiments were not given instructions or incentives. However, most common neural network architectures are designed for supervised learning and there are also natural formulations of abstract pattern recognition as supervised learning task in the form of classification or prediction, which will be used here.

In our case, abstract patterns are defined by identity relations. Expressed in logic, they can be described using the binary equality predicate $eq(\cdot,\cdot)$. For a sequence of three tokens α, β, γ the rule-based patterns ABA and ABB can be described by the following rules:

$$ABA : \neg eq(\alpha,\beta) \wedge eq(\alpha,\gamma) \tag{3.1}$$

$$ABB : \neg eq(\alpha,\beta) \wedge eq(\beta,\gamma). \tag{3.2}$$

These rules are independent of the actual values of $\alpha, \beta,$ and γ, which is why they are called abstract patterns. Concrete patterns, on the other hand, are defined in terms of values of from a vocabulary $a, b, c, \ldots$. E.g., sequences $a**$, i.e. beginning with 'a', or $*bc$, ending with 'bc', can be formulated as follows:

$$a** : \alpha = a \tag{3.3}$$

$$*bc : \beta = b \wedge \gamma = c. \tag{3.4}$$

For the remainder of this thesis, the informal notations ABA and $a**$ are used as far as they are unambiguous in their context.

For classification, the task is to assign a sequence to a class, i.e. ABA or ABB, after learning from labeled examples. For prediction, the task is to predict the next token given a sequence of two tokens in a test case, after exposure in training to sequences of one or more of the classes (e.g. only ABA, or ABB respectively). These tasks are suitable for the most commonly used neural network architectures.

3.2.1 Experimental set-up

Network set-up

The Feed-forward Neural Network (FFNN) (also called Multi-Layer Perceptron) (Rumelhart et al., 1985), the Simple Recurrent Neural Network (RNN, also called Elman network (Elman, 1990b)), the Gated Recurrent Unit (GRU) network (Cho et al., 2014), and the Long Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997a) are used for the experiments. For prediction experiments, RNN and its gated variants GRU and LSTM are used.

The input to the networks is a one-hot encoded vector representing each token with n neurons, where n is the size of the vocabulary. In the case of the FFNN, the whole sequence of 3 tokens is encoded as a vector of size $3n$. For the recurrent models, the tokens are presented sequentially, each as a n-dimensional vector. The number of neurons in each hidden layer are set to (10, 20, 30, 40, 50), using 1 or 2 hidden layers. For the hidden layers, Rectified Linear Units (ReLUs) are used as an activation function in all networks. The output layer uses the softmax activation function. The number of output units is 2 for classification and the size of the vocabulary for prediction. The Adam optimisation method (Kingma and Ba, 2014), using initial learning rates of $0.01, 0.1, 0.2, 0.4$ is considered for all the experiments and trained with the synthetic datasets in one batch. For regularisation Dropout rates of 0.1, 0,2, 0.4 are used and the number of epochs is set to 10, within which all trainings converged.

A full grid search over all hyperparameters using four-fold cross-validation is conducted to optimise the hyperparameters and determine test results. A total of 10 simulations is run for each evaluation and the results are averaged. All experiments have been programmed in PyTorch and the code is publicly available.[1]

[1]https://github.com/radhamanisha1/RBP-architecture

Datasets

For performing the rule learning experiments, artificial data is generated in the form of triples for each of the experiments. The sample vocabulary is considered as $a...l$ (12 characters) for both prediction and classification tasks. The triples are generated in all five abstract patterns: AAA, AAB, ABA, ABB, and ABC for the experiments. The sequences are then divided differently for the different cases of classification. For all the experiments a separate train, validation, and test sets are used with 50%, 25%, and 25% of the data, respectively. All sampling (train/test/validation split, down sampling) is done per simulation. Different train/validation/test splits are experimented in the later sections on vector equality learning task, however, for the abstract pattern learning experiments, the train/validation/test split is always fixed to 50%, 25%, and 25% respectively.

3.2.2 Classification

Firstly, the three different classification tasks are tested as listed below. For the experiments, half of the vocabulary is used for training and the other half is used for testing and validation (randomly sampled). The sequences are divided into two classes as follows, always maintaining an equal size of both classes:

1) ABA/ABB vs other: In task a), class one contains only pattern ABA while the other contains all other possible patterns (AAA, AAB, ABB, ABC) down-sampled per pattern for class balance. The task is to detect whether $eq(\alpha, \gamma) \land \neg eq(\alpha, \beta)$ is true or false. Analogously, the task in b) ABB vs other is to detect $eq(\beta, \gamma) \land \neg eq(\alpha, \beta)$. This case corresponds to the experiment in (Marcus et al., 1999), where only one rule-based pattern type is used for familiarisation.

2) ABA vs ABB: This task is like task 1 above, but only pattern ABB occurs in the second class, so that this task has less variance in the second class. This task is expected to be a bit easier to learn because two equality predicates

$eq(\alpha, \gamma), eq(\beta, \gamma)$ change their values between the classes and are each sufficient to indicate the class.

3) ABC vs other: In this case, class one (ABC) has no pair of equal tokens, while the *other* class has at least one of $eq(\alpha, \beta), eq(\alpha, \gamma), eq(\beta, \gamma)$ as *true*, i.e. detecting equalities without localising them is sufficient for correct classification.

In the experiments, the training converged quickly in all cases and the classification accuracy on the training data was 100%. The results on the test set are shown in Table 3.1. In all cases, the baseline that corresponds to random guessing is 50%. This baseline is only exceeded for task 1) by the RNNs and their gated variants, and even then the accuracy is modest at 55%.

Classification task	FFNN	RNN	GRU	LSTM
1a) ABA/other	50%	55%	55%	55%
1b) ABB/other	50%	55%	55%	55%
2) ABA/ABB	50%	50%	50%	50%
3) ABC/other	50%	50%	50%	50%

Table 3.1 Three classification tasks based on abstract patterns over 10 simulations. The numbers show test set accuracy after a grid search and cross validation as described in section 3.2.1. All values are rounded to the next percentage point.

3.2.3 Prediction

For the prediction experiments two tasks are performed. In task 1) ABA patterns are trained and tested and in task 2) ABB patterns are used. Training and test/validation set use different vocabularies. The training converged quickly in less than 10 epochs, and after training the classification accuracy on the training set is 100%.

The results on the test set are shown in Table 3.2. The baseline is 8.3% as the vocabulary is of size 12. Similar to the classification experiments, again half of the vocabulary (6 values) is used for training and the other half is used for validation/testing. The results show that the tested networks fail completely to make correct predictions.

They perform below the baseline at 0% accuracy, which is mostly because they predict only tokens that appear in the training set but not in the test set.

Prediction task	RNN	GRU	LSTM
1) ABA	0%	0%	0%
2) ABB	0%	0%	0%

Table 3.2 Prediction results for two different abstract patterns. The numbers show test set prediction accuracy after a grid search and cross validation as described in section 3.2.1.

3.2.4 Discussion

The results show clearly that FFNNs, RNNs, GRUs and LSTMs do not learn general abstract patterns based on identity rules. This agrees with the previously reported experiments by Marcus et al. (1999). However, since there was some conflicting evidence in the literature, the clarity of the outcome was not expected.

Questions raised

This result raises the question of why these neural networks do not learn to generalise abstract patterns from data. There are two aspects worth considering for an explanation: the capacity of the network and the necessary information for the network to solve the problem.

Regarding the capacity: the solution to the task is in the hypothesis space of the neural networks, since proofs exist of universal approximation properties for feed-forward networks with unbounded activation functions (Leshno et al., 1993) and of Turing-completeness for recurrent networks (Siegelmann and Sontag, 1995). A constructive solution is presented below, putting that result into practice, with a design of network instances that solve the problem.

The relevant question, as has been pointed out by Alhama and Zuidema (2016), is therefore why learning with backpropagation does not lead to effective generalisation here. There are three different steps that are necessary to detect identity rules: a comparison of input neurons, a comparison of tokens, represented by multiple neurons, and a mapping of comparison results to classes or predictions.

Vocabulary hypothesis

A possible reason for the failure of the networks to generalise may be due to the vocabulary hypothesis. It is based on the separated vocabulary in one-hot encoded representation. This leads to some input neurons only being activated in the training set and some only in the validation and test sets.

In order to learn suitable weights for an input comparison, there would have to be a suitable gradient of the weights of the outgoing connections from these inputs. If parts of the vocabulary do not appear in the training data, i.e. the activation of the corresponding input neurons is always zero during training, the weights of their outgoing connections will not be adapted. Therefore it is expected that the separation of the vocabulary prevents generalisation from the training to the test set as the weights going out from neurons that are used during testing have not been adapted by the gradient descent. Based on this consideration another experiment with a shared vocabulary was conducted.

This experiment is called ABA-BAB vs other. The vocabulary is represented as $a...l$ (12 letters) for this task with train/validation/test split as 50%/25%/25%. The same vocabulary is now used for training, validation, and testing, but different sequences of the form ABA that use the same tokens between the training and validation/test sets are separated. E.g., if the *ded* is in the test set, then *ede* is in the training or validation set, so that there is no overlap in terms of actual sequences. Like in classification experiment 1), training converged quickly and resulted in perfect classification performance on the training set.

Classification task	FFNN	RNN	GRU	LSTM
ABA-BAB vs other	50%	50%	50%	50%

Table 3.3 Classification results on test sets with the same vocabulary used in test, validation and training set.

The results on the test set presented in Table 3.3 show performance at the baseline with no evidence of generalisation. This shows that activating all inputs by using a shared vocabulary is not sufficient to enable generalisation in the learning process.

Other explanations

A second potential problem is which neurons should be compared. The FFNN has no prior information about neurons belonging to the same or different tokens or about which input neurons correspond to the same token values. In the RNN, one token is presented per time step, so that a comparison between the previous hidden state and the current input is possible as the same neurons are activated. However, with a full set of connections between the previous hidden layer and the current, there is no reason that relations between the same neurons at different time steps would be processed differently from those between different neurons.

On the other hand, if the representation includes the information of which tokens are identical or different, then all the information needed for a mapping is available, as these are the relations in which our defining rules are formulated (e.g. ABA is defined as $eq(\alpha, \gamma) \wedge \neg eq(\alpha, \beta)$). This idea has led to the Relation Based Pattern (RBP) model that is introduced in the next section and then evaluated with respect to its effect on both abstract and concrete pattern learning.

3.3 Design of Relation Based Pattern models

To address the inability of neural networks to generalise rules in neural network learning, the Relation Based Pattern (RBP) model is developed as a constructive solution, where the comparisons between input neurons and between tokens and the mappings to outputs are added as a predefined structure to the network. The purpose of this structure is to enable standard neural networks to learn abstract patterns based on the identity rules over tokens while retaining other learning abilities.

In the RBP model there are two major steps. The first step is defining comparison units for detecting repetitions, called DR units, and the second step is adding the DR units to the neural network.

3.3.1 Comparison units

Comparing neurons

The input to the network is a one-hot encoded vector of the current token along with the $n-1$ previous vectors for a given context length n (in this study context length 3 for classification and 2 for prediction). The comparison units, called DR units (differentiator-rectifier) are used here. As the name suggests, they apply a full wave rectification to the difference between two inputs:

$$f(x,y) = |x-y|$$

The first level of DR units are DR_n units that are applied to every pair of corresponding input neurons (representing the same value) within a token representation, as shown in Figure 3.1.

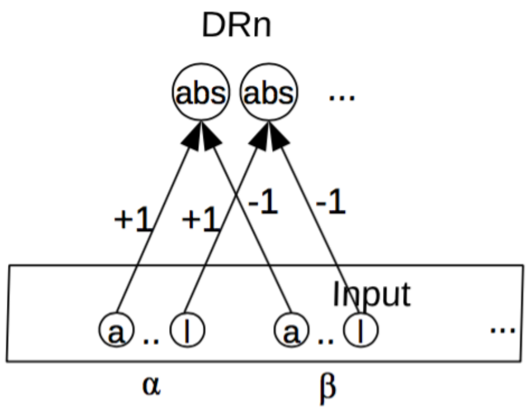


Fig. 3.1 DR_n units comparing related inputs with an absolute of difference activation function. In one-hot encoding there are k DR_n units for every pair of input tokens, where k is the vocabulary size.

Comparing tokens

The next level of DR units are the DR_p units that sum the activations of the DR_n values that belong to one pair of tokens. Based on the sequence length n and vocabulary size k, a total of $k \times n(n-1)/2$ DR_n units are created for all the possible pairs of tokens and i.e. in our classification example, for a sequence of 3 tokens and a vocabulary size of 12, there are $12 \times 3(3-1)/2 = 36 \times 3$ DR_n units. All the DR_n units for a pair of tokens are then summed in a DR_p unit using connections with a fixed weight of $+1$. E.g. there are $5 \times (5-1)/2 = 10$ DR_p units for a context of length 5. Figure 3.2a shows the network structure with DR_n and DR_p units.

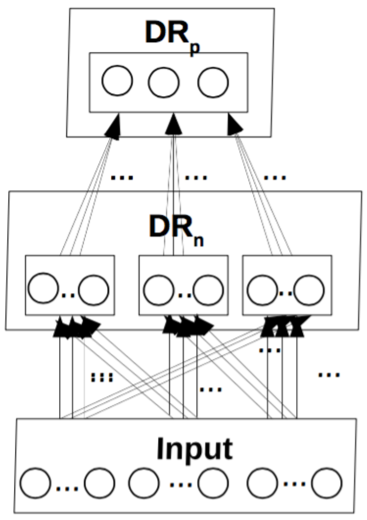
For the prediction case, the same approach is used to represent the difference between each input token and the next token (i.e., the target network output). There are n DR_pout units created that calculate the difference between each input in the given context and the next token. There are $k \times n$ DR_nout units that compare the corresponding neurons for each pair of input/output tokens, in the same way as for the pairs of input tokens. The overall network structure is shown in Figure 3.2b.

3.3.2 Neural network integration

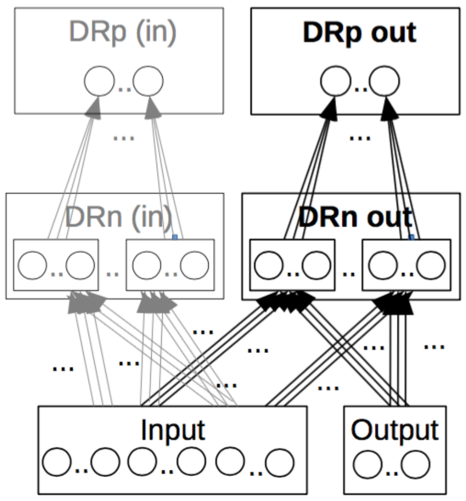
DR units (DR_n and DR_p) are combined with the neural network models in early, mid and late fusion approaches. The weights that connect DR_n units to input and output, and the DR_n to DR_p units and the offset layer are fixed, all other weights that appear in the following models are trainable with back-propagation. The three RBP approaches are called as RBP Early Fusion (two variants RBP1n and RBP1p), RBP Mid Fusion and RBP Late Fusion as explained below.

Early Fusion (RBP1n/p)

In this approach, DR_n or DR_p units are added as additional inputs to the network, concatenated with the normal input. For the sake of brevity, RBP1n is used when DR_n units are added in Early Fusion and RBP1p is used when DR_p are added in Early



(a) The DR_p and DR_n units that are used in the RBP1 and RBP2 structures with $3 \times k$ DR_n and 3 DR_p units for a vocabulary size k and sequence length 3.



(b) The DR_{out} structure for detecting repetitions between input and target. The DR_pout values are calculated at training time and a model is trained to predict them conditional on DR_pin (see Figure 3.5).

Fig. 3.2 DR_n and DR_p units for inputs (all RBP) and outputs (RBP Late Fusion).

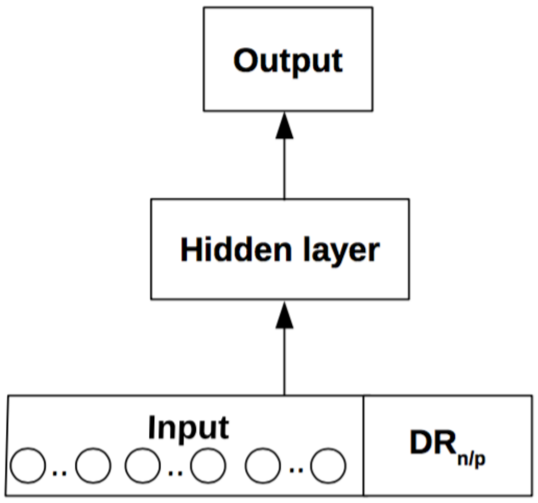
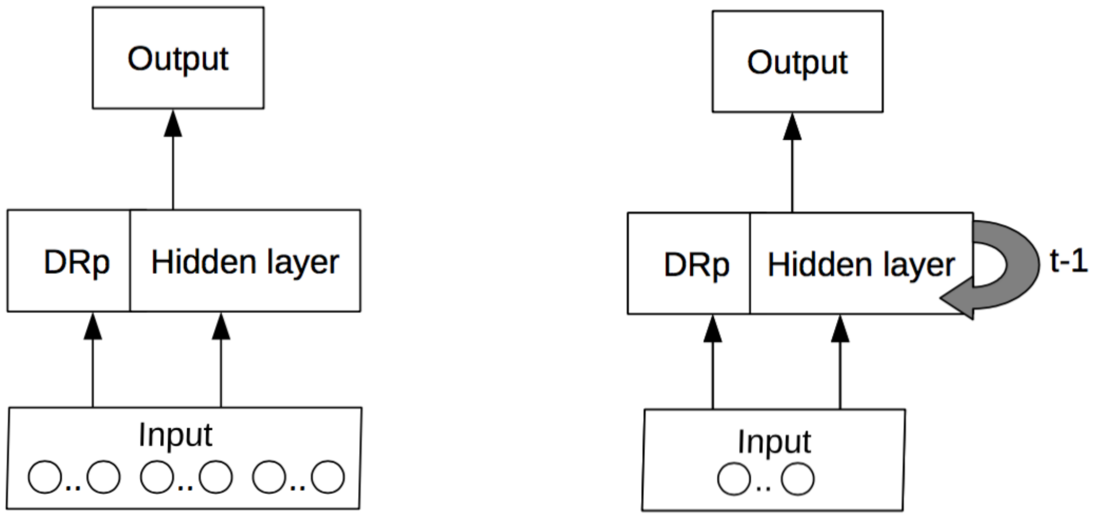


Fig. 3.3 Overview of the RBP1n/RBP1p structure.



(a) RBP Mid Fusion (a) (b) RBP Mid Fusion (b)

Fig. 3.4 Overview of RBP Mid Fusion approaches, where the $DR_{p}out$ units are concatenated to the hidden layer. In the case of recurrent network, an additional context from the previous layer is there, the rest of the network is the same as shown in (a) where DR_p units are added to the first hidden layer. DR_p units are calculated using the tokens in the input layer. The only difference from Early Fusion is that here the DR_p are added to the hidden layer unlike previously where they are added to the input layer.

Fusion. In Figure 3.3, the RBP1n/p structure is depicted. Early fusion is used in both the prediction and classification tasks.

Mid Fusion

The DR_p units are added to the hidden layer. Figure 3.4a shows the mid fusion structure for the feed-forward network and Figure 3.4b for the recurrent network respectively. In Figure 3.4a, DR_p units are calculated using the tokens in the input and are added to the first hidden layer. For example, in Figure 3.4a the input has three tokens, the DR_p units are calculated for the three tokens and added to the first hidden layer. In the case of RNNs, there is an additional context layer i.e. a recurring connection, which is represented as a loop as shown in Figure 3.4b. This context has the information from the previous hidden layer at time step $t-1$. The input is passed one token at a time, and the DR_p units are calculated using the tokens from

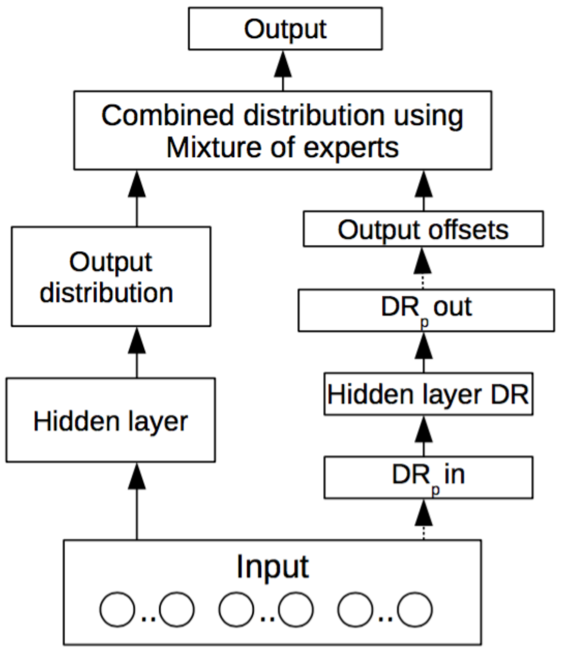


Fig. 3.5 Overview of the RBP Late Fusion approach. The $DR_p in$ values are calculated as in RBP Mid Fusion. From there, a fully connected layer is used to predict $DR_p out$ (trained with teacher-forcing). The predicted $DR_p out$ values are mapped back to the vocabulary (based on the context tokens) and used as probability offsets in a mixture of experts with the standard neural network in the left part of the diagram. All connections are trainable except *Input* to $DR_p in$ and $DR_p out$ to *Output offsets* (dotted arrows).

the previous time step $t-1$, and added to the hidden layer. The other set up and DR_p calculation remains the same as in Figure 3.4a, however the only difference is the recurrent connection from the previous hidden layer. The RBP Mid Fusion approach is used for classification and prediction tasks.

Late Fusion

In this approach, same structure as in RBP Mid Fusion (which is called as $DR_n in$ and $DR_p in$ in this context) is used, and in addition to that the probability of identity relations between the input and the output is also estimated, i.e., that the token in the current context is repeated as the next token. A structure called $DR_p out$ is used for this, which is projected back to the vocabulary space, to generate a probability offset for the tokens appearing in the context.

Figure 3.5 gives an overview of the RBP late fusion scheme. The DR_pin units detect identities between the input tokens in the current context as before. The DR_pout units model the identities between the context and the next token, as shown in the Figure 3.4b, where repetition is encoded as 1, and a non-repeated token as a -1.

During training teacher-forcing is used, i.e., the values of the DR_pout units are set to the true values. A feed-forward neural network with one hidden layer is used to learn a mapping from the DR_{in} to the DR_{out}. This gives us an estimate of the DR_{out} units given the DR_{in} units. The DR_{out} values are then normalised subtracting the mean, and then mapped back to the output space (the one-hot vocabulary representation), using a zero value for the output values that do not appear in the input. These output offsets are then combined in a weighted sum (mixture of experts) with the output distribution estimated by the standard normal network (on the left side in Figure 3.5). The weights in the mixture are trainable. The outputs from the combined distribution of mixture of experts are clipped between [0,1] and renormalised. The final output distribution is a softmax layer providing the probability distribution over the vocabulary for the next token.

3.4 Neural networks with RBP structures

In the following section, the experiments from section 3.2 are repeated and the neural networks are tested by adding the RBP structures. For convenience and better comparison previous results in the tables are repeated again in this section.

3.4.1 Classification experiments

This experiment is analogous to the first classification experiment. In the case of the feed-forward network, RBP Mid Fusion (a) was used in the mid fusion approach and for recurrent network, RBP Mid Fusion (b) was used. RBP Late Fusion is designed to model the similarity between elements of the input context and the output, which in a classification setting is not possible as the output is a class vector which is of a different nature than the input tokens and normally of different dimensionality. RBP Late Fusion is therefore not applied in the classification experiments. The neural networks are again trained for 10 epochs and all networks converged to perfect classification on the training set. Table 3.4 provides the overall test accuracy for the three approaches.

Task	RBP	FFNN	RNN	GRU	LSTM
1a) ABA vs other	-	50%	55%	55%	55%
	RBP1n	50%	55%	55%	55%
	RBP1p	65%	70%	70%	70%
	RBP Mid Fusion	100%	100%	100%	100%
1b) ABB vs other	-	50%	55%	55%	55%
	RBP1n	50%	55%	55%	55%
	RBP1p	65%	70%	70%	70%
	RBP Mid Fusion	100%	100%	100%	100%
2) ABA vs ABB	-	50%	50%	50%	50%
	RBP1n	50%	60%	65%	65%
	RBP1p	75%	75%	75%	75%
	RBP Mid Fusion	100%	100%	100%	100%
3) ABC vs other	-	50%	50%	50%	50%
	RBP1n	55%	65%	65%	65%
	RBP1p	55%	70%	70%	70%
	RBP Mid Fusion	100%	100%	100%	100%
4) ABA-BAB vs other	-	50%	50%	50%	50%
	RBP1n	55%	72%	75%	75%
	RBP1p	69%	74%	75%	76%
	RBP Mid Fusion	100%	100%	100%	100%

Table 3.4 Classification experiments with RBP: test accuracy for the different models and tasks, as explained above. Results with '-' in the RBP column are the same as in section 3.2 and shown here again for comparison.

The results with RBP1n models already show some improvement over the baseline in most configurations, but the result are only slightly above the standard networks, with RNNs, GRUs and LSTMs benefiting more than FFNN. This supports our hypothesis that learning to compare corresponding input neurons is a challenging task for neural networks. However, the results show that providing that comparison is not sufficient for learning identity rules.

RBP1p structures also aggregate all the l DR_n neurons that belong to a pair of input tokens. The results show that providing that information leads to improved accuracy and provide evidence that this aggregation is another necessary step that the networks do not learn reliably from the data.

The RBP Mid Fusion models enable the neural networks to make predictions and classifications that generalise according to identity rules that they learn from data. The RBP Mid Fusion leads to perfect classification for all network types tested. This confirms the design consideration that comparing pairs of tokens provides the relevant information in the form required for classification, as the classes are defined by *equals* relations, so that the activations of the DRp units are directly correlated with the class labels.

A surprising result is the big difference between the generalisation using the RBP1p and the RBP Mid Fusion structures. They both provide the same information, only in different layers of the network, but RBP1p only reaches at most 75% with a 50% baseline. It is hypothesized that the additional expressive power provided by the non-linearities in the hidden layer here provides effective learning. This effect deserves further investigation.

3.4.2 Prediction experiments

Here two experiments are performed separately on ABA and ABB patterns as in Experiment done on prediction. The tasks are the same as previously and the neural networks are trained again for 10 epochs after which all networks had converged to perfect prediction accuracy on the training data. Table 3.5, summarises the accuracy for RNN, GRU and LSTM without RBP, and with RBP1n, RBP1p Early Fusion, RBP Mid Fusion, and RBP Late Fusion.

Overall, it is observed that only the LSTM benefits from RPB Early Fusion, and RBP Mid Fusion structures, all other networks can apparently not make use of the

Pattern	RBP	RNN	GRU	LSTM
1) ABA	-	0%	0%	0%
	RBP1n (Early Fusion)	0%	0%	16%
	RBP1p (Early Fusion)	0%	0%	18%
	RBP Mid Fusion	0%	0%	20%
	RBP Late Fusion	100%	100%	100%
2) ABB	-	0%	0%	0%
	RBP1n (Early Fusion)	0%	0%	17%
	RBP1p (Early Fusion)	0%	0%	20%
	RBP Mid Fusion	0%	0%	22%
	RBP Late Fusion	100%	100%	100%

Table 3.5 Test set accuracy in prediction experiments for patterns ABA and ABB. As before, results are averaged over 10 simulations and rounded to the nearest decimal point. Results with '-' in the RBP column are the same as in section 3 and shown here again for comparison.

information provided. The RBP Late Fusion model, on the other hand, leads to perfect classification on our synthetic dataset.

The interpretation of this study is that the standard recurrent networks do not learn the more complex mapping that prediction requires, as not only recognition of a pattern but also selecting a prediction on the basis of that pattern is required. The somewhat better results of the LSTM networks are interesting and show that their gating mechanisms learn some of the necessary mapping. In the RBP Late Fusion model, the mapping between the identity patterns and back to the vocabulary adds considerable prior structure to the model and it is very effective in achieving the generalisation of rule-based patterns.

3.5 Learning Identity/Equality Rules of binary vectors

To show the effectiveness of RBP models, further experiments with RBP structures are performed on learning equality (identity) of binary vectors in different dimensions and other numerical tasks like digit reversal and parity which are summarized below.

- Different sets of experiments for estimating vector equality in relation to vector dimensionality, vector coverage i.e. distribution of vectors in training, data size and data structure.
- Experiments on numeric comparison of vectors, classification by sum of all digits >= 3, recognising digit reversal by (swapping least significant bit with most significant bit).

Basic relations such as equality are fundamental to the data structures in many learning tasks. Although equality as such is typically not learned from data, equality or approximate equality may be embedded as part of other tasks. The equality function is clearly in the hypothesis space of feed-forward neural networks (FFNNs) as they are universal approximators (Leshno et al., 1993). However, this does not mean that the function can be found by training with gradient descent. Marcus et al. (1999) and Marcus (2001) already highlighted that learning identity relationships within sequences using neural networks does not generalise to unseen data, while 7-month old infant learn and generalise these rules within minutes. This problem is evidently relevant in tasks like natural language understanding and general reasoning tasks. Therefore, in this part of the study we focus now exclusively on the case of feed-forward neural networks and the equality of two binary vectors. We evaluate why standard neural networks do not learn equality.

The task here is a simple one: detecting if two binary vectors v_1, v_2 with n dimensions are equal. v_1 and v_2 are concatenated as input to a neural network. Different experiments are performed by testing with various factors to determine the effect of binary vectors on generalisation. For this task, a plain FFNN network is used with one hidden layer and ReLU activation. FFNN is chosen because the task of learning vector equality is relatively simple one and do not require any recurrent neural networks or any of the variants. RBP in Early Fusion and Mid Fusion are used for these experiments, as RBP in Late Fusion is not applicable here.

Throughout all the experiments, the neural network is trained for 20 epochs, which led to convergence in all cases and 100% average training accuracy. A total of 10 simulations are run for each configuration and the accuracies are averaged. The differences between model accuracies are tested with a Wilcoxon Signed Rank Test with threshold $p = .05$, over the results of the simulations This test does not assume a normal distribution and tests for different medians.

3.5.1 Inductive bias creation with DR units

We use differential rectifier (DR) units that compare input values by calculating the absolute difference: $f(x, y) = |x - y|$ as defined in Section 3.3 above. One DR unit for every vector dimension is created with weights from the inputs to the DR units fixed at 1, thus learning the suitable summation weights for the DRs is sufficient for creating a generalisable equality detector.

There are two ways of integrating DR units into the neural networks for the vector equality task: *RBP Early Fusion* and *RBP Mid Fusion* also called as RBP1 and RBP2. In Early Fusion, DR units are concatenated to input units 3.6b), and in Mid Fusion they are added to the hidden layer 3.6c). In both cases the existing input and hidden units are unchanged. The plain network without DR units is shown in 3.6a). These are similar to the RBP Early and Mid Fusion approaches described in Section 3.3.2. The diagrams are reproduced here to give a better clarity of how RBP is modeled in the case of classifying binary vectors.

3.5.2 Variations of the data size

Vector dimensionality

Pairs of random binary vectors with dimensionality n between 2 and 100 are generated. For $n < 10$ all the possible binary vectors are used to generate equal pairs, i.e. 2^n equal pairs, and random vectors for the unequal pairs. For $n \geq 10$, a random class balanced selection of 10000 vectors is used. The same number of unequal vector pairs

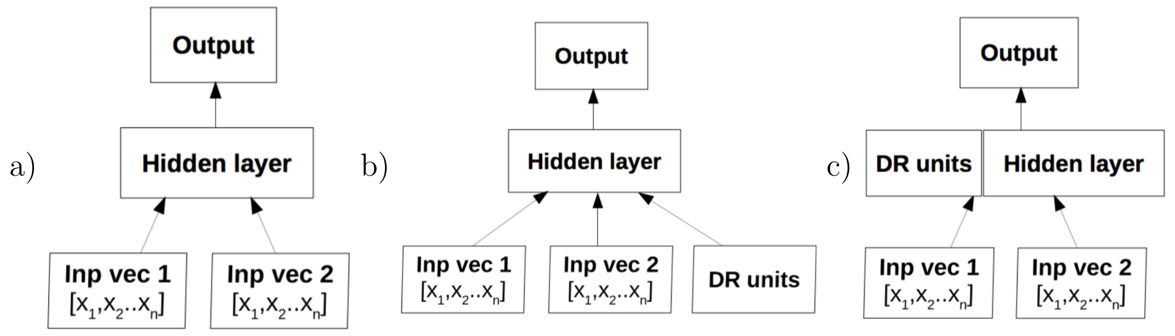


Fig. 3.6 Network architectures: a) standard feed-forward network, DR integration with b) early fusion and c) mid fusion. The DR units receive their input in both cases from *vec* 1 and *vec* 2.

are randomly generated. Then a stratified sampling is used to split the data i.e. 75:25 into train and test set. The results are shown in Table 3.6.

It is observed that the standard FFNNs barely exceed chance level (50%). The early fusion model improves results, but never reaches full generalisation. The Mid Fusion reaches close to perfect test performance. The significant differences is tested between the lowest dimensionalities ($n = 2, 3$) vs the highest ($n = 90, 100$), with a sample size of 20 each. The significant differences are found only for the RBP in Early and Mid Fusion ($p < .01$), where accuracy is better for lower dimensionalities, but not for the plain FFNN.

Training data size

Here the effect of training data size on the performance is studied. For this the training data size is varied by keeping the test set and all other parameters constant. The training data sizes of 1% to 50% is used (in relation to the totally available data as defined above) and the accuracy achieved in various conditions is plotted in Figure 3.7. The RBP Mid Fusion network reaches 100% accuracy from 10% data size on while the FFNN shows only small learning effects.

Vector Dimension	Plain FFNN	RBP Early Fusion	RBP Mid Fusion
n=2	52%	66%	100%
n=3	55%	65%	100%
n=5	51%	67%	100%
n=10	51%	65%	100%
n=20	49%	63%	100%
n=30	50%	65%	100%
n=40	50%	64%	100%
n=50	51%	65%	100%
n=60	48%	64%	100%
n=70	50%	62%	100%
n=80	50%	63%	100%
n=90	49%	62%	100%
n=100	50%	64%	100%
SD	1.59	1.23	0.07

Table 3.6 Accuracy of the different network types on pairs of vectors of different dimensions. The joint train and test data covers all possible equal vector pairs for $n < 10$, and a random, class-balanced selection of 10000 vector pairs where $n \geq 10$. The standard deviation (SD) is given in percentage points.

Effect of train/test split

In order to further study the effect of increased training data higher proportions of the total data is used as training data, which means that the test data size should be reduced. The train/test split is varied from 75/25% of training and testing data up to 95/5% on pairs of 10-dimensional vectors. The results are given in Figure 3.8, showing gradual improvements of the RBP Early Fusion with DR units and plain FFNN models, but even with 95% of all possible combinations in the training set the accuracy never exceeds 72% and 68%, respectively.

Vector coverage

A possible hypothesis for the results of the FFNN is that the coverage of the vectors in the training set plays a role. The vectors are not shared in equal pairs between training and test set, as it would mean to train on the test data. Instead a training set

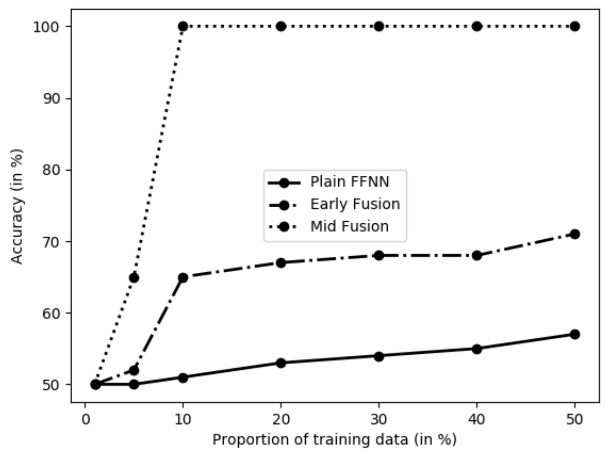


Fig. 3.7 Accuracy of plain FFNN and RBP variants for 10-dimensional binary vectors when varying the distributions of training data from 1% to 50%, keeping the testing data fixed at 50%.

is created that contains in its unequal pairs all vectors that appear in the test set. The results are shown in column a) of Table 3.7. A training set is also created where each vector appeared as above, but in two pairs, one in position v_1 and once in v_2. The results are shown in column b) of Table 3.7. The results in both cases are similar to those without this additional coverage in Table 3.6.

Type	Plain FFNN	RBP Early Fusion	RBP Mid Fusion
a) one	50% (1.56)	75% (1.17)	100% (0.03)
b) both	52% (1.56)	87% (1.14)	100% (0.04)

Table 3.7 Test set accuracy (standard deviation) of plain FFNNs and with DR units i.e. RBP in Early and Mid Fusion for test set vectors appearing in training in a) one position, or b) both positions for $n = 10$. Accuracies for a) and b) are not significantly different to the case of random vectors for plain FFNNs, but the improvement is significant for Early and Mid Fusion RBP models when compared to models without RBP.

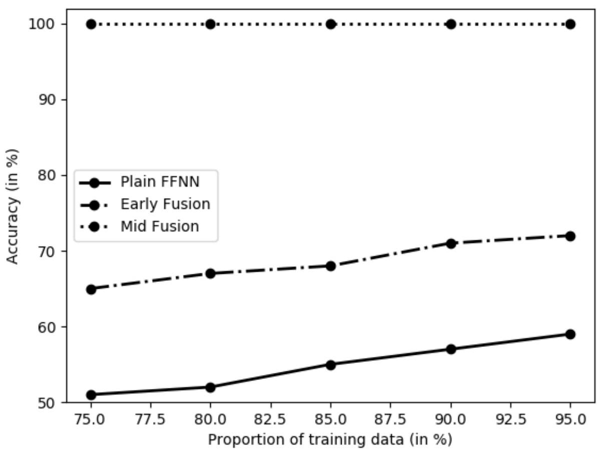


Fig. 3.8 Accuracy of plain FFNN and DR variants for 10-dimensional vectors when varying the size of training data set from 75% to 95% of the total, with the remaining data used for testing.

3.5.3 Variations of the network architecture

Network depth

Given the general success of deep networks, it is hypothesised that depth might help in learning a generalisable solution and thus testing networks with greater numbers of hidden layers is beneficial. In the mid fusion architecture, the DR units are always concatenated to the first layer that is connected to the inputs. The results show some improvement, but not to a large extent, as the number of hidden layers are increased as given in Table 3.8. The difference between the shallow networks (1 and 2 layers) and the deep networks (4 and 5 layers) is statistically significant ($p < .01$). This is an interesting result, as the additional depth is not necessary for representing a generalisable solution, but does help somewhat to find such a solution. It is not obvious, why deeper networks learn somewhat more generalisable solutions, given that Early Fusion models, where there is one more layer between the DR units and the outputs, are doing worse than the Mid Fusion models.

Number of hidden layers	Plain FFNN	RBP Early Fusion	RBP Mid Fusion
h=1	55%	65%	100%
h=2	57%	69%	100%
h=3	58%	69%	100%
h=4	60%	72%	100%
h=5	61%	73%	100%
SD	1.57	1.59	0.05

Table 3.8 Test accuracy for different numbers of hidden layers with 10 neurons each, for vector dimension *n=3*. Deeper networks are significantly better than shallower ones (see text for details).

Hidden layer width

Based on the positive effect of a larger model with more parameters, the performance of the network is also evaluated using a single hidden layer and by varying the number of neurons in that layer. The hidden layer size of 10 to 100 is considered, again with and without DR units, the results are tabulated in Table 3.9. The observation here is that the larger models do not improve in performance when width is changed instead of depth. There is no significant difference between the two smallest networks ($h_n = 10, 20$) and the largest ($h_n = 80, 100$) for Plain FFNNs, while for the RBP models the smaller networks perform significantly better.

Hidden layer size	Plain FFNN	RBP Early Fusion	RBP Mid Fusion
$h_n = 10$	50%	65%	100%
$h_n = 20$	49%	63%	100%
$h_n = 30$	51%	61%	100%
$h_n = 40$	45%	65%	100%
$h_n = 50$	47%	65%	100%
$h_n = 80$	51%	62%	100%
$h_n = 100$	50%	64%	100%
SD	1.62	1.23	0.04

Table 3.9 Accuracy of different network types for *n=3*. The networks contain a single hidden layer of variable size h_n.

The overall accuracy with RBP Early Fusion does not seem to improve with the hidden layer size and the performance varies and doesn't increase with the increase in the hidden layer size. For example, as seen in Table 3.9, the accuracy with 10 hidden layers is similar to the one with 40 and 50. However, RBP in Mid Fusion leads to 100% accuracy in all the cases.

3.5.4 Other factors

Activation function

Another approach to change the learning behaviour is using different activation functions in the hidden layer. For *n=3*, the networks are evaluated with ReLU, Sigmoid and Tanh activations. The results are given in Table 3.10 and they show that the type of activation in the hidden layer has small positive effect in the overall accuracy, which is significant for all models.

Activation function	Plain FFNN	RBP Early Fusion	RBP Mid Fusion
ReLu	55% (1.38)	65% (1.23)	100% (0.05)
Sigmoid	58% (1.28)	69% (1.18)	100% (0.03)
Tanh	58% (1.29)	69% (1.17)	100% (0.03)

Table 3.10 Test accuracy (standard deviation) of the network for different activation functions for vector dimension *n=3*

Data representation

A different data representation of the binary vectors is tested where 0;1 has been replaced with -1;1, like the activation levels networks binarised with the sign function but only applied to the inputs like Courbariaux et al. (2016). The results of the accuracy for *n=3* with and without DR units are given in Table 3.11. An improvement is seen with the DR units, it is relatively small but significant across all models.

Type	Plain FFNN	RBP Early Fusion	RBP Mid Fusion
a) 0/1	55% (1.23)	65% (1.09)	100% (0.03)
b) -1/1	58% (1.19)	69% (1.07)	100% (0.03)

Table 3.11 Accuracy of the network for data representation. a) standard 0/1 representation and b) 'sign' -1/1 representation for vector dimension *n=3*.

Further tests are performed on FFNN with 5 hidden layers, sigmoid activation function and -1/1 representation. Apparently the gains do not accumulate, as the test set accuracy was only 58%.

3.6 Numerical Tasks

Given the positive results for the DR mid fusion architecture for vector pairs, we want to test whether the DR units have a, possibly negative, effect on other learning tasks. The study of vector equality learning is now extended to numerical problem learning with the RBP structures for solving digit reversal and parity. The tests are to check a) numeric comparison of the two vectors in the pair, We also tested a task that is not a comparison of the two vectors in the pair, by b) calculating the digit sum. We classify by checking if the digit sum is ≥ 3. The results are shown in Table 3.12. In both a) and b) it is seen that the performance is actually not hindered but helped by the DR units adding through RBP in Early and Mid Fusion.

There are further tests performed on tasks the DR architecture was not explicitly designed for: i.e. c) digit reversal ($v_1 = flip(v_2)$) and d) parity checking (digit sum mod 2). The DR units organised per corresponding input neurons do not deliver a perfect solution here, but still lead to better results than a plain FFNN. The results are shown in Table 3.12.

The accuracy and standard deviation for the numerical tasks are summarized in Table 3.12. The differences of Plain FFNN vs Early Fusion and Early vs Mid Fusion DR are statistically significant for all tasks.

Task	Plain FFNN	RBP Early Fusion	RBP Mid Fusion
a)	75% (1.08)	92% (0.94)	100% (0.03)
b)	77% (1.05)	82% (1.03)	100% (0.03)
c)	50% (1.62)	55% (1.53)	58% (1.50)
d)	51% (1.62)	55% (1.53)	62% (1.45)

Table 3.12 Test set accuracy (standard deviation) of FFNN without and with DR units for a) numeric comparison ($v_1 \geq v_2$), b) digit sum ≥ 3, c) inversion of digit order and d) parity check.

The digit reversal and parity problem do not generalise as RBP is not explicitly designed to model both the tasks. To evaluate the problem, $n = 6$ dimension vectors are used and certain experiments are performed. The effect of increase in the amount of training data and deeper networks are tested for digit reversal and parity recognition task. The results for the increase in the data size achieved in various conditions are shown in Tables 3.13 and 3.14.

There are no standard deviation values reported in Tables 3.13 and 3.14 as the experiments had only a single run here and not averaged across 10 simulations like other experiments.

Train/test split	Plain FFNN	RBP Early Fusion	RBP Mid Fusion
75%-25%	50%	55%	58%
80%-20%	51%	55%	58%
85%-15%	51%	58%	60%
90%-10%	52%	58%	61%
95%-5%	52%	58%	62%

Table 3.13 Accuracy in digit reversal of FFNN for 6-dimensional binary vectors after varying the distributions of training data from 75%-95% with the remaining data used for testing accordingly.)

As can be seen from the results above, increasing the amount of training data, leads to some improvement especially significant in the case RBP Mid Fusion, but does not lead to perfect generalisation. The effect of train/test data distribution doesn't seem

Train/test split	Plain FFNN	RBP Early Fusion	RBP Mid Fusion
75%-25%	51%	55%	62%
80%-20%	51%	55%	61%
85%-15%	53%	58%	65%
90%-10%	55%	59%	68%
95%-5%	58%	65%	72%

Table 3.14 Accuracy in parity recognition of FFNN for 6 dimensional binary vectors after varying the distributions of training data from 75%-95% with the remaining data used for testing accordingly.)

to make much difference, as can be seen in Table 3.14 where for example, having a train/test split of 80%/20% doesn't improve over the case 75%/25% split. One point to note here is that the model accuracies are reported over a single run, and not averaged across 10 simulations as done in the other numerical tasks, but the overall effect of varying train/test data distribution is negligible. One possible reason for that could be that in RBP only a pair of input vectors are connected to a DR unit. The other possible connections which include all pairs of input neurons are not connected to the DR unit.

So an extended DR structure is tested, where every possible pair of input neurons are connected to a DR unit. This extends the performance, leading to perfect generalisation on both tasks, as shown in Table 3.15. However, this gain comes at the expense of $O(n^2)$ DR units.

Task	Plain FFNN	RBP Early Fusion	RBP Mid Fusion
a) Digit Reversal	50% (1.63)	76% (1.04)	100% (0.05)
b) Parity check	51% (1.63)	82% (1.02)	100% (0.06)

Table 3.15 Accuracy of the network a) digit reversal and b) parity using DRs for all the pairs on 6 dimensional binary vectors.

The results presented so far were all obtained with synthetic data where classification was exclusively on rule-based abstract patterns. This raises the question whether the

RBP will impede recognition of concrete patterns in a mixed situation. Furthermore, it is beneficial to know whether RBP is effective with real data where the abstract and concrete patterns may interact. Therefore we perform experiments using mixed abstract and concrete patterns.

3.7 Mixed abstract and concrete patterns

An experiment was conducted where the classes were defined by combinations of abstract and concrete patterns. Specifically 4 classes were defined based on the abstract patterns ABA and ABB combined with the concrete patterns $a**$ and $b**$. E.g., the class $ABA, a**$ can be expressed logically as

$$eq(\alpha,\gamma) \wedge \neg eq(\alpha,\beta) \wedge \alpha = a. \tag{3.5}$$

A vocabulary of 18 characters is used, out of which 12 are used for training and 6 are used for validation/testing in addition to ‘a’ and ‘b’, which need to appear in all sets because of the definition of the concrete patterns.

The classes are mixed with abstract and concrete patterns. There are four classes (2 abstract and 2 concrete patterns). Each class is a combination of one abstract and one concrete pattern. Each of the 4 classes are defined as follows. For classes 1 and 3 abstract pattern ABA is used and for classes 2 and 4, ABB abstract pattern is used. Similarly, classes 1 and 2 are defined to start with token ‘a’ and classes 3 and 4 are defined to start with token ‘b’ respectively. This is ensured so that there is a mix of each of the classes when performing a four-way classification. The train, validation and test split is 50%, 25%, and 25% respectively.

The network was trained for 10 epochs, leading to perfect classification on the training set. A total of 10 simulations has been performed. A feed forward and a recurrent neural network were tested both in RBP Early and Mid Fusion.

The results are shown in Table 3.16.

RBP	FFNN	RNN
Without RBP	23%	42%
RBP Early Fusion	49%	57%
RBP Mid Fusion	100%	100%

Table 3.16 Test set accuracy for mixed abstract/concrete pattern classification.

As in the previous experiments, networks without RBP fail to generalise the abstract patterns. The results for RBP Early Fusion and Mid Fusion show, that the ability to learn and recognise the concrete patterns is not impeded by adding the RBP structures.

3.8 Observations and Remarks

3.8.1 Standard neural networks

The results of the experiments described above confirm the results of Marcus et al. (1999) and others that standard recurrent (and feed-forward) neural networks do not learn generalisable identity rules. From the tested models and settings of the task it can be seen that the lack of activation of input neurons impedes learning, but avoiding this is not sufficient. The task assumes that the identity of input tokens is easy to recognise, classify and base predictions on, but the models that are tested do not learn to generalise in this way. These results confirm the view that in order to generalise it is necessary to know which input neurons are related, similarly on the next level, which comparisons of input belong to a pair of tokens so that they can be aggregated per token. The structure of neural networks does not provide any prior preference for inputs that are related in this way over any other combinations of inputs. This makes it seem plausible that the solutions by Altmann and Dienes (1999); Elman (1999); Seidenberg and Elman (1999) could not be replicated by Vilcu and Hadley (2001, 2005).

3.8.2 Constructive model with RBP

The RBP model addresses the learning of identity rules by adding neurons and connections with fixed weights. From the input neurons connections are added to a DR (differentiator-rectifier) unit from each pair of corresponding input neurons within any pair of tokens (represented in one-hot encoding). These DR units calculate the absolute of the difference of the activations of the two input neurons. They are followed by DR units that aggregate by taking the sum of the DR unit activations for each pair of tokens. The fact that the DR units relate to the difference between each pair of neurons makes the learning task for classification much simpler, as has been confirmed by our results. An open question in this context is why the RBP Mid Fusion is so much more effective than RBP Early Fusion for classification, although the only difference is the layer in which the information is added into the network.

For prediction, a more complex structure is needed, as beyond recognition of identity, also the selection of the token to predict is required, that depends on the tokens in the context and their similarity relations. The constructive RBP solution requires a transformation into a representation of identity relations in the input that is mapped to identities between input and output and that is mapped back to the token space by adding prediction probability to the tokens that are predicted to be identical between input and output. This created a complex predefined structure, but without it even the models that achieved prefect classification failed to make correct predictions with new data. Only the LSTM models could use the RBP1 and RBP2 information to make prediction above the baseline (22% vs 8.3%). It is therefore hypothesises that the gating structure of the LSTMs enables at least some effective mapping. The 100% correct predictions by all models using RBP Late Fusion shows the effectiveness of this structure.

Adding a bias into the network with a predefined structure such as RBP raises the question whether there is a negative effect on other learning abilities of the network and whether interactions between the abstract and concrete tasks can be learnt. In the mixed pattern experiment, RBP is still effective and showed no negative effect. In experiments with prediction it was observed that that RBP Late Fusion has a positive effect overall.

3.8.3 Extrapolation and inductive bias

The results in this study confirm that an inductive bias is needed for extrapolation, in the terminology of Marcus (2018), in order to generalise in some sense outside the space covered by the training data. This general challenge has recently attracted some attention. E.g., Mitchell et al. (2018) provided several solutions to the related problem of learning equality of numbers (in binary representation), which does not generalise from even to odd numbers as pointed out already by Marcus (2001). As the authors point out in Mitchell et al. (2018), an essential question is which biases are relevant to the domain and problem. The identity problem addressed here is in itself fundamental to learning about relations Battaglia et al. (2018), as relations depend on object identity. This further raises the question what is needed to enable more complex concepts and rules to be learnt, such as more general logical concepts and rules.

The identity rules also point to the lower-level problem that the natural relations of position and belonging to objects are not naturally addressed in neural networks. Other tasks may require different structures, relating for example to arithmetics, geometry or physics (Cohen and Shashua, 2016b). It is therefore seen as an important task the definition or predefined structures in neural networks, so that they create useful inductive bias, but do not prevent learning of functions that do not conform to that bias.

To summarize, the question why standard neural networks do not learn abstract patterns and vector equality relations in a generalisable way is studied in this chapter and a solution called 'Relation Based Patterns' (RBP) has been proposed. RBP can be integrated in standard neural networks in three ways, *Early Fusion*, *Mid Fusion* and *Late Fusion*.

A simple modification to the network with differential rectifier (DR) units resulted in substantial improvements on unseen test data. This improvement is largely independent of vector dimension, data size and other parameters. Similar results are observed in other numerical tasks like numeric inequality and sum of bits of binary vectors, digit reversal and parity.

RBP has shown positive effect so far on synthetic and artificially simulated tasks leading to significant improvements. This provides the evidence that relatively simpler tasks like equality learning have a positive effect after introducing inductive biases like RBP into standard neural networks.

It is therefore important to investigate the design of further measures for creating and controlling inductive biases in neural network learning for various real-world tasks. In the next chapter, the effect of RBP is tested on the real-world application of music modeling as melodic sequences in music have repetition cues that can be possibly modeled with the RBP structures.

Chapter 4

Case study: Applying RBP to Melody Modeling

4.1 RBP for Pitch Prediction

Repetition is generally seen as an essential feature of music, but not often used as a feature to model and characterize music. Computational modeling of music involves studying abstract patterns like repetition of notes. A useful way of testing the relevance of a melodic feature or model is to test its capacity to support prediction.

In this chapter, abstract repetition patterns of pitches are studied in the context of monophonic folk melodies as a case study using various models ranging from statistical to neural network models. Specifically in statistical melody models, the often used Markov (or n-gram) models lack a mechanism for representing repetitions as such. The recently popular recurrent neural networks as Turing-complete models are able to represent them given enough resources, but typically fail to detect repetition patterns when trained with gradient descent. In this chapter, the use of repetition patterns as features in music analysis and prediction are explored.

Several experiments are performed using the recurrent neural network models and their performance is compared by using the new RBP approach described in the previous chapter. The details about RBP approach as adapted to music modeling, the methodology and the dataset used for the analysis are described in the rest of the chapter.

This RBP music language model is inspired by the time delay neural networks (Waibel et al., 1989), where the input context is the current note along with the $t - 1$ previous notes. Each note is represented as a one-hot-encoded vector of its pitch. The output is a softmax over the possible pitches for the next note. The basic model is extended with the RBP as shown below in the next Section. The approaches are adapted from the RBP model described already in Chapter 3, however the only difference is in terms of the input given to the neural network, in this case being a set of pitches in a melodic context. To give a better understanding of how RBP can be used for notes in a melody context instead of set of abstract tokens as shown in Chapter 3, the formulation of RBP is explained here again.

The rest of the Chapter focuses on describing the RBP method again for music modeling, showing importance of similarity correlation in the context of melodies, experiments with RBP using various fusion models on Markov models, RNNs and their gated variants followed by performance analysis and plots.

4.2 Repetition Detection

There are two stages in our analysis. Firstly a repetition detector called a DR unit is defined, which identifies notes with the same pitch. This is applied to all pairs of notes in a given note sequence of length t, our *input context*, and to all notes in the input context in comparison to the following note, the *output*, or prediction.

The repetition pattern on the context is then used to predict the repetitions in the output, which are combined with a general music language model. Alternatively, the repetition patterns are used in the input context directly as an additional input for a music language model. These are the early, mid and late fusion approaches as defined in Chapter 3, respectively, which are described again in the sections below.

4.2.1 Repetition Pattern Representation

The first step in the analysis is to construct a mechanism by which the repetitions patterns in an input context of length t can be represented as a vector. To do this, a differential rectifier unit is implemented. This is related to the standard ReLU, but applies a full wave rectification to the difference between two inputs. The standard ReLU function for input x is defined as

$$f(x) = max(0, x),$$

while here the differential rectifier unit (DR units) is defined as

$$f(x, y) = |x - y|,$$

where x and y are the input values, in this case the pitches of two notes. These units will output 0 if two pitches are the same, and a positive value (the pitch interval in semitones) otherwise.

Based on the context length t, $k = t(t-1)/2$ *DR input units* are created for all the possible pairs of notes in the input context which are called as $(DRin_1, ..., DRin_k)$ as shown in Figure 4.1, which defines the input repetition vector.

For the next note, *DR output units* $(DRout_1, ..., DRout_t)$ are defined as the *output repetition vector*. Figure 4.2, defines the output repetition vector. The DR_{out} takes the pitch of each of the t notes in the input context in combination with the output note's pitch as input for the DR units.

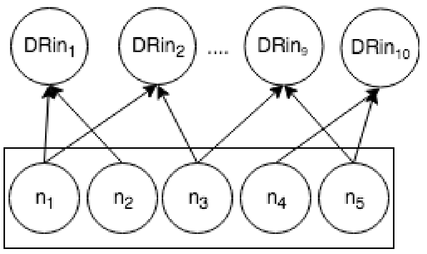


Fig. 4.1 DR input units for time window t=5.

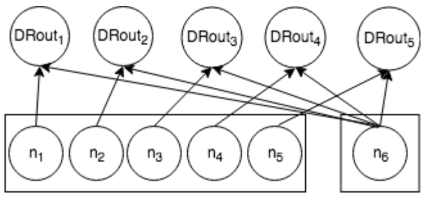


Fig. 4.2 DR output units for time window t=5.

4.2.2 Calculating DR input and output units

The DR input units can be used as additional inputs to the music language model. However, here the key hypothesis is that the repetitions in the input pattern have an influence on the output pattern. Therefore, for a full implementation of the RBP, DR output vectors needs to be predicted based on the DR input vectors. A feed-forward neural network with logistic sigmoid as the hidden layer activation function is used for this, which is trained on the DR input and DR output patterns computed from the data to arrive at the distribution of the DR output units given the DR input units.

In order to understand the correlations between the input and output patterns, several tests are performed on the patterns across different genres. The analysis of the correlation of repetition patterns is done. The dataset used for the experiments is summarized below, followed by the results of correlation between input and output patterns.

4.3 Corpus

The corpus used for the experiments is Essen Folk Song Collection (Schaffrath, 1995a). The corpus consists of 8 datasets of monophonic folk chorales and melodies for different genres. This has been previously used by (Pearce and Wiggins, 2004), (Cherla et al., 2013), (Cherla et al., 2015) for melody prediction tasks. The corpus contains 119 Yugoslavian folk melodies, 91 Alsatian folk melodies, 93 Swiss folk melodies, 104 Austrian folk melodies, 213 German folk melodies, 185 Bach chorales, 152 Canadian folk melodies and 237 Chinese folk melodies. The number of predictable musical events in these melodies ranges between 2,691 for the smallest one (Yugoslavian folk melodies) and 11,056 for the largest (Chinese folk melodies).

Examples of repetition cues of the melodies taken from the Essen Folk Song Collection are shown in Figures 4.3 and 4.4.

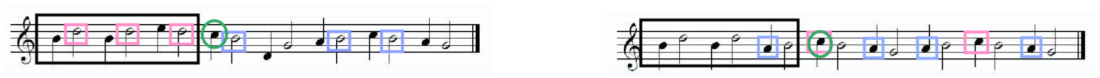


Fig. 4.3 Examples from the Essen Folk Song Collection : Repetition of notes.

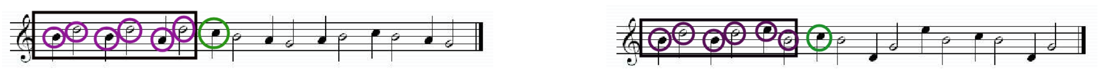


Fig. 4.4 Examples from the Essen Folk Song Collection : Context length *n=6* (pink) and corresponding note (green).

Table 4.1 provides an overview of the 8 datasets in the corpus. The corpus consists of monophonic melodies in MIDI format. Each of the MIDI files in the 8 datasets are encoded to the **kern format. Music21 library (Cuthbert and Ariza, 2010a) is used to parse and extract pitch information. The pitches are one-hot and integer encoded in several experiments performed, the details of which are given in the subsequent sections.

Genre	Songs	Events	Pitches
Elsass	152	8553	25
Bach	185	9227	21
Schweiz	91	4496	32
Jugoslav	119	2691	25
Swiss	93	4586	34
Austrian	104	5306	35
German	213	8393	27
Chinese	237	11056	41

Table 4.1 Essen Folk Song Collection (Schaffrath, 1995a).

A 10 fold cross validation is performed for each of the songs in the 8 datasets as done in (Pearce and Wiggins, 2004), (Cherla et al., 2013) as the dataset is small and also to make our results comparable with the previous models.

4.3.1 Correlation between input and output patterns

To calculate the correlation between input and output patterns DR units are used as described above. Given a context size n, the mapping of the DR output patterns given the DR input patterns is calculated. DR units are always calculated for a pair of notes. For the ease of analysis, binary DR values are considered. A value of 0 corresponds to pair of notes being equal and 1 corresponds to pair of notes being unequal.

For example, consider a context size of length 5. In this case, the DR input patterns are calculated as outlined in Section 3.1 which results in a total of 10 DR inputs. Here as n is 5, the output is the sixth note. To get the output DR units, a simple neural network is trained using scikit-learn [1] MLPRegressor (Hinton, 1989) with logistic sigmoid as the activation function, to arrive at the distribution of the output DR units given the input DR units.

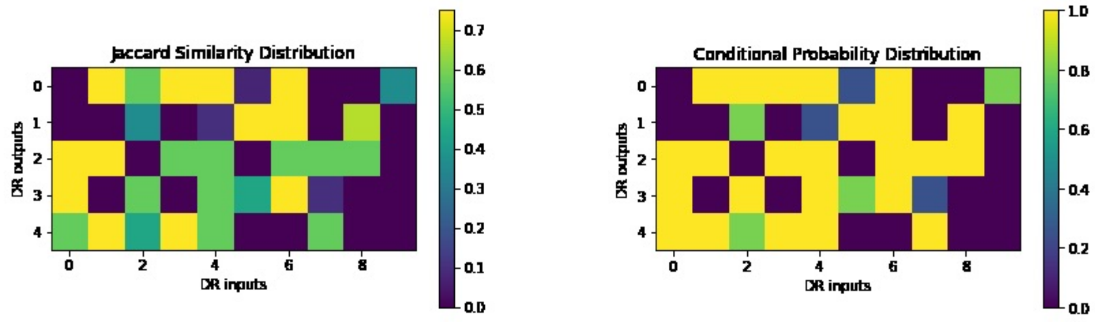
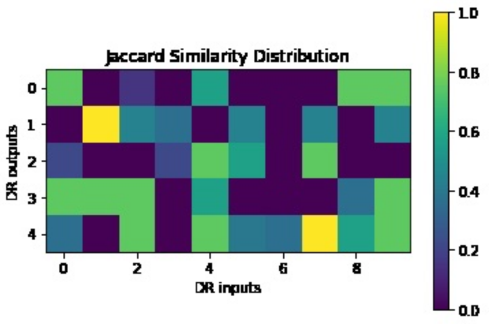


Fig. 4.5 Jaccard similarity and Conditional probability distribution of 10 input DR units and 5 output DR units for Schweiz dataset

Figures 4.5 and 4.6, describe the similarity matrix of the 10 DR input units and 5 output DR units for Schweiz (Swiss) dataset and Shanxi (Chinese) dataset, both taken from the Essen Folk Song collection (Schaffrath, 1995a). In the Essen Folk Song Collection, there are a total of 8 datasets, as described in the above section. Here for showing the similarity correlation, Swiss and Chinese datasets are considered as they seem to have more variation and abstract repetition patterns when compared to other datasets.

[1] http://scikit-learn.org



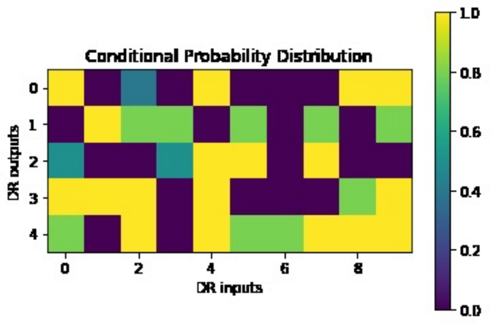


Fig. 4.6 Jaccard similarity and Conditional probability distribution of 10 input DR units and 5 output DR units for Shanxi dataset.

Continuing from the example above, if the context size of the sequence is 5, then for every note there are 10 input DR units and 5 output DR units. Each input DR is compared with each of the output DR to get a 10x5 matrix where each of the elements represents the similarity index over all 4581 notes for the Schweiz (Swiss) dataset as an example. The similarity correlation between the inputs in a given context and the corresponding output is calculated.

To compute the similarity, Jaccard index [2] and conditional probability distribution are used. [3] The above methods are used for visualising the correlation of input and output DR units in a given melodic context. Jaccard similarity is defined as follows :

$$J(X,Y) = |X \cap Y| / |X \cup Y|$$

where X corresponds to the set of notes in the input context being equal and Y corresponds to the set of notes in the input context and the output being equal. Each of the above events constitute a set, so here Jaccard similarity (J) is looking at a certain equality relationship between the input context and the input context and the output being the same, so we are comparing intersection of the sets with the union of the sets.

Similarly, conditional probability is also calculated as follows :

[2]http://www.statisticshowto.com/jaccard-index/
[3]http://setosa.io/conditional/

$$P(X/Y) = P(X \cap Y)/P(Y)$$

where P denotes the probability that X i.e. notes in the given input context are equal and Y i.e. notes between the input context and the output are equal given the condition that Y hold true. This is another way of measuring how DR output and input units are related.

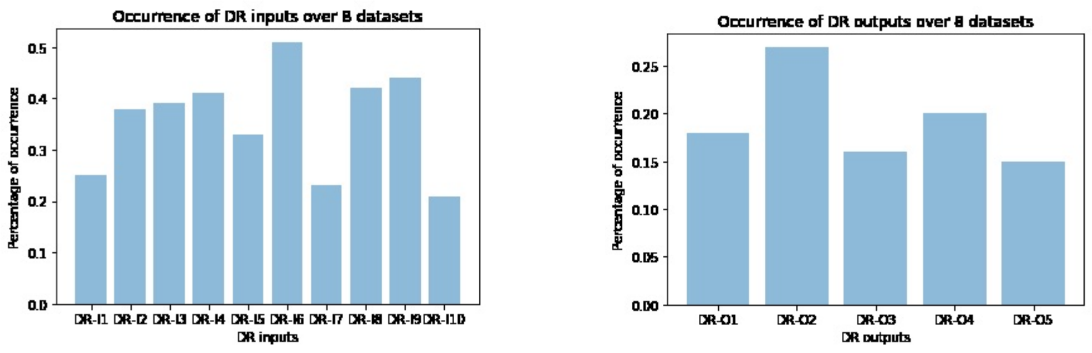


Fig. 4.7 Overall distribution of DR input and output units for 8 datasets.

Similarly, the distributions of all the other datasets are calculated. The overall proportion of repetitive input and output DR patterns for a context length of 5 for all the 8 different datasets is given in Figure 4.7. This shows that repetition is indeed an essential feature of music and it is important to take the information of the repetition patterns while performing music analysis. Once the DR input and output units are computed, they are used in a Markovian setting to estimate the pitch prediction first, followed by using three variants of recurrent neural network models. The process is outlined in the next section.

4.3.2 Prediction Task

Markovian modeling

The outline for the prediction task for Markov models is as follows. This is similar to RBP in Late Fusion.

- The inputs are integer encoded pitch values (midi numbers).

- The prediction task in a zero order, first order and second order Markov model (pitch space) setting is performed for all the 8 datasets individually and the average cross entropy loss of the outputs is calculated.
- Now DR units for each pair of input notes is calculated.
- Given each DR input units, the corresponding correlation of the DR output units is calculated using MLPRegressor as already outlined above.
- These output DR units are mapped back into the pitch space using mixture of experts approach (Jacobs et al., 1991) as a post processing step after computing the pitch prediction probability distribution.
- The overall output distribution is normalised and the cross entropy loss after adding the DR outputs is calculated.
- The average cross entropy loss with and without adding DR units is now compared for all the 8 datasets. The results are tabulated in Tables 4.2, 4.3, 4.4.

Neural Networks

RBP integration in neural network models(RNN, GRU and LSTM) are carried out in three ways described as below. The approaches are similar to those described in Chapter 3 and briefly summarized again here.

- Early Fusion : DR units are added as extra inputs to the network.
- Mid Fusion : DR units are added to the hidden units of the network.
- Late Fusion : In this approach, the whole RBP training and prediction process are performed separately from the neural language model. The predictions of the RBP, which are repetition patterns, are then projected back into pitch space, and combined with the language model in final network layer.

  The exact procedure is outlined below. It comprises of two phases.

– The first step is to train the neural language model as normal without adding any DR units or RBP.

– In the second phase, the RBP is trained to predict DR output units from DR input units.

– In the third phase, the Neural Language Model and the RBP are combined. For this, the DR outputs are mapped back to pitch space. The DR output values are normalised to zero mean as

$$DRoutn_k = DRout_k - avg_l(DRout_l).$$

– This is done, because the DR output values are not a probability distribution but rather a prediction of whether the next note will be a repetition of any n_k.

– Then it is mapped back to the output o_{RBP} per pitch $(p_1, ..., p_m)$ as follows:

$$o_{RBP}(p_l) = \begin{cases} \sum_k DRoutn_k \quad \text{if} \quad pitch(n_k) = p_l \\ 0 \ \text{ if } \nexists k \text{ such that } pitch(n_k) = p_l \end{cases}$$

– This output is treated as an offset that increases or decreases the probability of a pitch in the next note.

– Then the output vectors of the Neural Network and o_{RBP} are combined with separate weights and a bias per combined output.

– In this configuration the whole model is trained end to end.

For performing the analysis, monophonic folk melodies and chorales are used. The detailed description of the dataset is given in the next section.

4.4 Experimental Results

4.4.1 Model Parameters

Two different set of experiments are performed here. One of them is on evaluating the performance using the zero order, first order and second order Markov models. Even though Markov models cannot learn abstract repetition patterns, we perform the experiments on them to notice the effect of RBP integration. Then, the three neural network models, i.e. RNN, LSTM and GRU, are used with different input context length [3,4,5,6,7,8,9].

The choice of context length was fixed as the above because of the musical intuition that repetition cues are more effective in shorter segments, and in order to limit the complexity, as the number of DR input units grows quadratically with the size of the input context.

The number of hidden units and epochs are set to 20 and 30 respectively after performing a grid search over [10,20,30,50] for hidden units and no of epochs. A single hidden layer is used in all the three neural network models and the learning rate is set to 0.01.

To evaluate the model, cross entropy is used to compare the effect of DR units on melody prediction. An estimate of cross entropy between two probability distributions original distribution p and predicted distribution q is given by

$$[\ H(p,q) = -\Sigma_x p(x) \log q(x)] \tag{4.1}$$

where x is the possible set of events. Lower cross-entropy values indicate better prediction.

4.4.2 Markov models

Tables 4.2, 4.3, 4.4, show the cross entropy (rounded to four decimal points) of all the 8 datasets without DR units and with DR units for Zero order, First order and Second

Genre	Without DR units	With DR units
Nova scotia	3.2312	3.0682
Elsass	3.5814	3.3176
Jugoslav	3.5908	3.4211
Bach	3.3786	3.1502
Kinder	3.4862	3.2312
Shanxi	3.8929	3.7693
Schweiz	3.8265	3.6896
Osterrh	3.8704	3.5989
Average	**3.6072**	**3.4057**

Table 4.2 Cross entropy loss with and without DR units across 8 datasets for Zero order Markov model.

order Markov models respectively. It is seen that there is a consistent decrease in the overall cross entropy for all the 8 genres for all the Markov models. This suggests that Markov models lack a mechanism to understand abstract patterns and providing this abstract pattern distribution to the model results in a considerable change in the predictions for all the 8 datasets.

Genre	Without DR units	With DR units
Nova scotia	2.8508	2.6430
Elsass	2.9670	2.7576
Jugoslav	2.9890	2.7668
Bach	2.8623	2.6416
Kinder	3.0142	2.7905
Shanxi	3.0768	2.7072
Schweiz	3.0350	2.8285
Osterrh	2.9609	2.7442
Average	**2.9695**	**2.7349**

Table 4.3 Cross entropy loss with and without DR units across 8 datasets for First order Markov model.

Genre	Without DR units	With DR units
Nova scotia	2.8086	2.6116
Elsass	2.9125	2.7875
Jugoslav	2.9176	2.4002
Bach	2.8176	2.6291
Kinder	2.8353	2.3256
Shanxi	3.0477	2.9537
Schweiz	2.9215	2.7456
Osterrh	2.9094	2.8947
Average	**2.8697**	**2.6685**

Table 4.4 Cross entropy loss with and without DR units across 8 datasets for Second order Markov model.

4.4.3 Neural Network Models

The performance of the three neural network models are evaluated by varying in the following conditions:

- Neural network without RBP
- RBP Early Fusion
- RBP Mid Fusion
- RBP Late Fusion

To evaluate the models, cross entropy loss is used as the metric for prediction performance. The experimental configurations are listed in Table 4.5.

The performance of the models are evaluated using different forms of encoding ie. both integer and one-hot encoding and then further experiments are performed using different context lengths as explained below.

4.4.4 Integer Encoding vs One Hot Encoding

The input encoding in the neural network models has been tested using both one hot encoding (Figure 4.8) and integer encoding (Figure 4.9).

No	Experiment
Exp 1	RNN without RBP
Exp 2	RNN + RBP (Early, mid and Late Fusion)
Exp 3	GRU without RBP
Exp 4	GRU + RBP (Early, mid and Late Fusion)
Exp 5	LSTM without RBP
Exp 6	LSTM + RBP (Early, mid and Late Fusion)

Table 4.5 Overview of the experiments performed on 8 datasets using different context lengths.

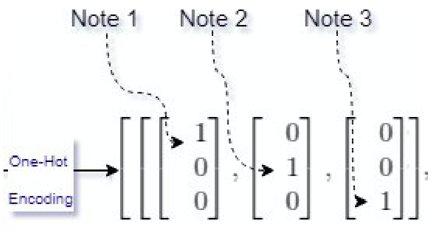


Fig. 4.8 Example of one hot encoding with Note 1 (A2), Note 2 (B2), Note 3 (C2).

It is observed that the integer encoding performed better than one hot encoding. For example, for a context size of length 5, the performance of RNN, GRU and LSTM models for three different genres Bach, Elsass and Shanxi from the Essen Folk Song data are tabulated below in Tables 4.6, 4.7 and 4.8 respectively. The reason behind choosing three different genres is that all the three genres are very different from each other, so it would be interesting to note how encoding them in various forms influences the prediction at the output.

Type	Songs	Integer encoding	One hot encoding
Bach	185	2.4336	2.8548
Elsass	152	2.7715	2.8322
Shanxi	237	2.9913	2.9981

Table 4.6 Performance of Integer vs One hot encoding using the RNN model for Bach, Elsass and Shanxi datasets.

The accuracy plots of the integer encoding and one-hot encoding for sequence length 9 are given in Figures 4.10 and 4.11 respectively.

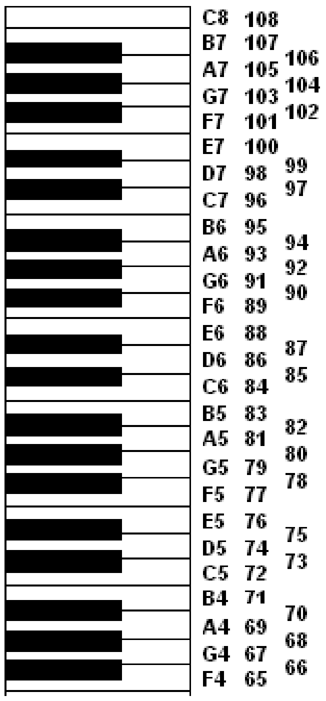


Fig. 4.9 MIDI values of notes - Integer encoding.

Type	Songs	Integer encoding	One hot encoding
Bach	185	2.4334	2.8579
Elsass	152	2.7721	2.8322
Shanxi	237	2.9925	2.9948

Table 4.7 Performance of Integer vs One hot encoding using the GRU model for Bach, Elsass and Shanxi datasets.

Integer encoding was significantly better than one-hot encoding in all cases. Integer encoding with LSTM in late fusion resulted in the best performance overall. As integer encoding performed better than one hot encoding, for the next set of experiments using different context lengths only integer encoding is used.

4.4.5 Experiments over different context lengths

Tables 4.9, 4.10 and 4.11 provide the values of the average cross entropy loss across different context lengths and the corresponding accuracy of the models are given in Figures 4.12, 4.13 and 4.14 respectively.

Type	Songs	Integer encoding	One hot encoding
Bach	185	2.4348	2.8540
Elsass	152	2.7635	2.8320
Shanxi	237	2.9901	2.9967

Table 4.8 Performance of Integer vs One hot encoding using the LSTM model for Bach, Elsass and Shanxi datasets.

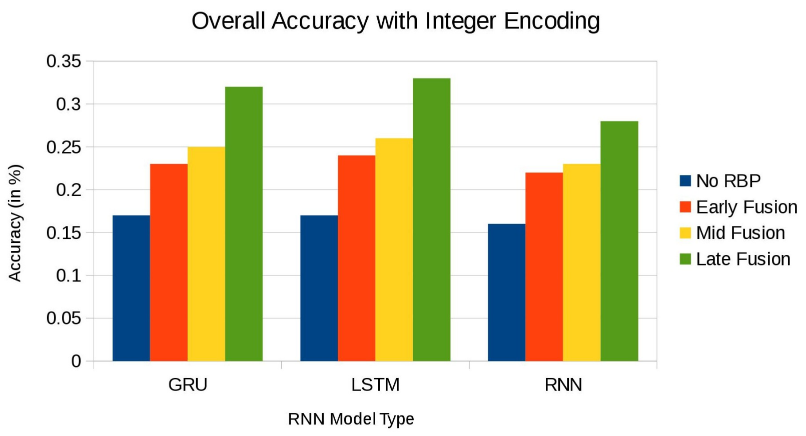


Fig. 4.10 Overall Accuracy with Integer Encoding for sequence length : 9 without RBP and RBP in Early, Mid and Late Fusion.

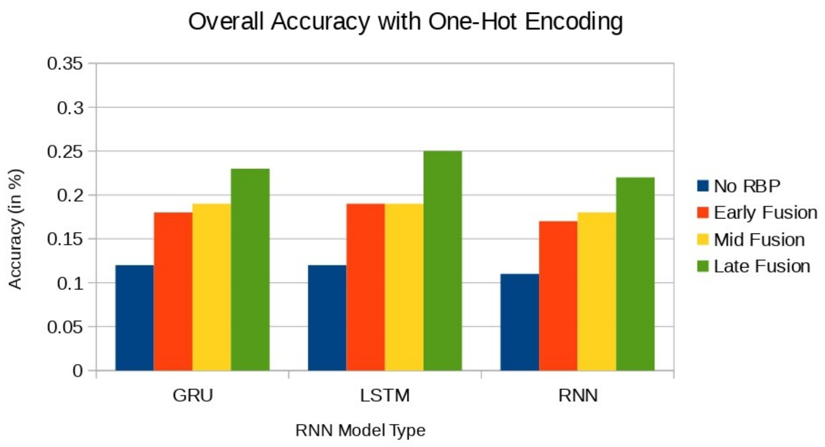


Fig. 4.11 Overall Accuracy with One-Hot Encoding for sequence length : 9 without RBP and RBP in Early, Mid and Late Fusion.

Seq	Without	With RBP		
Len	RBP	Early Fus	Mid Fus	Late Fus
n= 2	2.8467	2.7743	2.7572	2.6232
n= 3	2.8323	2.7682	2.7585	2.6254
n= 4	2.8166	2.7516	2.7264	2.6232
n= 5	2.8012	2.7365	2.7356	2.6065
n= 6	2.8023	2.7387	2.7243	2.5878
n= 7	2.7704	2.6803	2.6632	2.5957
n= 8	2.7856	2.6823	2.6745	2.5868
n= 9	2.7503	2.6702	2.6535	2.5927

Table 4.9 Average Cross Entropy Loss with and without RBP units across 8 datasets using RNNs for various context lengths.

Seq	Without	With RBP		
Len	RBP	Early Fus	Mid Fus	Late Fus
n= 2	2.8459	2.7703	2.7568	2.6172
n= 3	2.8338	2.7623	2.7565	2.6208
n= 4	2.8179	2.7498	2.7238	2.6224
n= 5	2.7968	2.7328	2.7302	2.6012
n= 6	2.7995	2.7312	2.7213	2.5856
n= 7	2.7687	2.6756	2.6614	2.5945
n= 8	2.7823	2.6745	2.6704	2.5867
n= 9	2.7456	2.6678	2.6528	2.5907

Table 4.10 Average Cross Entropy Loss with and without RBP units across 8 datasets using GRUs for various context lengths.

4.4.6 Genre wise analysis

Along with encoding and context length, it is also interesting to evaluate the affect of RBP on different genres in the Essen Folk Song Collection. The performance of different datasets using an RNN without RBP and with RBP in Early Fusion, Mid Fusion and Late Fusion are given in Figure 4.15.

Seq	Without	With RBP		
Len	RBP	Early Fus	Mid Fus	Late Fus
n= 2	2.8339	2.7588	2.7552	2.6123
n= 3	2.8268	2.7610	2.7512	2.6157
n= 4	2.8080	2.7456	2.7214	2.6165
n= 5	2.7928	2.7314	2.7219	2.5962
n= 6	2.7875	2.7266	2.7175	2.5831
n= 7	2.7646	2.6711	2.6554	2.5933
n= 8	2.7811	2.6709	2.6679	2.5832
n= 9	2.7428	2.6618	2.6517	2.5867

Table 4.11 Average Cross Entropy Loss with and without RBP units across 8 datasets using LSTMs for various context lengths.

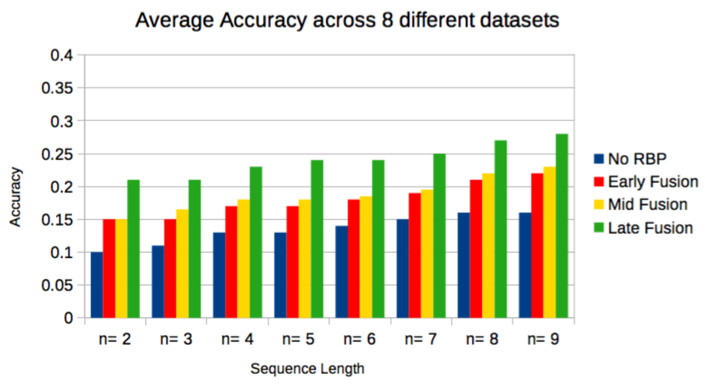


Fig. 4.12 Overall Accuracy with and without RBP across 8 datasets using RNNs for various context lengths n with integer encoding.

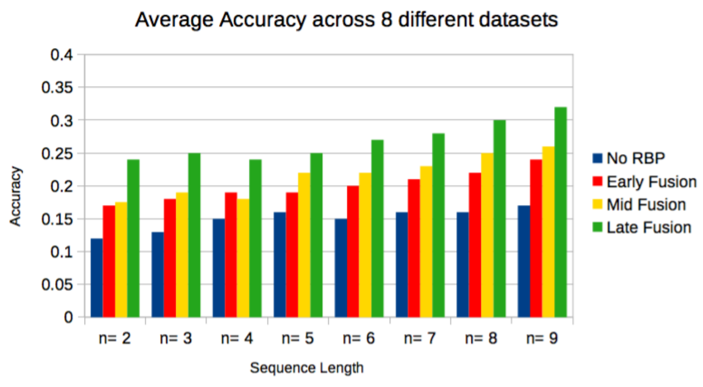


Fig. 4.13 Overall Accuracy with and without RBP across 8 datasets using GRUs for various context lengths n with integer encoding.

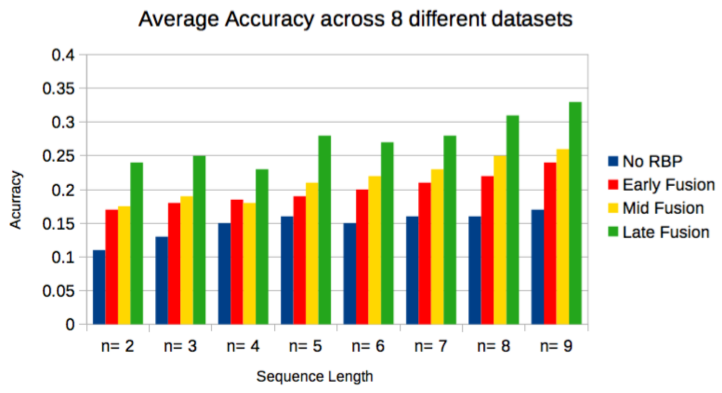


Fig. 4.14 Overall Accuracy with and without RBP across 8 datasets using LSTMs for various context lengths n with integer encoding.

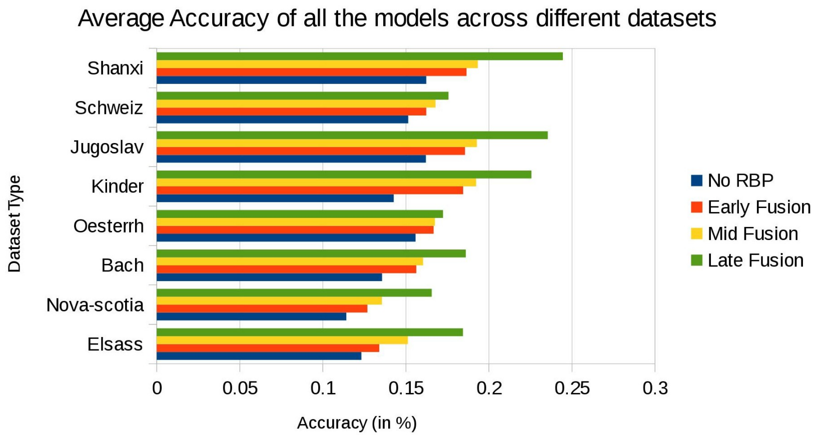


Fig. 4.15 Overall Accuracy across 8 different datasets of the Essen Folk Song Collection using an RNN model.

4.4.7 Extending RBP in Late Fusion with octave intervals using D units

Similar to the DR units, another variant of RBP in late fusion is tried now with D units ie. Differential Units. D units are defined as

$$f(x, y) = x - y,$$

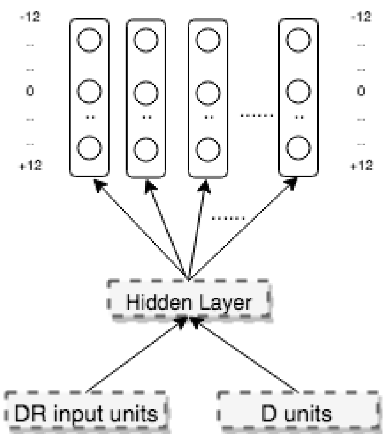


Fig. 4.16 Combining DR units and D units in RBP Late Fusion.

where x and y are the input values, in this case the pitches of two notes. These units will output 0 if two pitches are the same, and a signed value of the the pitch interval in semitones otherwise.

Both the DR units and D units are added to the RBP in Late Fusion as shown in Figure 4.16. This octave interval offset is now mapped back similar to the standard RBP in Late Fusion. The overall cross entropy loss and accuracy after combining DR units and D units are given in Table 4.12 and Figure 4.17 respectively. The results show consistent improvement after adding the D units as it gives rise to a much richer representation in terms of learning exact octave interval offsets.

Seq	Late Fusion	
Len	DR units	DR and D units
n= 2	2.6232	2.5984
n= 3	2.6254	2.5925
n= 4	2.6232	2.5864
n= 5	2.6065	2.5802
n= 6	2.5878	2.5724
n= 7	2.5957	2.5714
n= 8	2.5868	2.5654
n= 9	2.5927	2.5658

Table 4.12 Average Cross entropy loss with DR units and DR and D units for RBP in Late Fusion across 8 datasets using LSTMs for various context lengths.

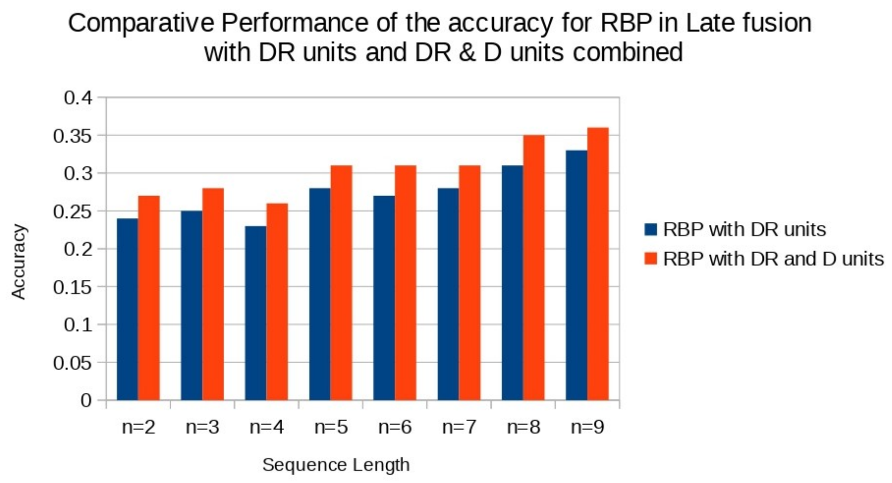


Fig. 4.17 Overall Accuracy for RBP in Late Fusion with DR units and combining DR and D units.

4.5 Observations and Remarks

Starting with Markov models, it is seen that there is a consistent decrease in the overall cross entropy for all the 8 genres in the Essen Folks Song Collection as observed by the experiments using Zero, First and Second order Markov models after adding the DR units. This suggests that Markov models lack a mechanism to understand abstract patterns and providing this abstract pattern distribution to the model results in a considerable change in the predictions for all the 8 datasets.

For evaluation cross entropy has been used in all the experiments. Lower cross-entropy values indicate better prediction. There is also a consistent decrease in the overall cross entropy for all the 8 genres in the case of all the neural network models as well after adding the DR units in early, mid and late fusion. Across the three models, the late fusion approach has best average results overall when compared to the early and mid fusion approach or the models without RBP.

Out of the three neural network models, LSTMs performed the best, followed by GRUs and then RNNs. Overall, LSTMs with late fusion produced the best result among all models tested here and also improves over the best reported performance in pitch prediction across various context lengths. In terms of statistical significance, each of the RBP models were better when compared on LSTMs, GRUs and RNNs. Although the difference in terms of cross entropy loss reported in the results seems small, but the clear difference in accuracy shows that modeling repetition based patterns with RBP does benefit overall prediction. This does seem to have much practical significance in general for building better melody models, where RBP can be applied to capture abstract repetition patterns between notes in a given melodic context and add the patterns into standard neural networks. The fact that all neural network models performed better when combined with RBP indicates that these models are not making full use of the identity relations represented in our repetition pattern vectors. The LSTM models in late fusion with integer encoding led to the best performance overall. The performance did benefit from the RBP for all the network models and this concurs with the claims by Marcus (2018) that neural networks in their current form fail to effectively learn abstractions from data, especially small amounts of data. Therefore approaches like RBP are necessary to model melody sequences using neural networks and also can be applied to other sequence to sequence learning tasks.

In the next chapter, the approach of RBP is extended to ERBP which is the Embedded Relation Based Pattern where further improvements (state of the art in many) are observed in the performance of all the neural network models on various tasks. The ERBP approach and the experimental details are described in detail in the next chapter.

Chapter 5

Embedded Relation Based Patterns Approach

5.1 Bayesian Approach to Relation Based Patterns

Relation Based Pattern (RBP) adds units with hard-wired connections and non-standard activation function, which limits the flexibility of that part of the network and makes practical use more complex. By contrast, Embedded Relation Based Pattern (ERBP) introduces a modified Mid Fusion RBP structure (shown in Figure 5.1 and remodels it as a Bayesian prior on a standard weight matrix, thus avoiding hard-coded weights and non-standard network structures and activation functions.

DR units are modeled with two standard neurons and model the fixed weights with a default weight matrix D, that is applied at initialisation and through a loss function to the weights between input and hidden layer with ReLU activation. A diagram of the ERBP structure is given in Figure 5.2 where α and β are the two input values being compared. An example of this default matrix for context length $n = 3$ is shown in Figure 5.3. The incoming connections from two corresponding inputs to a neuron in the hidden layer to be compared have values of $+1$ and -1.

For the same pair another hidden neuron is used with inverted signs of the weights, as in rows 1 and 2. Therefore at least $2n$ hidden units are required in the first hidden

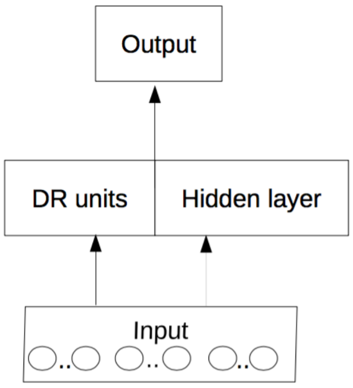


Fig. 5.1 Structure of the RBP Mid Fusion. The DR units are concatenated with the first hidden layer. The weights of incoming connections of the DR units are fixed in RBP. In ERBP, the same structure is used but with a trainable weights. Then the only difference to a standard Feed Forward Neural Network is the prior on the weights between input and hidden units as described in the text. This structure can be applied to Recurrent Neural Networks, as long as there are n items represented in the input.

layer for each comparison of two n-dimensional vectors. All other incoming weights to the hidden layer are set to 0, including the bias. This ensures that in all cases where corresponding inputs are not equal, there will be a positive activation in one of the hidden neurons.

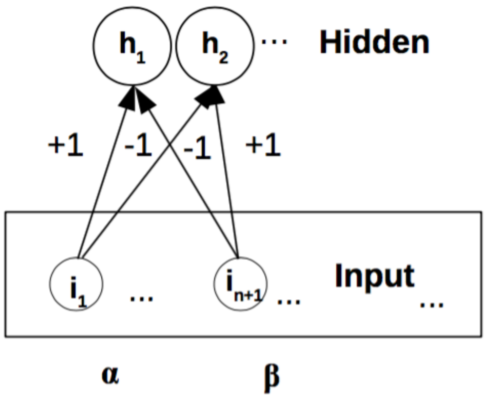


Fig. 5.2 ERBP: The DR units are each modelled by two hidden layer neurons h_1, h_2. α and β indicate two input vectors of dimensionality n.

As stated, Matrix D is used for initialisation and in the loss function. In the initialisation, weight matrix W between the input and hidden layer is first initialised as normal, e.g. with low-energy noise, and then the non-zero values of D are copied to

$$D = \begin{pmatrix} +1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & +1 & 0 & 0 \\ 0 & +1 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & +1 & 0 \\ 0 & 0 & +1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & +1 \end{pmatrix}$$

Fig. 5.3 Default weight matrix D for two input vectors of dimension n=3. Each row corresponds to the incoming weights of a hidden neuron. If there are more hidden neurons than pairs of input neurons, the additional rows contain only zeros.

W. If n_{inp} is the number of input units then D matrix with i rows and j columns is initialised and updated as follows :

Algorithm 1 ERBP weight matrix initialisation

```
$j \leftarrow 0$
**for** $i$ in range($n_{inp}/2$) **do**
    $D_{j,i} = +1$
    $D_{j,i+n_{inp}/2} = -1$
    $j+ = 1$
    $D_{ji} = -1$
    $D_{j,i+n_{inp}/2} = +1$
    $j+ = 1$
    **if** $j > n_{inp}$ **then**
        break
    **end if**
**end for**
```

Further, ERBP loss is defined based on the difference between D and the actual weights in W. Both $L2$ or $L1$ norm of the difference $D - W$ is computed. This loss term corresponds to a Bayesian prior on the weights with the mean defined by the values of D. The $L2$ loss corresponds to a Gaussian prior and $L1$ loss to a Laplacian prior, such that backpropagation maximises the posterior likelihood of the weights given the data (Williams, 1995). The overall training loss l_t is defined as

$$l_t = l_c + \lambda \times l_{ERBP} \tag{5.1}$$

where l_c is the classification loss, l_{ERBP} is

$$l_{ERBP} = \begin{cases} \sum_{i=1}^{k}(w_i - d_i)^2, & \text{if } L2 \\ \sum_{i=1}^{k}|(w_i - d_i)|, & \text{if } L1 \end{cases}$$

and λ is the regularisation parameter, corresponding to the inverse of the variance of the prior, effectively regulating the strength of the ERBP regularisation. These methods are called ERBP L1 and ERBP L2 respectively in the rest of the chapter.

The ERBP approach can be applied to feed froward and to recurrent networks. The only additional condition for recurrent networks is that all n items on which abstract patterns need to be detected are represented in the input vector, i.e. a sliding window of size n is needed.

5.2 Experiments: Abstract Pattern Learning

In this section, the ERBP method is evaluated on generalising abstract relations from training data. ERBP is compared to standard neural networks and Early and Mid-Fusion RBP, and also evaluated on the mixed abstract and concrete patterns.

5.2.1 Setup

For all tasks on synthetic data, a standard feed-forward neural network is used. A grid search over hyper-parameters is run: the number of epochs with values [10,20,30], and the number of neurons in the hidden layer with values [10,20,30], except for ERBP where minimal number necessary for the ERBP matrix is used.

The regularisation parameter λ is varied with values [0.01,0.03,0.1,0.3,1,3,10,30] for the ERBP approaches. The train/test split was set to 75% / 25% for all tasks. A total of 10 train-test runs is run using Adam optimiser and Stochastic Gradient Descent (SGD) is also for comparison. A single hidden layer and a mini-batch size of 1 is used unless indicated otherwise. The networks have been implemented using the PyTorch library[1].

5.2.2 Abstract Patterns

In this experiment, the aim is to learn classification based on abstract patterns (as shown in Chapter 3) based on equality as in the experiments by Marcus et al. (1999). Triples of items (α, β, γ) are presented to the network, following the abstract patterns AAA, AAB, ABA, ABB and ABC. These abstract patterns can be described in logic using a binary equality predicate $eq(\cdot, \cdot)$. E.g., the abstract patterns ABA and ABB can be represented by the following predicates:

$$ABA : \neg eq(\alpha, \beta) \wedge eq(\alpha, \gamma) \tag{5.2}$$

$$ABB : \neg eq(\alpha, \beta) \wedge eq(\beta, \gamma). \tag{5.3}$$

These predicates only depend on the equality relations and not the values of $\alpha, \beta,$ and γ. They are also called as algebraic patterns or abstract rules (Dehaene et al., 2015; Marcus, 2001). A vocabulary of size 12 is used for the task, e.g. letters 'a...l', which are presented to the network in one-hot encoded vectors. Five different experiments are performed as follows. The various cases correspond to the experiment in Marcus et al. (1999), where only one rule-based pattern type is used for familiarisation.

1. ABA vs other: In this task, one class contains only patterns of ABA and the other class contains all the other possible patterns (AAA, AAB, ABB, ABC), downsampled for class balance. Expressed in logic, the task is to detect whether $eq(\alpha, \gamma) \wedge \neg eq(\alpha, \beta)$ is true or false.

2. ABB vs other: In this task, one class contains only patterns of the form ABB and the other class contains all the other possible patterns (AAA, AAB, ABA, ABC), downsampled for class balance. The logical task is to detect $eq(\beta, \gamma) \wedge \neg eq(\alpha, \beta)$.

[1]http://pytorch.org

3. ABA-BAB vs other: In this task, class one contains patterns of type ABA as in 1. For ABA sequences in the training set, the corresponding BAB sequences appear in the test set. The other class contains all other possible patterns (AAA, AAB, ABB, ABC) downsampled per pattern for class balance as before.

4. ABC vs other: In this case, class one (ABC) has no pair of equal tokens, while the *other* class has at least one of $eq(\alpha, \beta), eq(\alpha, \gamma), eq(\beta, \gamma)$ as *true*, i.e. detecting equalities without localising them is sufficient for correct classification.

5. ABA vs ABB: This task is like task 1 above, but only pattern ABB occurs in the other class, so that this task has less variance in the second class. This task is expected to be easier to learn because two equality predicates $eq(\alpha, \gamma), eq(\beta, \gamma)$ change their values between the classes and are each sufficient to indicate the class.

Type	1)	2)	3)	4)	5)
Standard	50 (1.86)	50 (1.83)	50 (1.73)	50 (1.68)	50 (1.81)
Early Fusion	65 (1.26)	65 (1.29)	75 (1.22)	69 (1.04)	55 (1.18)
Mid Fusion	100 (0.00)	100 (0.00)	100 (0.05)	100 (0.00)	100 (0.00)
ERBP L1	100 (0.00)	100 (0.00)	100 (0.02)	100 (0.00)	100 (0.00)
ERBP L2	100 (0.00)	100 (0.00)	100 (0.00)	100 (0.00)	100 (0.00)

Table 5.1 Accuracy (in %) and standard deviation over 10 simulations (in brackets) using different models for Abstract Pattern Learning: 1) ABA vs other, 2) ABB vs other, 3) ABA-BAB vs other, 4) ABC vs other, 5) ABA vs ABB).

In Table 5.1 (Abstract patterns) test set accuracies are presented, the training set accuracy was 100% in all cases. It is found that abstract patterns are almost perfectly generalised with ERBP L1 and ERBP L2, like with Mid Fusion RBP. The differences between network performances across 10 simulations are compared using the Wilcoxon Signed Ranked Test. The differences between ERBP L1/L2 and Standard as well as between ERBP L1/L2 and Early Fusion were statistically significant in all Abstract Pattern experiments at a threshold of $p < 0.05$. Those between Mid Fusion, ERBP L1 and ERBP L2 were not, which was expected as ERBP is based Mid-Fusion RBP.

5.2.3 Parameter Variations

The effect of several parameters are studied: number of hidden layers, choice of optimizer, regularisation factor and weight initialisation on abstract pattern learning task as detailed below.

Network Depth: The abstract pattern learning experiment is tested with deeper neural network models, using $h = 2, 3, 4, 5$ hidden layers. The results are tabulated in Table 5.2, showing only minor improvements in the network performance for deeper networks. However, Mid Fusion RBP and ERBP generalisation is consistent and independent of the network depth.

Hidden Layer	No RBP	Early Fusion	Mid Fusion	ERBP L1	ERBP L2
h=2	50 (1.56)	65 (1.23)	100 (0.05)	100 (0.00)	100 (0.00)
h=3	52 (1.59)	66 (1.08)	100 (0.03)	100 (0.00)	100 (0.00)
h=4	55 (1.63)	68 (1.12)	100 (0.02)	100 (0.00)	100 (0.00)
h=5	56 (1.55)	70 (0.89)	100 (0.02)	100 (0.00)	100 (0.00)

Table 5.2 Classification accuracy (in %) along with standard deviation (test) for abstract pattern learning (ABA vs other) using deeper networks with hidden layers h= 2,3,4,5.

Type	ERBP L1	ERBP L2
Adam	100 (0.00)	100 (0.00)
SGD	98 (0.06)	96 (0.04)

Table 5.3 Accuracy (in %) and standard deviation of abstract pattern learning (ABA vs other) using Adam and SGD optimiser for ERBP L1 and L2. SGD can also lead to 100% accuracy on the test set, but needs higher λ values.

Optimiser: Both Stochastic Gradient Descent (SGD) and the Adam optimiser are used for training the ERBP. It is observed that faster convergence and greater improvement in the overall accuracy is achieved with the Adam optimiser compared to SGD even though SGD by itself also led to good performance. Similar results were observed for both ERBP L1 and L2.

Table 5.3 summarises the results of abstract pattern learning for both the optimisers with the regularisation parameter λ set to 1. It is observed that the SGD does not reach full generalisation in this setting, however it does so at higher values of λ.

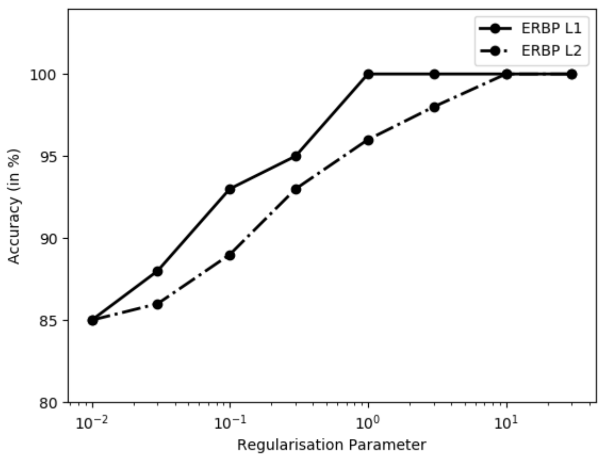


Fig. 5.4 Accuracy (in %) of the network with ERBP L1 and L2 when varying the regularisation parameter λ (in logarithmic scale) for abstract pattern learning (ABA vs other).

Regularisation Factor: The regularisation factor λ is varied in the loss function of the ERBP model and it is observed that a large factor reliably leads to perfect generalisation in abstract pattern learning task. Figure 5.4 shows how the effect depends on the size of the regularisation factor λ using L1 and L2 loss functions for abstract pattern learning task.

Weight initialisations: The effect of different weight initialisations is also tested without the ERBP weight prior, i.e. without the weight matrix D as shown in Figure 5.3. For random initialisation, weight and bias are initialised from a uniform distribution over *(-stdv, stdv)* where $stdv = 1/\sqrt{n}$ and $n =$ size of the weight matrix. The results of the network using random initialisation, zero weight initialisation, Xavier initialisation (Glorot and Bengio, 2010) and ERBP initialisation are shown in Table 5.4. Although ERBP initialisation shows the best generalisation in this comparison, it falls far short of the performance with the ERBP weight loss (see Table 5.1).

5.2.4 Mixed Abstract and Concrete Patterns

An important question about any inductive bias for abstract patterns is whether it has negative effects on other tasks. Here concrete patterns are used as test cases.

Initialisation:	Random	Zero	Xavier	ERBP
Task 1	53 (1.82)	51 (1.67)	54 (1.53)	**61** (1.12)
Task 2	53 (1.76)	52 (1.72)	54 (1.57)	**61** (1.09)
Task 3	55 (1.73)	54 (1.59)	56 (1.49)	**65** (1.04)
Task 4	52 (1.89)	54 (1.62)	55 (1.62)	**60** (1.23)
Task 5	52 (1.54)	53 (1.58)	54 (1.67)	**62** (1.20)

Table 5.4 Classification accuracy (in %) and standard deviation with different weight initialisation methods on standard neural networks for various tasks ie. Task 1-5 Abstract Pattern Learning (ABA vs other, ABB vs other, ABA-BAB vs other, ABA vs ABB, ABC vs other)

An experiment is conducted where the classes were defined by combinations of abstract and concrete patterns similar to the task defined in Chapter 3. Specifically four mixed classes are defined based on combinations of the abstract patterns ABA and ABB with the concrete patterns $a**$ and $b**$.

Type	Accuracy
Simple Network	23 (1.67)
Early Fusion	49 (1.52)
Mid Fusion	100 (0.00)
ERBP L1	100 (0.00)
ERBP L2	100 (0.00)

Table 5.5 Accuracy (in %) and standard deviation for the Mixed pattern learning task using various approaches of RBP. Both Mid Fusion and ERBP approaches result in 100% accuracy as shown above.

The results of the experiment are given in Table 5.5 and show that the mixed patterns are perfectly generalised by Mid Fusion RBP and ERBP. This shows that the (E)RBP does not impede neural network learning of concrete patterns. A range of values for λ are tried and it is found that a minimum was necessary for learning but higher values did not impede the learning of the concrete patterns.

5.3 Generalising and Learning of Identity/Equality Relations

For the task of learning identity relations, synthetic data is generated and a standard feed-forward neural network is used. The input vector is binary and the target values are $[0, 1]$ for unequal and $[1, 0]$ for equal vector halves.

A grid search over hyper-parameters: the number of epochs varied as [10,20,30], the number of neurons per hidden layer was varied as [10,20,30]. For the Bayesian weight prior, the regularization parameter λ was varied with values [0.01,0.03,0.1,0.3,1,3,10,30]. A total of 10 simulations was run using the SGD and Adam optimizers, training for 20 epochs. A single hidden layer was used and a batch size of 1 unless indicated otherwise. The networks have been implemented using the PyTorch library[2].

The train/test split was set to 75% / 25% for all tasks. The vector dimensionalities tested were $n = 3, 10, 30$. All vectors with equal halves are generated and then a random sample of all those with unequal halves was taken to balance the classes. The size of the dataset was down-sampled to 1000 when it is greater.

5.3.1 Identity Relations

Identity is an abstract relation in the sense that it is independent of the actual values of the individual arguments, it just depends on their combined configuration. In this task, pairs of vectors are presented to a feed-forward network and the task is to distinguish whether the two vectors are equal or not. The performance of the network was evaluated in different configurations on a held-out test set and the results are tabulated in Table 5.6 for different vector dimensions.

As observed in experiments in Chapter 3, standard networks do not improve much over random guessing, while ERBP L1 and ERBP L2, as well as Mid Fusion, almost always achieve perfect generalisation with sufficiently strong λ (see next section for details).

[2]http://pytorch.org

Type	Standard	Early Fusion	Mid Fusion	ERBP L1	ERBP L2
$n = 3$	55 (1.91)	65 (1.34)	100 (0.04)	100 (0.00)	100 (0.00)
$n = 10$	51 (1.67)	65 (1.32)	100 (0.08)	100 (0.04)	100 (0.02)
$n = 30$	50 (1.52)	65 (1.27)	100 (0.07)	100 (0.05)	100 (0.04)

Table 5.6 Test set classification accuracy (in %) and standard deviation over 10 simulations (in brackets) using different models for Identity Learning (vector dimensions $n = 3, 10, 30$). The networks were trained with the Adam optimizer for 20 epochs.

5.3.2 Parameter Variations

The effect of several parameters like number of hidden layers, choice of optimizer, regularization factor and weight initialization on identity relation learning tasks is also studied. More details are given below in this section.

Network Depth: The identity learning task is tested with deeper neural network models, using $h = 2, 3, 4, 5$ hidden layers. The results are tabulated in Table 5.7, showing only minor improvements in the network performance for deeper networks. However, ERBP L1 and L2 generalization is consistent and independent of the network depth.

Hidden layers	No RBP	Early Fusion	Mid Fusion	ERBP L1	ERBP L2
h = 2	55 (1.65)	65 (1.26)	100 (0.02)	100 (0.00)	100 (0.00)
h = 3	55 (1.67)	67 (1.14)	100 (0.03)	100 (0.00)	100 (0.00)
h = 4	58 (1.63)	68 (1.25)	100 (0.02)	100 (0.00)	100 (0.00)
h = 5	59 (1.68)	72 (1.23)	100 (0.02)	100 (0.00)	100 (0.00)

Table 5.7 Test set classification accuracy (in %) with standard deviation (in brackets) for identity learning ($n = 3$) using deeper networks. The networks were trained with the Adam optimizer for 20 epochs.

Optimiser: Both Stochastic Gradient Descent (SGD) and the Adam optimizer are used for training the ERBP L1 and L2. Overall faster convergence and greater improvement in the overall accuracy was observed with the Adam compared to SGD. Similar results were observed for both ERBP L1 and L2. Table 5.8 summarizes the

Type	ERBP L1	ERBP L2
Adam	100 (0.00)	100 (0.00)
SGD	98 (0.06)	96 (0.05)

Table 5.8 Accuracy (in %) and standard deviation of identity learning ($n = 3$) using Adam and SGD optimizer for ERBP L1 and L2. SGD can also lead to 100% accuracy on the test set, but needs higher λ values.

results of identity learning for both the optimizers with the regularization parameter λ set to 1. It is observed that the SGD does not reach full generalization in this setting, however it does so at higher values of λ.

Regularization Factor: The regularization factor λ in the loss function of the ERBP models is varied to see its effect in the overall performance. It is observed that a factor of 3 or above reliably leads to perfect generalization in identity learning task. Figure 5.5, shows how the effect depends on the size of the regularization factor λ using L1 and L2 loss functions for learning identity relations.

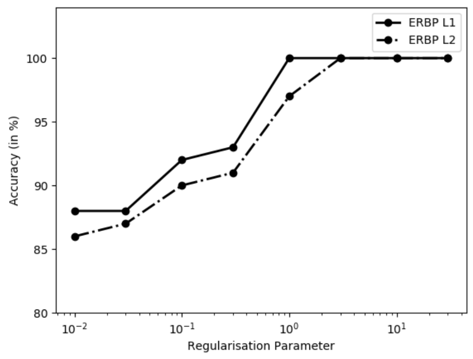


Fig. 5.5 Test set classification accuracy (in %) of the network with ERBP L1 and L2 when varying the regularization parameter λ (shown in logarithmic scale) for identity learning with ($n = 3$) and using the Adam optimizer.

5.3.3 Jointly learning identity relations and bit patterns

In order to test, whether the RBP and ERBP impedes other learning tasks, learning with some non-relational patterns that are based on values of specific neurons are also tested. The simplest form of the task considered is to determine the target class based on one bit. This case is learned with 100% generalization performance for a network with 2 additional outputs for the pattern classification.

Furthermore, the learning of classification based on all even or odd elements in the vector being 0 is also tested. In this case, a perfect generalization is achieved for λ up to 10. For $\lambda = 30$, first deterioration of the accuracy is observed, but there is a wide range of values where both the identity relations and the even/odd classification generalize perfectly. The results with ERBP are comparable to the RBP in Mid Fusion.

5.4 Experiments: Language Modeling

In order to evaluate the effect of ERBP on language models, further experiments are performed with real world data on character and word prediction in natural language and also on pitches in melodies. RNN, GRU and LSTM models are used for character, word and pitch prediction.

The input vectors represent items in a context window of length n. Therefore there are $n \times (n-1)/2$ pairs of vectors to compare. The ERBP is applied in the same way as before in the first hidden layer after the input. In these tasks, the encoding of the information is not binary, as in the synthetic tasks, but integer-based or continuous. Thus, the differences in DR units and the ERBP hidden neurons represent not just equality but a form of distance of the input values, which is expected to be useful for prediction.

5.4.1 Character and Word Prediction

For the experiments on character and word prediction, Wikitext-2 (Merity et al., 2016) has been used. For word prediction, the entire dataset with 2 million words was used with train/validation/test split as in the original dataset with pretrained Glove embeddings (Pennington et al., 2014) with 50 dimensional vectors. For character prediction, a truncated version with $60,000$ words was used with a train/validation/

test split of 50/25/25 and with integer encoding of the characters. The predictions are evaluated by perplexity. Perplexity is calculated as follows :

$$P(p) = 2^{H(p)} = 2^{-\Sigma_x p(x) \log q(x)}$$

where, $H(p)$ is the cross entropy between original distribution p and predicted distribution q and x is the set of events. Here the base is 2, in general the perplexity is independent of the base. The lower the perplexity, the better the result.

Type	n=10	n=20	n=30	n=40
RNN	21.99	21.73	21.42	21.47
Early Fusion	21.90	21.64	21.35	21.24
Mid Fusion	21.89	21.60	21.33	21.20
ERBP L1	20.66	**20.50**	20.04	20.32
ERBP L2	**20.60**	20.53	**20.03**	**20.27**

Table 5.9 Perplexity per character for various context lengths $n = 10,20,30,40$ with RNNs on the Wikitext-2 dataset truncated to 60k words with integer encoding tested on plain RNN, Early, Mid Fusion, ERBP L1 and L2. The best results (in bold) occur for the ERBP approaches with $\lambda = 0.3$.

A network with 2 hidden layers with 50 neurons each is used. The networks were trained for up to 30 epochs for character prediction task and up to 40 for word prediction task. The learning rate was set to 0.01. Training typically converges after less than 30 epochs and the best model is selected according to the validation loss. The regularisation parameter λ is tested using values [0.01,0.03,0.1,0.3,1,3] for ERBP. A context length n of [10,20,30,40] is used and evaluated again with ERBP L1 and L2, RBP Early and Mid Fusion, and standard networks.

The results for character prediction and word prediction are summarised in Tables 5.9, 5.10, 5.11 and 5.12, 5.13, 5.14 for RNN, GRU and LSTM models respectively. The performance was best for LSTMs with ERBP and the choice of λ value didn't have a major effect in the case of word prediction, unlike character prediction where $\lambda = 0.3$ gave best performance.

Type	n=10	n=20	n=30	n=40
GRU	21.97	21.72	21.39	21.45
Early Fusion	21.89	21.63	21.33	21.20
Mid Fusion	21.87	21.59	21.30	21.19
ERBP L1	20.61	**20.43**	20.02	20.25
ERBP L2	**20.53**	20.46	**19.96**	**20.23**

Table 5.10 Perplexity per character for various context lengths $n = 10,20,30,40$ with GRUs on the Wikitext-2 dataset truncated to 60k words with integer encoding tested on plain GRU, Early, Mid Fusion, ERBP L1 and L2. The best results (in bold) occur for the ERBP approaches with $\lambda = 0.3$.

Type	n=10	n=20	n=30	n=40
LSTM	21.88	21.64	21.34	21.33
Early Fusion	21.84	21.63	21.27	21.14
Mid Fusion	21.81	21.57	21.24	21.16
ERBP L1	20.54	**20.32**	19.89	**20.18**
ERBP L2	**20.45**	20.33	**19.87**	20.19

Table 5.11 Perplexity per character for various context lengths $n = 10,20,30,40$ with LSTMs on the Wikitext-2 dataset truncated to 60k words with integer encoding tested on plain LSTM, Early, Mid Fusion, ERBP L1 and L2. The best results (in bold) occur for the ERBP approaches with $\lambda = 0.3$.

Wilcoxon signed rank test has been used with threshold p=0.05 as above to compare the standard models with ERBP L1 and L2 for both character and word prediction across 60000 words for context length $n = 10$. For character prediction, RNN, GRU and LSTM with ERBP L1 and L2 are significantly better than the standard RNN, GRU and LSTM models. For word prediction, all ERBP models except LSTM ERBP L1 are significantly better than the corresponding standard models.

The results show that the prediction performance benefits from ERBP in all tested tasks: character and word prediction in natural language, and pitch prediction in melodies. The prediction improvements are relatively small, but very consistent and statistically significant. There is little difference in performance between ERBP L1 and L2.

The size of the improvements achieved by ERBP is not dramatic, but its consistency shows that standard neural networks do not detect patterns of a pair-wise equality

Type	n=10	n=20	n=30	n=40
RNN	186.97	164.56	114.26	101.95
Early Fusion	186.80	164.52	114.21	101.89
Mid Fusion	186.52	164.41	114.19	101.69
ERBP L1	185.83	163.98	**113.96**	100.25
ERBP L2	**185.54**	**163.90**	113.99	**100.23**

Table 5.12 Perplexity per word for various context lengths n = 10,20,30,40 with RNNs on the Wikitext-2 dataset consisting of 2 million words with integer encoding tested on plain RNN, Early, Mid Fusion, ERBP L1 and L2. The best results (in bold) occur for the ERBP approaches.

Type	n=10	n=20	n=30	n=40
GRU	174.62	160.31	114.05	97.62
Early Fusion	174.35	160.26	113.86	96.39
Mid Fusion	174.42	160.27	113.55	96.35
ERBP L1	173.33	159.36	113.42	**96.21**
ERBP L2	**173.21**	**159.32**	**113.32**	96.32

Table 5.13 Perplexity per word for various context lengths n = 10,20,30,40 with GRUs on the Wikitext-2 dataset consisting of 2 million words with integer encoding tested on plain GRU, Early, Mid Fusion, ERBP L1 and L2. The best results (in bold) occur for the ERBP approaches.

Type	n=10	n=20	n=30	n=40
LSTM	160.44	153.61	113.40	93.28
Early Fusion	160.32	153.42	113.26	93.23
Mid Fusion	160.38	153.30	113.23	93.15
ERBP L1	159.29	**152.78**	113.02	92.96
ERBP L2	**159.22**	152.89	**112.99**	**92.82**

Table 5.14 Perplexity per word for various context lengths n = 10,20,30,40 with LSTMs on the Wikitext-2 dataset consisting of 2 million words with integer encoding tested on plain LSTM, Early, Mid Fusion, ERBP L1 and L2. The best results (in bold) occur for the ERBP approaches.

or similarity between tokens. Adding a structure in the form of RBP and ERBP does improve predictions, providing evidence that addressing systematicity in neural network learning is not just a theoretical issue, but that it has an impact on practical tasks. By exploring these systematicity issues further, it may be possible to achieve larger improvements, but that was outside the scope of this PhD work.

The perplexity values shown above for character and word prediction are not the desired (best) values i.e. i.e. not the state of the art (SOTA) values, we can arrive at the best perplexity with proper fine-tuning. The aim of the task was not to improve the SOTA performance in language modeling, but to evaluate effect of RBP/ERBP using the standard recurrent neural network models.

5.4.2 Melody Prediction

In another experiment, effect of ERBP has been tested for predicting the pitch of the next note in melodies with a selection of the Essen Folk Song Collection (Schaffrath, 1995b), which comprises songs in 8 different datasets. Pitches are integer-encoded. Using RNN, GRU and LSTM two hidden layers are used each of size 20.

Type	RNN	GRU	LSTM
Standard	2.8012	2.7968	2.7928
Early Fusion	2.7365	2.7328	2.7314
Mid Fusion	2.7356	2.7302	2.7219
ERBP L1	2.7264	**2.7252**	**2.7132**
ERBP L2	**2.7240**	2.7260	2.7144

Table 5.15 Average cross entropy per note for pitch prediction using context length $n = 5$ on the Essen Folk Song dataset using standard models, Early, Mid fusion and ERBP approaches. The best results (in bold) are achieved with the ERBP approaches at $\lambda = 0.3$.

For optimisation, Adam is used with a learning rate of 0.01 and the values of [0.01, 0.03, 0.1, 0.3, 1,3] were tested for the regularisation factor λ. Context lengths of both 5 and 10 are tested and the train/validation/test split was set to 50/25/25%.

Tables 5.15 and 5.16 summarise the results of the neural networks without and with RBP in Early, Mid fusion and using ERBP L1 and L2. For statistical significance

Type	RNN	GRU	LSTM
Standard	2.7502	2.7457	2.7432
Early Fusion	2.6735	2.6723	2.6712
Mid Fusion	2.6653	2.6650	2.6648
ERBP L1	**2.6642**	**2.6641**	2.6639
ERBP L2	2.6649	2.6645	**2.6632**

Table 5.16 Average cross entropy per note for pitch prediction using context length $n = 10$ on the Essen Folk Song dataset using standard models, Early, Mid fusion and ERBP approaches. The best results (in bold) are achieved with the ERBP approach at $\lambda = 0.3$.

over the results for the 8 subsets, we tested using the Wilcoxon signed rank test with threshold $p = 0.05$ as in above experiments. It is found that all model types (Standard/Early/Mid/ERBP L1/L2) are different with statistical significance, and ERBP L1 and L2 outperform standard networks and RBP models for this task. The results on character, word and pitch prediction are not close to the SOTA model perplexities, as the aim of experiment to test the effect of RBP and ERBP in each of the experiments, and not to improve the state of the art performance. The perplexities values can be made better with proper hyper-parameter turning and configuration.

5.5 Observations and Remarks

A new approach called ERBP as a re-modeling of RBP as a Bayesian prior is introduced in this chapter, because of the simple integration into standard recurrent and feed-forward network structures as a regularisation term and because it retains the full flexibility of the network learning. The experiments in the chapter show that the ERBP L1 and L2 models are effective in learning classification based on abstract patterns. ERBP is even slightly more effective than the original RBP formulation. The learning of abstract patterns with ERBP is robust over a wide range of parameter settings as shown. ERBP also does not limit the network's learning of concrete pattern in combination with abstract patterns. This is an empirical confirmation that was to be expected, given that with ERBP the network is still a universal approximator as the proof of (Leshno et al., 1993) still applies.

The second point of this study was, whether the enabling of the abstract pattern learning would improve neural networks learning on real-world data, was addressed in the second set of experiments in the later part of the chapter. The results show that the prediction performance benefits from ERBP in all tested tasks: character and word prediction in natural language, and pitch prediction in melodies. The prediction improvements are relatively small, but very consistent and statistically significant. There is little difference in performance between ERBP L1 and L2. Overall, this provides some evidence that addressing systematicity in neural network learning is not just a theoretical issue, but that it has a impact on practical tasks.

In general, identity based relations are fundamental to relational learning. In this work, it is observed that creating a weight prior on the network weights leads to generalisable solutions of learning identity based relations. It is believed that addressing these issues and coming up with effective solutions is necessary for more higher level relational learning tasks and also is relevant to address problems in general neural network learning. While most approaches to learning abstractions focus on the structure of the network and non-standard operators, this is to our knowledge the first approach that addresses abstract pattern recognition with a prior on the network weights.

In future, this work can be extended towards learning other complex relational learning tasks and different neural network architectures as shown in the next chapter. The (E)RBP approach has been applied on other tasks like abstract compositionality and tested on recently popular Transformer models and Graph neural networks. More details in the next chapter.

Chapter 6

Experimenting with (E)RBP in different architectures and tasks

The E(RBP) method described in Chapter 5, so far has been experimented on language modeling tasks using RNN and their gated variants GRUs and LSTMs. In this chapter, the ERBP method is extended and applied to the state of the art language models and recently popular neural network architectures i.e. Transformers.

Transformers have self-attention based representation in their core architecture with absolute and relative positional encodings (Shaw et al., 2018; Vaswani et al., 2017). ERBP is tried here as a weight prior for the positional encoding as a variant of the absolute / relative positional encoding used in the standard Transformer models. After trying the relational representations, (E)RBP is used on a different task i.e. abstract compositionality comparison.

Further, ERBP is tested on the graph neural networks (Zhou et al., 2018), specifically on the graph edit distance task, as the task involves learning similarity between two graphs. The standard similarity function in graph neural networks is replaced with ERBP i.e. a default weight prior is used and the rest of the network remains the same. This work specifically explores how ERBP performs on different relational learning tasks using Transformers and graph neural networks exploring aspects of compositionality and similarity/ distance based learning.

6.1 Transformer Models with (E)RBP

A Transformer model introduced by (Vaswani et al., 2017) is used in this work with embedding and hidden dimension size set to 20, 2 hidden layers with learning rate set to 0.02, regularization parameter $\lambda = 0.3$, number of attention heads = 2 and sequence length 'n' = [5,10,20,30,50] and batch size 50. The Transformer model with ERBP with $X \in x_1....x_n$ as the input context with sequence length 'n' comprises

$$\begin{aligned} &l_0 = X \times D \\ &l_1 = l_0 \times W_e + W_p \\ &l_i = transformerlayer(Q_i, K_i, V_i) \forall i \in [2, n-1] \\ &l_n = P(y|x_1....x_n) = softmax(l_{n-1} \times W_e^t), \end{aligned} \tag{6.1}$$

where $l_0...l_n$ are the layers of the Transformer network, X is the input context, D is the default weight matrix as shown above with different dimensions n = [5,10,20,30,50], W_e is the embedding matrix, W_p is the position embedding matrix. Each of the layers calculates the matrix vector multiplication ($\times$)

Self-Attention is applied over the input context in the transformer layer with Q_i, K_i and V_i query, key and vector values. For an i^{th} query token, attention weights with respect to other tokens at position 'j' are calculated as $\frac{\exp(\mathbf{q}_i \mathbf{k}_j^\top)}{\sqrt{d_k} \sum_{r \in S_i} \exp(\mathbf{q}_i \mathbf{k}_r^\top)} \mathbf{v_i}$ where S_i is the set of all the key positions to attend for the i^{th} query. Finally a feed forward layer is used to produce an output probability distribution for target 'y'.

Absolute and Relative Attention are similar to RBP in Early Fusion models as the positional encodings in both cases are concatenated to the input tokens directly. In absolute attention, positional encodings are computed based on random sine and cosine wave functions with different frequencies (Vaswani et al., 2017). In Relative Attention, pairwise difference is calculated between tokens in a given context (Shaw et al., 2018), while in RBP, the DR units calculate the absolute of the difference. In Mid Fusion RBP, the DR units are concatenated to the hidden layer. In ERBP, a weight prior is used that is modeled on a modified RBP Mid Fusion model.

6.1.1 Music Language Modeling

The set up for this experiment is similar to the one mentioned in Chapter 4 using the Essen Folk Song Collection and similar hyper-parameter configuration. The input context length is varied in the range [5,10,20,30,50] as the largest length of the melody was 50. Firstly, experiments are run without RBP and then with RBP in Early, Mid and Late Fusion using LSTMs as a baseline. After that ERBP is compared with the RBP models and further with other relational representations(absolute and relative). A grid search is performed on the number of epochs [20,30,50] and number of hidden layers was set to 2 with Adam optimizer. The model usually converged with less than 30 epochs. The learning rate was set to 0.02. As loss function and main evaluation metric cross entropy was used. The cross entropy H between the original distribution p and predicted distribution q is defined as

$$H(p,q) = -\Sigma_{x \in S}\, p(x) \log q(x) \tag{6.2}$$

where S is the set of possible events, i.e. pitches. Lower cross-entropy values indicate better prediction. The results for the LSTM model with and without RBP are shown in Table 6.1. This is similar to experiment results given in Table 4.11, however the experiments here are done on longer context lengths i.e. n = [5,10,20,30,50] unlike in Chapter 4, where smaller context lengths of n are tested.

Context Length	Without RBP	With RBP			ERBP
		Early Fus	Mid Fus	Late Fus	
n= 5	2.592	2.531	2.521	2.506	2.502
n= 10	2.432	2.401	2.391	2.386	2.372
n= 20	2.025	2.015	2.012	1.994	1.992
n= 30	1.985	1.976	1.973	1.970	1.963
n=50	1.976	1.964	1.952	1.949	1.946

Table 6.1 Average Cross entropy loss with and without RBP and ERBP across 8 datasets using LSTMs for various context lengths n with integer encoding.

LSTMs with RBP in late fusion performed the best overall when comparing the RBP and all the results were statistically significant with $p < .05$ using the Wilcoxon signed rank test. Further, LSTM with ERBP seems to be slightly better than RBP in late fusion when evaluated on different context lengths, however the difference is very small.

Experiments are now conducted with the Transformers using different relational representations i.e. Absolute Attention, Relative Attention and the ERBP approach. Table 6.2 has the results with different context lengths. RBP in late fusion has been used as the baseline as it gave the best performance so far in other experiments.

Context	Transformers			
Length	Late Fusion	Absolute	Relative	ERBP
n = 5	2.421	2.321	2.304	2.287
n = 10	2.326	1.952	1.946	1.932
n = 20	2.016	0.983	0.973	0.971
n = 30	1.954	0.953	0.942	0.938
n= 50	1.952	0.906	0.893	0.886

Table 6.2 Average Cross entropy loss with and without RBP across 8 datasets using Transformer models with different context lengths n and context representations with integer encoding.

As shown in Table 6.2, ERBP with Transformers outperformed all other models when tested on the Essen Folk Song Collection giving a new state of the art in monophonic pitch prediction task as shown in Table 6.3. All Transformer models except late fusion benefit dramatically from increased input context lengths of 20 and more. When comparing performance across different context lengths for the different subsets of the Essen Folk Song Collection, the differences in the results are statistically significant at $p < .05$ using the Wilcoxon signed rank test.

As shown in Table 6.3, Transformers drastically outperform LSTMs on the monophonic pitch prediction task. The difference between each of the Transformers models i.e. with absolute, relative and ERBP is very small, however ERBP with Transformers performs slightly better overall.

Model Type	Avg Cross Entropy
(Cherla et al., 2015)	2.479
Langhabel (et.al., 2017)	2.395
Transformers (absolute)	0.906
Transformers (relative)	0.893
Transformers (with ERBP)	**0.886**

Table 6.3 Comparative performance with state of the art monophonic pitch prediction models. Transformers with ERBP leads to best performance overall.

Further experiments are also conducted using the melodies of chorales by J.S. Bach Chorales and from the Nottingham dataset[1], using the same configuration as in the previous experiment. The results are shown in Table 6.4. A similar trend is observed here as well, i.e. that the Transformers using ERBP performed better than relative and absolute attention. However, the difference between each of the Transformer models is very minimal, the main significant difference is seen from LSTMs to Transformers when tested across various context lengths.

Model Type	JS Bach	Nottingham
Transformers (absolute)	1.893	2.234
Transformers (relative)	1.862	2.123
Transformers with ERBP	**1.856**	**2.119**

Table 6.4 Performance of Transformer model types on JSB Chorales and Nottingham datasets with different contextual representations for input size $n = 50$.

6.1.2 Relational Mapping Approach

Two different variants of Transformers have been experimented in late fusion and mid fusion settings. The process is called relational mapping, as the mapping between DR input and output units is mapped back into the standard output distribution of the Transformers. Both the processes of late fusion and mid fusion relational mapping are

[1]The datasets are available here: http://www-etud.iro.umontreal.ca/~boulanni/icml2012.

outlined below. The key hypothesis is that the abstract relations in the input context have an influence on the relations between the context and the predicted word.

The relation between the input context units and the corresponding output are modeled as shown in Figures 6.1 and 6.2 respectively. The difference between late fusion and mid fusion lies in the way the offsets are added in the neural network. The late and mid fusion shown in Chapter 3 is adapted here to Transformers. The current self-attention approach is briefly described below followed by the late and mid fusion mapping approaches.

Current Self-Attention approach

- Input words converted into word vectors using embedding algorithm
- For each word corresponding query, key and value vector projections are created
- A score is calculated by taking dot product between query vector (current word) and key vector (all the words in the context)
- All the corresponding query, key scores are scaled (divided by 8 in the paper as key vector has dim 64) for having more stable gradients and then normalised (softmax) so they all are positive and sum upto 1
- Each value vector is multiplied with the softmax score and weighted sum of the value vectors is the final attention score

Mapping in Late Fusion

- Input words converted into word vectors using embedding algorithm
- Two parallel neural networks are trained Transformers (NN1) and Transformers (NN2)
- NN1 : Transformer is trained to get standard output distribution with self attention

- NN2 : For each word query,key and value vector projections are created
- Score based on query (current word) and key values (all words in context) using absolute difference and dot product is calculated for input context
- Using the corresponding (query,key) pairs and their scores as ground-truth - score between the key and the next word are computed using a feed-forward network with logistic sigmoid as activation function
- Final scores are normalised and projected to the vocabulary space to produce value vectors
- These value vectors are added to the output distribution vector in NN1 based on a set threshold using a gated mixture product of experts model

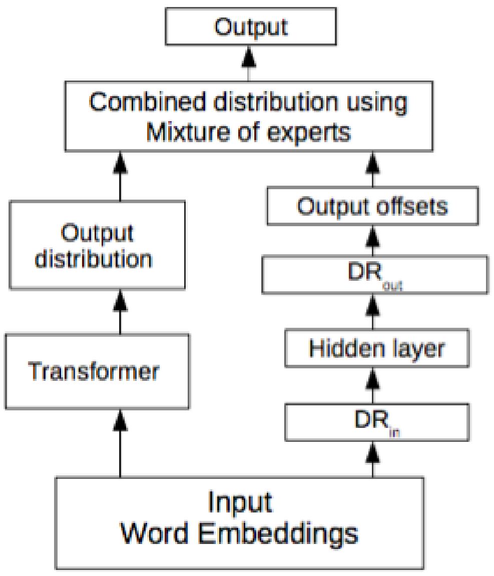


Fig. 6.1 Relational Mapping in Late Fusion. This is the ERBP Late Fusion model with Transformers where the DR offsets are normalised and projected back to the vocabulary space (NN2) and added to the output probability distribution of the standard Transformer (left part i.e. NN1).

Mapping in Mid Fusion

- Input words converted into word vectors using embedding algorithm
- Two parallel neural networks are trained Transformers (NN1) and Transformers (NN2)
- NN1 : Transformer is trained to get standard output distribution with self attention
- NN2 : For each word query,key and value vector projections are created
- Score based on query (current word) and key values (all words in context) using absolute difference and dot product is calculated for input context
- Using the corresponding (query,key) pairs and their scores as ground-truth - score between the key and the next word are computed using a feed-forward network with logistic sigmoid as activation function
- These offsets are added to the scores of the self-attention module after normalisation

6.1.3 Results

Tables 6.5 and 6.6 described the performance of different types of Transformer representations. LSTM model is compared as the baseline and tested with four different variants i.e. Transformers with dot product, Transformers with absolute difference, Transformers in late fusion and mid fusion relational mapping as outlined in the above section.

Two different experiments are performed using the WikiText2 dataset, for the first experiment, the entire dataset with 2 million tokens are considered from 720 articles in total and for the other experiment, a truncated version with 50k tokens is used from one single article, the results of which are given in Tables 6.5 and 6.6 respectively. The

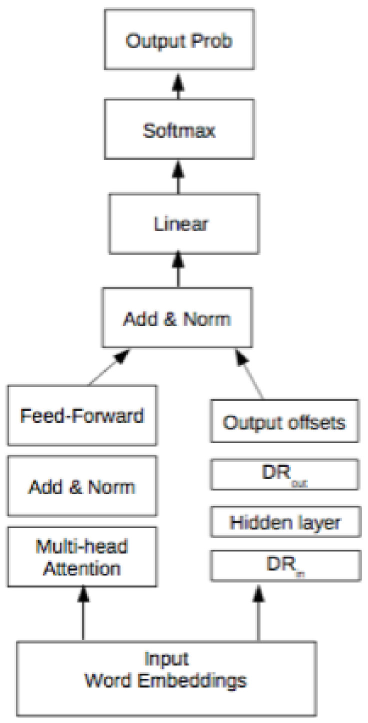


Fig. 6.2 Relational Mapping in Mid Fusion. The left part of the model is the standard Transformer model (NN1) and the right part (NN2) is where the DR unit offsets are added to the attention scores of Transformer model (left part i.e. NN1) after normalisation.

Type	Perplexity
Plain LSTM	85.67
Transformers (dot product)	**77.32**
Transformers (abs diff)	77.36
Transformers(late fusion)	78.07
Transformers(mid fusion)	77.83

Table 6.5 Test set Perplexity of different models on WikiText2 dataset. Here the entire dataset with 2 million tokens is considered with Train/Valid/Test split set to 50/25/25 %. Best results occur with Transformers using dot product for calculating attention scores.

vocab size of the entire 2 million data is 33,278. For the second experiment, only one article is considered with the vocab size of 1230. Train/Valid/Test split for both the experiments are set to 50/25/25 %.

For the hyper-parameter configuration, the embedding and hidden dimension size are set to 20, 2 hidden layers with learning rate set to 0.02, regularization parameter λ = 0.3, number of attention heads = 2 and batch size is set to 50.

Type	Perplexity
Plain LSTM	35.25
Transformers (dot product)	29.32
Transformers (abs diff)	29.64
Transformers(late fusion)	28.62
Transformers(mid fusion)	**28.56**

Table 6.6 Test set Perplexity of different models on truncated WikiText2 dataset (a total of 50k tokens from one article are considered from the 2 million tokens in the entire dataset with 720 articles in total). Train/Valid/Test split is set to 50/25/25 %. Transformers in Mid Fusion relational mapping leads to best performance overall.

It is observed that the relational mapping does seem to have some effect in the truncated setting when compared to the one with the entire dataset. One possible reason could be that for the truncated version, articles from the similar Wikipedia pages are considered, so that could have added to some bias in modeling. In general, it is observed that the Transformers using dot product attention led to the best performance overall as can be seen in Table 6.5.

6.2 Abstract compositionality task

6.2.1 Notion of Compositionality

In another experiment, the effect of ERBP has been tested on an abstract compositionality task in natural language data. Previously, ERBP was tested on synthetic data generated based on abstract rules such as ABB or ABA. In this task, the notion of abstract rules has been extended to words and sentences formed by words following certain abstract rules on their semantics. The key aspect here is in understanding the compositionality of the words that constitute the sentence. The notion of compositionality here is in the sense of understanding of how words combine to form a sentence, in a way that generalizes to words and sentences that have not previously been encountered. Examples of sentences based on simple abstract rules are given in Table 6.7.

Type	Entailment	Contradiction	No of Pairs
Same	X is more Y than Z	Z is more Y than X	14670
More-Less	Z is less Y than X	X is less Y than Z	14670
Not	Z is not more Y than X	X is not more Y than Z	14670

Table 6.7 Comparisons dataset summary. Set of rules for premise: X is more Y than Z

Table 6.7 holds true for any X, Y and Z that may never have occurred in that combination before. In fact, it is expected of the neural network models to generalize to X, Y and Z that have never been encountered before at all. The goal of this task is to test this generalization ability beyond rule based grammars to words and sentences based on such abstract rules.

6.2.2 Compositional Comparison Dataset

The dataset used in the task is the Compositional Comparison Dataset introduced by Dasgupta et al. (2018). The task is to classify a pair of sentences into neutral, contradiction or entailment originally motivated from the popular Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) natural language inference classification task. In this abstract compositionality task, pairs of sentences are generated which differ by permutation of words, such that each of the pairs represent different relations. This dataset is called as the Compositional Comparison dataset as all the pairs of sentences are compositionally generated by changing the word ordering in the sentences. There are three categories ‘Same’,‘More-less’ and ‘Not’ type as shown in Table 6.7.

For the scope of the experiments in this work, a subset of the dataset is considered. Two categories namely ‘Same’ and ‘More-Less’ type are considered for the task. The task was reduced to classify the given pair of sentences into either Contradiction or Entailment.

For example,

Same Type : A-B sentence pairs differ only in the order of the words.

A : The woman is more cheerful than the man

B : The man is more cheerful than the woman (Contradiction)

A : The woman is more cheerful than the man

B : The woman is more cheerful than the man (Entailment)

More-Less Type : A-B pairs differ by whether they contain the word 'more' or the word 'less'.

A : The woman is more cheerful than the man

B : The woman is less cheerful than the man (Contradiction)

A : The woman is more cheerful than the man

B : The man is less cheerful than the woman (Entailment)

The experiment details and results are outlined below.

6.2.3 Experimental Results

ERBP is applied here on the sentences A and B together (i.e. a specific word from A and same corresponding position word from B) rather than considering all the possible pairs. Glove embeddings (Pennington et al., 2014) are used with a single LSTM layer and a sigmoid layer for classification as the output layer. The dataset consists of about 5k sentences with equal split between each of the categories. The results are tabulated in Table 6.8. Bag of Words (BOW) Multi-Layer Perception (MLP) model that averages the Glove embeddings (Pennington et al., 2014) is used as the baseline for the experiments. The results of the ERBP are compared against the state of art InferSent model (Conneau et al., 2017).

As can be seen from Table 6.8, both the Bag of Words MLP model and InferSent model perform just about 50% chance level for the 'Same' type class. Plain LSTMs without ERBP gave an accuracy of about 50%, similar to the InferSent model for both

Model	Same type	More-Less type
BOW-MLP	50%	30.24%
InferSent model (Conneau et al., 2017)	50.37%	50.35%
LSTM with ERBP	51.32%	50.64%

Table 6.8 Accuracy of various models for abstract compositionality task.

'Same' and 'More-Less' types. LSTMs with ERBP results in about an accuracy of 51.32%. For the 'More-Less' type, the Bag of Words MLP model gives only about 30% accuracy where as the InferSent and the LSTM model with ERBP gives above 50% accuracy.

There is small difference in the overall accuracy of the models after adding ERBP for both the 'Same' and 'More-Less' types. Adding ERBP to LSTMs seems to be slightly better when compared to the Bag of words and InferSent models. So far, ERBP has been applied as a relational positional embedding for the Transformers and as a weight prior for learning compostionality in abstract sentence embeddings on a sentiment classification task. Both the experiments did benefit from ERBP. Up next, ERBP has been applied as a similarity metric in the graph neural networks for the graph edit distance task. The details for the experiment are given in the next section.

6.3 ERBP with Graph Neural Networks

Graph neural networks (GNN) are popularly used in relational learning tasks. In a GNN, a graph based data structure is used where there are nodes and edges to infer relations between entities. The nodes are the items and the edges represent relationship between the items. In this task, the goal is primarily to learn a similarity function between graphs. Given two graphs G_1, G_2, a graph similarity model can be written as a function $f(G_1, G_2)$ that computes a scalar similarity value.

The model is based on embedding each graph independently into a vector and then use an existing distance (or similarity) metric in the vector space to compute the distance between graphs. More concretely, it is defined as

$$d(G_1, G_2) = d_H(embed(G_1), embed(G_2)),$$

where *embed* is a model that maps any graph G into an H-dimensional vector, and d_H is a distance metric in that vector space. Typical examples are the Euclidean distance in $\mathbb{R}^H$, i.e.

$$d_H(x, y) = \sqrt{\sum_{i=1}^{H} (x_i - y_i)^2}$$

, and the Hamming distance in H-dimensional space of binary vectors, i.e.

$$d_H(x, y) = \sum_{i=1}^{H} \mathbb{I}[x_i \neq y_i]$$

.

6.3.1 Graph Embedding Module

Each graph input contains a set of nodes V and edges E. Each node $i \in V$ may have a feature vector x_i associated with it, and each edge $(i, j) \in E$ may also have a feature vector x_{ij} encoding e.g. edge type or attributes.

The embedding model will therefore jointly reason about the graph structure as well as the graph features to come up with an embedding that reflects the notion of similarity described by the training examples.

The embedding graph module has 3 basic steps :

- An encoder with two separate MLPs to learn node and edge representations ie. mappings between x_i and x_{ij}

$$\begin{aligned} h_i^{(0)} &= \mathrm{MLP}_{\mathrm{node}}(x_i) \\ e_{ij} &= \mathrm{MLP}_{\mathrm{edge}}(x_{ij}) \end{aligned}$$

- Iterative message passing inside GNN to compute node presentations that encode local neighbour structure and semantics.

  In the t-th round of message passing, a message vector on each edge is computed, and then each node aggregates all the incoming messages and updates its own representation:

$$\begin{aligned} m_{i\to j} &= f_{\text{message}}(h_i^{(t)}, h_j^{(t)}, e_{ij}) \\ h_i^{(t+1)} &= f_{\text{node}}(h_i^{(t)}, \textstyle\sum_{j:(j,i)\in E} m_{j\to i}) \end{aligned}$$

  MLPs are used for f_{message}, while for f_{node} MLPs or even recurrent neural network cores like LSTMs or GRUs can be used.

- After obtaining the final node representations after T rounds of message passing, they are aggregated to get graph representations $h_G = f_G(\{h_i^{(T)}\}_{i\in V})$ by implementating a simple sum that reduces the node representations into a single vector and then transform it:

$$h_G = \text{MLP}_\text{G}\left(\sum_{i\in V} h_i^{(T)}\right).$$

On top of the embedding graph module is the graph matching networks. The only difference to the embedding model is in the message passing step, where each node not only gets messages from within the same graph, but also gets cross-graph messages by attending to all the nodes in the other graph.

Within-graph messages as before are defined as :

$$m_{i\to j} = f_{\text{message}}(h_i^{(t)}, h_j^{(t)}, e_{ij}).$$

Each node in one graph attends to all the other nodes in the other graph. The cross graph attention weight (node i in one graph attending to node j in the other graph, and vice versa) is computed as

$$\begin{aligned} a_{i\to j} &= \frac{\exp(s(h_i^{(t)},h_j^{(t)}))}{\sum_j \exp(s(h_i^{(t)},h_j^{(t)}))} \\ a_{j\to i} &= \frac{\exp(s(h_i^{(t)},h_j^{(t)}))}{\sum_i \exp(s(h_i^{(t)},h_j^{(t)}))}, \end{aligned}$$

where $s(.,.)$ is again a vector space similarity function, like Euclidean, dot-product or cosine.

6.3.2 Graph Edit Distance Task

One of the tasks to evaluate the effectiveness of the above algorithm is the task of computing edit distance (Li et al., 2019). A graph similarity or distance model that aligns with the GED is trained, by giving graphs with small edit distance a high similarity score or low learned distance, and otherwise low similarity score or high learned distance.

For training, random pairs of graphs are generated using GraphSimilarityDataset (Li et al., 2019). Given a dataset of pairs (G_1, G_2) and labels $t \in \{-1, 1\}$, the following margin-based loss is used for minimising the loss if using Euclidean distance is used as the similarity function:

$$L_{\text{pair}} = \mathbb{E}_{(G_1,G_2,t)}[\max\{0, \gamma - t(1 - d(G_1, G_2))\}]$$

This loss encourages similar graphs to have distance smaller than $1 - \gamma$, and dissimilar graphs to have distance greater than $1 + \gamma$, where γ is a margin parameter.

Here ERBP is used as the similarity function in the standard GNN and the entire set up remains the same. For the baseline, a Siamese network (Chicco, 2021) [2] and Plain GNN without ERBP are considered.

The performance of standalone GNN model, GNN with ERBP and Siamese network are compared. The dimensionality of node vectors are set to 32 and the dimensionality

[2]https://en.wikipedia.org/wiki/Siamese_neural_network

of graph vectors are set to 128. A total of 1000 pairs are used for the analysis. The results are tabulated in Table 6.9.

Model	Accuracy
Siamese network	96.09%
GNN	97.54%
GNN + ERBP	97.72%

Table 6.9 Accuracy of various models on graph edit distance task.

6.4 Observations and Remarks

ERBP has been tested on wide range of tasks like melody prediction, character and word prediction, language modeling with Transformers, abstract compositionality task and the graph edit distance task. Different ways of adding ERBP weight prior into the neural networks have been developed.

Firstly, ERBP as a relative pair wise attention prior has been applied on Transformers for music modeling tasks. It has been observed that Transformers outperform LSTM models when tested across different context lengths. Transformers with ERBP lead to further improvements when compared to Transformers in absolute and relative self attention. This has resulted in a new state of the art in monophonic pitch prediction task on the Essen Folk song collection, Bach chorales and Notthingham datasets.

Secondly, ERBP as a relational mapping prior has been applied on Transformers in Late and Mid fusion settings. The approach is similar to the fusion modeling described in Chapter 4 where ERBP weight prior mappings are added to the standard Transformers in mid (hidden layer) and late fusion (output layer) settings. The results were evaluated on WikiText-2 dataset, where it has been observed that ERBP did lead to some improvement when tested on smaller subsets of WikiText-2. However, on the entire WikiText-2 dataset, Transformers with dot product for self-attention (standard Transformers) led to best performance overall. This shows that ERBP tends to have a positive effect on smaller datasets, than on larger datasets. For larger models and datasets, the weight prior needs to be re-modeled and optimised to handle pair-wise relations effectively.

More generally, it can be seen that the effect of ERBP on Transformers in various settings like mid and late fusion doesn't lead to improvements unlike the RNN and their gated variants. One of the reasons could be the self attention mechanism in Transformers which has a way of modeling and capturing comparisonal pair-wise relationships, very similar to the ERBP approach. Hence, standard Transformers perform well without ERBP as can be seen in the language modeling experiments. Although, it looks like Transformers have a way of modeling abstract relationships in data, it is believed that the wider problem of systematicity is still something which they may not learn, which is an interesting aspect to be studied for the future work.

Further, ERBP has been applied on abstract compositional comparison task and graph neural networks. The compositional comparison task has two categories i.e. classifying sentences based on 'same type' and 'more-less type'. The baseline encoder-decoder model is compared with LSTM using ERBP for the experiments. For the 'same type' category, LSTMs with ERBP lead to best for performance outperforming the state of the art model for this task. In the 'more-less type' ERBP leads to improvement from the baseline but not better than the state of the art. It is interesting to see how Transformers like models performe in this task, which is a future scope of work.

As can be seen from Table 6.9, Graph neural network (GNN) outperforms a Siamese network in the Graph edit distance task. GNN with ERBP results in further improvements when compared to a plain GNN, even though the difference is small. Therefore, ERBP as a weight prior is an easy way of integrating into the standard networks, and as can be seen in a wide range of experiments, all the neural networks models be it sequence models like Markov models, RNN and their gated variants to Transformers to Graph neural networks, did benefit from ERBP.

In general, ERBP-like mechanisms are seen as a way of modeling inductive bias in neural network architectures addressing aspects of data efficiency, systematic learning, memorisation and the need for explicit data modeling in neural networks for solving abstract relational learning tasks. This approach of creating an inductive bias with weight priors can be extended easily to other forms of relations and will be beneficial for many other learning tasks as well.

The effect of ERBP can be studied further to assess its effectiveness on different tasks and datasets. Beyond its current form, ERBP can be adapted to other forms of abstract relations and improving generalisation on more complex tasks such as question answering and natural language understanding.

Chapter 7

Conclusions

7.1 Summary

This project started from the observation that the controversy about abstract pattern recognition posed by Marcus et al. (1999) had not been resolved. The finding that 7 month old infants learned to recognise patterns that neural networks fail to learn is intriguing, and had caused a high-profile debate over several years. One reason for controversy was that the task was not very clearly defined as a machine learning problem. Thus the first task was to define machine learning formulations of the problem in logic as classification and prediction tasks. By systematically testing the learning ability of standard neural network architectures it became clear that they do not learn generalisable solutions to this problem, although they find perfect solutions to the test sets. This raised two questions: what is the reason for this failure and how can this problem be solved.

While a mathematical answer to the first question has not yet been found, the problem can be reduced to that of learning equality relations, which are necessary to recognise the ABA and ABB patterns proposed by Marcus et al. (1999). It turns out that learning equality is a surprisingly difficult problem. It occurs also where the problem of coverage of the input neurons, that was pointed out by Marcus (2001), does not apply and even when 95% of all possible equal pairs are covered in the training set.

The experiments performed have shown that equality relations and abstract patterns based on equality/identity relations are not generalised by standard feed-forward and recurrent neural network architectures when tested on various settings. It was found that this failure to learn simple grammatical patterns is independent of various parameters like the vector dimension, amount of training data, train/test split, type of activation function, data representation, vector coverage, and aspects of the network architecture.

The Relation Based Patterns (RBP) approach, that was proposed as a first solution to the abstract pattern and equality problems, is designed to enable the learning of equality relations by introducing comparator neurons with fixed weights to and the absolute value as activation function (DR units) to encourage equality learning. Experiments show that with RBP equality is learnt with close to perfect generalisation. The RBP approach is easy to implement with a library like PyTorch, but it deviates somewhat from a standard NN architecture, as it uses the absolute value as activation and fixed weights, both of which are non-standard. Therefore, Embedded RBP (ERBP) was introduced, which uses the popular rectified linear function and employs two ReLUs instead of one DR unit. ERBP is based on RBP, but modelled as a Bayesian prior on network weights and implemented as a regularisation term in otherwise standard network learning. ERBP is very easy to integrate into standard neural networks and does not affect their learning capacity. In the experiments, the ERBP prior leads to almost perfect generalisation when learning abstract patterns from synthetic data on a wide range of tasks, beating RBP approach in most of the cases. A possible drawback of (E)RBP is that the preference for learning equality relationships may impede other learning tasks. Although this may be true to some extent, the results in this work show that this is not the case for ‘concrete’ vs ‘abstract’ patterns, and we have not observed any adverse effects in other experiments.

Both RBP and ERBP have been applied on several of real-word tasks, namely melodic pitch and textual character and word prediction, i.e. language modelling. The application of (E)RBP has shown improvement in melody prediction task when tested with a range of models for RNNs, GRUs to LSTMs. A similar trend was observed on character and word prediction experiments on natural language data. Integrating ERBP into Transformers did yield a small improvement on melodies, but no improvement in word prediction.

The effect of ERBP has also been tested on abstract compositional comparison task and graph edit distance estimation, where the aim was to learn distance based relationships in data. ERBP improves consistently over RBP and standard networks in both tasks, albeit by a small margin.

7.2 Discussion and Reflections

Equality or identity based relations are an important characteristic that defines abstract relations in data e.g., the grammar-like rules ABA or ABB rules this work started with. It is still not fully understood why the standard neural network models in their current form are not able to learn equality or abstract rules based on equality. This problem of abstract pattern learning did not start the debate on systematicity in neural network learning, which has been going on for over 30 years, but it played an important role. The lack of learning basic structures was often presented as evidence that neural networks cannot learn systematically more complex tasks especially in real-world data. The modelling of relational equality with (E)RBP enables abstract pattern learning in standard neural networks as can be seen from the various experiments performed. The consistent improvement in performance in a wide range of abstract pattern learning tasks ranging from grammatical rules, musical patterns, natural language, relational reasoning, graph edit distance and abstract compositionality provides evidence that modelling lower-level equality based relations in data does benefit and lead to improved performance in higher level relational learning tasks. To our knowledge, this link has not been shown experimentally.

Many real-world applications of rules depends on the ability to recognise identity or similarity according to some criterion, and if that can not be learned for new items, then the generalisation and applicability of the learning is severely limited. The experiments on tasks like language and music (word/note prediction, natural language inference) show that there are improvements when biases are added to the networks. This is the evidence, that this is not just a theoretical problem, but that addressing the structures underlying these problems does address fundamental and relevant problems in neural network learning.

This line of research work has raised some questions. The results on music language modelling, shows that adding structural weight priors does benefit in melody prediction, which means that even recurrent neural networks in their current form lack a mechanism to learn and generalise abstract repetition patterns in music melodies. RNNs including their gated variants GRUs and LSTMs give to rise to better performances with the added inductive biases, unlike Transformers where we see that the effect is minimal. Transformers did benefit from the priors by a very small amount for melodies, but results on natural language show very slightly worse results than standard transformers. A possible explanation for this is that the attention mechanism using the scalar product in transformers already serves a similar purpose as (E)RBP, namely comparing vectors that represent items within a given context. Although (E)RBP works slightly better with melodies than the scalar product, it seems slightly worse suited for word vectors.

Another interesting question is about the ways of adding DR units to the neural networks. Abstract patterns are generalised perfectly with RBP in *mid fusion*, where having DR_p neurons in the hidden layer is fully effective, but while having them in the input layer in *early fusion* is not. This holds even true with deeper network architectures, which shows that the distance between the DR units and the output units is not the reason.

The RBP approach also solves learning for other numerical tasks that are difficult to generalise, like digit reversal, sum of digits and it helps with parity, although this is not fully solved. It is observed that increasing the amount of training data, leads to some improvement especially significant in the case RBP *mid fusion*, but doesn't lead to perfect generalisation for the digital reversal and parity task. One possible reason for that could be that in RBP we only have a pairs of corresponding vectors elements connected in a DR_n unit. In experiment where each element in one vector is connected to each element in the other vector, parity was learned and generalised perfectly. However, this gain comes at the expense of $O(n^2)$ DR_n units. The problem of network size complexity is also relevant to models like Transformers where we still make $O(n^2)$ pair-wise comparisons and this has a problem when we try to model long-term dependencies in data. There are solutions like Draguns et al. (2020) proposed to achieve efficient sequence processing across longer sequences that could also be adapted to (E)RBP.

The most general question that remains after the work in this PhD stems from the fact that the improvements in performance after adding the priors is consistent but relatively small. This is an indication that modelling equality and comparisons is relevant, but not sufficient to reach substantial improvements in learning that we would expect from full systematicity in neural network learning. The question is: what other aspects need to be modelled and what kind of inductive bias would be required to model them? There are many possible aspects, such as hierarchical processing, logical reasoning, or just relations other than equality that are not modelled and it seems they are attracting increasing interest in current research.

7.3 Future Work

In general, mechanisms like RBP and ERBP can be seen as a way of modelling inductive bias in neural network architectures to improve data efficiency and systematic learning. This approach of creating an inductive bias can be extended to other forms of relational learning tasks, to address more general systematicity and compositionality in neural network architectures. For example, Tanneberg et al. (2020) used the differential rectifier units proposed as part of RBP approach (Weyde and Kopparti, 2018) to develop an architecture for learning efficient strategies that solve abstract algorithmic problems.

There are many other aspects and layers of systematicity that need to be addressed for achieving better systematic and compositional generalisation in neural network architectures. The work in Lake and Baroni (2018) has defined a test of systematicity and compositionality in a framework of translation, applied it to standard *seq2seq* neural network models Sutskever et al. (2014). It would be interesting to see how (E)RBP-like techniques perform on their compositional generalisation tasks. In the context of relational reasoning and natural language understanding there are works like Sinha et al. (2019, 2020) evaluating systematic out of order logical generalization in graph neural networks on CLUTRR dataset which can be experimented with (E)RBP like methods.

More generally, how can we make (E)RBP learn hierarchical structures in data, what kind of structural changes are needed for modelling the different forms of grammar in language is a very interesting future scope of work. The adapation of the existing methods towards improving systematic generalisation on more complex tasks like question answering or perception-based reasoning combined with memory models forms the future scope of work, thus approaching a more comprehensive human-like learning paradigm in neural network models.

References

Raquel G. Alhama and Willem Zuidema. Pre-wiring and pre-training: What does a neural network need to learn truly general identity rules. *CoCo at NIPS*, 2016.

Raquel G. Alhama and Willem Zuidema. A review of computational models of basic rule learning: The neural-symbolic debate and beyond. *Psychonomic Bulletin & Review*, 26(4):1174–1194, Aug 2019a. ISSN 1531-5320. doi: 10.3758/s13423-019-01602-z. URL https://doi.org/10.3758/s13423-019-01602-z.

Raquel G. Alhama and Willem H. Zuidema. A review of computational models of basic rule learning: The neural-symbolic debate and beyond. *Psychonomic Bulletin Review*, 26:1174 – 1194, 2019b.

Gerry Altmann. Rule learning by seven-month-old infants and neural networks. *Science*, 284:875–875, 05 1999. doi: 10.1126/science.284.5416.875a.

Gerry Altmann and Zoltan Dienes. Rule learning by seven-month-old infants and neural networks. *In Science*, 284(5416):875–875, 1999.

Jacob Andreas. Measuring compositionality in representation learning. *ArXiv*, abs/1902.07181, 2019.

Andrei Atanov, Arsenii Ashukha, Kirill Struminsky, Dmitriy Vetrov, and Max Welling. The deep weight prior. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByGuynAct7.

C. Baffioni, F. Guerra, and L. Lalli. The theory of stochastic processes and dynamical systems as a basis for models of musical structures. *In M. Baroni and L. Callegari (Eds.), Musical Grammars and Computer Analysis*, page (pp. 317–324), 1984.

Marco Baroni. Linguistic generalization and compositionality in modern artificial neural networks. *CoRR*, abs/1904.00157, 2019. URL http://arxiv.org/abs/1904.00157.

David G. T. Barrett, Felix Hill, Adam Santoro, Ari S. Morcos, and Timothy P. Lillicrap. Measuring abstract reasoning in neural networks. *CoRR*, abs/1807.04225, 2018. URL http://arxiv.org/abs/1807.04225.

Joost Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. Jump to better conclusions: SCAN both left and right. *CoRR*, abs/1809.04640, 2018. URL http://arxiv.org/abs/1809.04640.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Tarek R. Besold, Artur S. d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro M. Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luís C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. *CoRR*, abs/1711.03902, 2017. URL http://arxiv.org/abs/1711.03902.

J. J. Bharucha. Music cognition and perceptual facilitation: A connectionist framework. *Music Perception*, pages 5(1), 1–30., 1987.

Samuel R. Bowman, Christopher Manning Angeli, Gabor Potts, and Christopher D. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence : https://doi.org/10.1016/0004-3702(91)90053-M*, 47:139–159, 1991.

David Chalmers. Why fodor and pylyshyn were wrong: The simplest refutation. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society, Cambridge, Mass*, pages 340–347, 1990.

Srikanth Cherla, Tillman Weyde, Artur Garcez, and Marcus Pearce. A distributed model for multiple viewpoint melodic prediction. *International Society for Music Information Retrieval Conference*, pages 15–20, 2013.

Srikanth Cherla, Tillman Weyde, Artur Garcez, and Son N. Tran. Hybrid long-and short-term models of folk melodies. *International Society for Music Information Retrieval Conference*, pages 584–590, 2015.

Davide Chicco. *Siamese Neural Networks: An Overview*, pages 73–94. Springer US, New York, NY, 2021. ISBN 978-1-0716-0826-5. doi: 10.1007/978-1-0716-0826-5_3. URL https://doi.org/10.1007/978-1-0716-0826-5_3.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

N. Chomsky. *Knowledge of Language: Its Nature, Origins, and Use*. Convergence (New York, N.Y.). Praeger, 1986. ISBN 9780030055522. URL https://books.google.co.uk/books?id=BzAvmwEACAAJ.

Morten H Christiansen and Nick Chater. Generalization and connectionist language learning. *Mind & Language*, 9(3):273–287, 1994.

Morten H Christiansen, Christopher M Conway, and Suzanne Curtin. A connectionist single-mechanism account of rule-like behavior in infancy. In *Proceedings of the 22nd annual conference of the cognitive science society*, pages 83–88, 2000.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL http://arxiv.org/abs/1412.3555.

Nadav Cohen and Amnon Shashua. Inductive bias of deep convolutional networks through pooling geometry. *CoRR*, abs/1605.06743, 2016a. URL http://arxiv.org/abs/1605.06743.

Nadav Cohen and Amnon Shashua. Inductive bias of deep convolutional networks through pooling geometry, 2016b.

D. Conklin. Prediction and entropy of music. *Master's dissertation, Department of Computer Science, University of Calgary, Canada.*, 1990.

D. Conklin and J. G. Cleary. Modelling and generating music using multiple viewpoints. *In Proceedings of the First Workshop on AI and Music*, page (pp. 125–137), 1988.

D. Conklin and I. H. Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, pages 24(1), 51–73, 1995.

Darrell Conklin and Ian Witten. Multiple viewpoint systems for music prediction. *J. New Music Res*, 24, 06 2003a. doi: 10.1080/09298219508570672.

Darrell Conklin and Ian Witten. Multiple viewpoint systems for music prediction. *J. New Music Res*, 24, 06 2003b. doi: 10.1080/09298219508570672.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364, 2017. URL http://arxiv.org/abs/1705.02364.

E. Coons and D. Kraehenbuehl. Information as a measure of structure in music. *Journal of Music Theory*, pages 2(2), 127–161, 1958.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Michael Cuthbert and Christopher Ariza. Music21: A toolkit for computer-aided musicology and symbolic music data. *International Society for Music Information Retrieval Conference*, pages 637–642, 2010a.

Michael Cuthbert and Christopher Ariza. Music21: A toolkit for computer-aided musicology and symbolic music data. *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, pages 637–642, 01 2010b.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel Gershman, and Noah Goodman. Evaluating compositionality in sentence embeddings. 02 2018.

Luc De Raedt and Ingo Thon. Probabilistic rule learning. *International Conference on Inductive Logic Programming*, 6489:47–58, 06 2010. doi: 10.1007/978-3-642-21295-6_9.

Luc De Raedt, R. Manhaeve, S. Dumancic, Thomas Demeester, and A. Kimmig. Neuro-symbolic = neural + logical + probabilistic. In *Proceedings of the 2019 International Workshop on Neural- Symbolic Learning and Reasoning*, page 4, 2019. URL https://sites.google.com/view/nesy2019/home.

Stanislas Dehaene, Florent Meyniel, Catherine Wacongne, Liping Wang, and Christophe Pallier. The neural representation of sequences: from transition probabilities to algebraic patterns and linguistic trees. *Neuron*, 88(1):2–19, 2015.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings, 2016.

Peter Ford Dominey and Franck Ramus. Neural network processing of natural language: I. sensitivity to serial, temporal and abstract structure of language in the infant. *Language and Cognitive Processes*, 15(1):87–127, 2000.

Andis Draguns, Emīls Ozoliņš, Agris Šostaks, Matīss Apinis, and Kārlis Freivalds. Residual shuffle-exchange networks for fast processing of long sequences, 2020.

Peter D. Eimas, Einar R. Siqueland, Peter Jusczyk, and James Vigorito. Speech perception in infants. *Science*, 171(3968):303–306, 1971. ISSN 00368075, 10959203. URL http://www.jstor.org/stable/1731010.

W.D. Ellis. *A Source Book of Gestalt Psychology*. The Gestalt Legacy Press seminal series. Gestalt Legacy Press, 1938. ISBN 9781892966001. URL https://books.google.co.uk/books?id=kvjXAAAAMAAJ.

Jeffrey Elman. Generalization, rules, and neural networks: A simulation of marcus et. al. *https://crl.ucsd.edu/˜elman/Papers/MVRVsimulation.html*, 1999.

Jeffrey L. Elman. Finding structure in time. *In : Cognitive Science Multidisciplinary Journal*, pages Volume 14, Issue 2, 179–211, 1990a.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990b.

Jonas Langhabel et.al. Feature discovery for sequential prediction of monophonic music. *International Society for Music Information Retrieval Conference*, pages 649–655, 2017.

Reuben Feinman and Brenden M. Lake. Learning inductive biases with simple neural networks. *CoRR*, abs/1802.02745, 2018. URL http://arxiv.org/abs/1802.02745.

Jerry Fodor and Brian P McLaughlin. Connectionism and the problem of systematicity: Why smolensky's solution doesn't work. *Cognition*, 35(2):183–204, 1990.

Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

Stefan L Frank. Getting real about systematicity. In P. Calvo and J. Symons, editors, *The architecture of cognition: Rethinking Fodor and Pylyshyn's systematicity challenge*, pages 147–164. MIT Press, 2014.

Judy A. Franklin. Recurrent neural networks and pitch representations for music tasks. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, USA*, 2004.

Artur Garcez, Tarek Besold, Luc De Raedt, Peter Földiák, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luís Lamb, Risto Miikkulainen, and Daniel Silver. Neural-symbolic learning and reasoning: Contributions and challenges. *AAAI Spring Symposium Series*, 03 2015. doi: 10.13140/2.1.1779.4243.

Michael Gasser and Eliana Colunga. Babies, variables, and connectionist networks. In *Proceedings of the 21st Annual Conference of the Cognitive Science Society*, page 794. Lawrence Erlbaum, 1999.

Samuel J Gershman and Yael Niv. Novelty and inductive generalization in human reinforcement learning. *Topics in cognitive science*, 7 3:391–415, 2015.

Xavier Glorot and Y Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9: 249–256, 01 2010.

P. C. Gordon and K. J. Holyoak. Implicit learning and generalization of the mere exposure effect. *Journal of Personality and Social Psychology, 45, 492–500.*, 1983.

Thomas L Griffiths, Nick Chater, Charles Kemp, Amy Perfors, and Joshua B Tenenbaum. Probabilistic models of cognition: Exploring representations and inductive biases. *Trends in cognitive sciences*, 14:357–64, 08 2010. doi: 10.1016/j.tics.2010.05.004.

Robert F Hadley. Systematicity in connectionist language learning. *Mind & Language*, 9(3):247–272, 1994a.

Robert F Hadley. Systematicity revisited: reply to christiansen and chater and niklasson and van gelder. *Mind & Language*, 9(4):431–444, 1994b.

Jessica B Hamrick, Kelsey R Allen, Victor Bapst, Tina Zhu, Kevin R McKee, Joshua B Tenenbaum, and Peter W Battaglia. Relational inductive bias for physical construction in humans and machines. *arXiv preprint arXiv:1806.01203*, 2018.

G. E. Hinton. Connectionist learning procedures. *In : Artificial intelligence*, 40(40.1): 185–234, 1989.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997a. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997b. doi: 10.1162/neco.1997.9.8.1735.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, and Douglas Eck. An improved relative self-attention mechanism for transformer with application to music generation. *CoRR*, abs/1809.04281, 2018. URL http://arxiv.org/abs/1809.04281.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: how do neural networks generalise?, 2019.

Robert Jacobs, Michael Jordan, Steven Nowlan, and Geoffrey Hinton. Adaptive mixture of local expert. *Neural Computation*, 3:78–88, 02 1991. doi: 10.1162/neco.1991.3.1.79.

Ken Kansky, Tom Silver, David A Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. *arXiv preprint arXiv:1706.04317*, 2017.

Bence Keresztury and Elia Bruni. Compositional properties of emergent languages in deep learning, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

B. J. Knowlton and L. R. Squire. The information acquired during artificial grammar learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 20, 79 –91.*, 1994.

B. J. Knowlton and L. R. Squire. Artificial grammar learning depends on implicit acquisition of both abstract and exemplar-specific information. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 22, 169 –181.*, 1996.

D. Kraehenbuehl and E. Coons. Information as a measure of the experience of music. *Journal of Aesthetics and Art Criticism*, pages 17(4), 510–522, 1959.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2879–2888, 2018.

Brenden M. Lake. Compositional generalization through meta sequence-to-sequence learning. *CoRR*, abs/1906.05381, 2019. URL http://arxiv.org/abs/1906.05381.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *CoRR*, abs/1604.00289, 2016. URL http://arxiv.org/abs/1604.00289.

Stefan Lattner, Maarten Grachten, and Gerhard Widmer. A predictive model for music based on learned interval representations. *CoRR*, abs/1806.08686, 2018. URL http://arxiv.org/abs/1806.08686.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941, 2015. URL http://arxiv.org/abs/1504.00941.

Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. *CoRR*, abs/1904.12787, 2019. URL http://arxiv.org/abs/1904.12787.

Adam Liska, Germán Kruszewski, and Marco Baroni. Memorize or generalize? searching for a compositional RNN in a haystack. *CoRR*, abs/1802.06467, 2018. URL http://arxiv.org/abs/1802.06467.

G. F. Marcus. The algebraic mind: Integrating connectionism and cognitive science. *Cambridge MIT Press*, 2001.

G. F. Marcus. Deep learning : a critical appraisal. *arXiv:1801.00631*, 2018.

G.F. Marcus, S Vijayan, Shoba Bandi-Rao, and Peter Vishton. Rule learning by seven-month-old infants. *Science (New York, N.Y.)*, 283:77–80, 02 1999.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *CoRR*, abs/1609.07843, 2016. URL http://arxiv.org/abs/1609.07843.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michaël Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *CoRR*, abs/1412.7753, 2015.

Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Adversarial sets for regularising neural link predictors. *CoRR*, abs/1707.07596, 2017. URL http://arxiv.org/abs/1707.07596.

Pasquale Minervini, Matko Bošnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. Differentiable reasoning on large knowledge bases and natural language, 2019.

Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. Learning reasoning strategies in end-to-end differentiable proving, 2020.

Jeff Mitchell, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Extrapolation in NLP. *arXiv:1805.06648*, 2018.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *CoRR*, abs/1412.6614, 2014. URL http://arxiv.org/abs/1412.6614.

L Niklasson and Tim van Gelder. Systematicity and connectionist language learning. *Mind and Language*, 9(3):28–302, 1994.

Maxwell Nye and Andrew Saxe. Are efficient deep representations learnable? In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=B1HI4FyvM.

Maxwell I. Nye, Armando Solar-Lezama, Joshua B. Tenenbaum, and Brenden M. Lake. Learning compositional rules via neural program synthesis, 2020.

Marius Pachitariu and Maneesh Sahani. Regularization and nonlinearities for neural language models: when are they needed? *CoRR*, abs/1301.5650, 2013.

Jean-Francois Paiement, Samy Bengio, and Douglas Eck. Probabilistic models for melodic prediction. *Artificial Intelligence*, 173, 09 2009a. doi: 10.1016/j.artint.2009.06.001.

Jean-Francois Paiement, Yves Grandvalet, and Samy Bengio. Predictive models for music. *Connection Science*, 21:253–272, 06 2009b. doi: 10.1080/09540090902733806.

Marcus Pearce. *The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition.* PhD thesis, School of Informatics, City University, London, 2005.

Marcus Pearce and Geraint Wiggins. Improved methods for statistical modelling of monophonic music. *Journal of New Music Research*, 33, 12 2004. doi: 10.1080/0929821052000343840.

Marcus Pearce and Geraint Wiggins. Auditory expectation: The information dynamics of music perception and cognition. *Topics in cognitive science*, 4:625–52, 07 2012. doi: 10.1111/j.1756-8765.2012.01214.x.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Luc De Raedt, Robin Manhaeve, Sebastijan Dumancic, Thomas Demeester, and Angelika Kimmig. Neuro-symbolic = neural + logical + probabilistic. *Proceedings of the International Workshop on Neural- Symbolic Learning and Reasoning*, 2019.

Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. *CoRR*, abs/1705.11040, 2017. URL http://arxiv.org/abs/1705.11040.

Martin Rohrmeier and Patrick Rebuschat. Implicit learning and acquisition of music. *Topics in cognitive science*, 4:525–53, 10 2012. doi: 10.1111/j.1756-8765.2012.01223.x.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3856–3866, 2017.

Jenny R Saffran. Statistical learning of tone sequences by human infants and adult. *In Cognition*, 70(70.1):27–52, 1999.

H. Schaffrath. The essen folksong collection in the humdrum kern format. *Database edited by David Huron, CCAHR, Menlo Park, CA.*, 1995a. URL http://kern.ccarh.org/cgi-bin/ksbrow.

H. Schaffrath. The essen folksong collection in the humdrum kern format. *Database edited by David Huron, CCAHR, Menlo Park, CA.*, 1995b. URL http://kern.ccarh.org/cgi-bin/ksbrow.

Mark Seidenberg and Jeffrey Elman. Do infants learn grammar with algebra or statistics? *Science*, 284(5413):433–433, 1999.

Shastri and Chang. A spatiotemporal connectionist model of algebraic rule-learning. *International Computer Science Institute*, pages TR–99–011, 1999.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *CoRR*, abs/1803.02155, 2018. URL http://arxiv.org/abs/1803.02155.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1l6qiR5F7.

Thomas R. Shultz and Alan C. Bale. Neural network simulation of infant familiarization to artificial sentences: Rule-like behavior without explicit rules and variables. *Infancy, 2:4, 501-536, DOI: 10.1207/S15327078IN020407*, 2001.

Thomas R Shultz and Alan C Bale. Neural networks discover a near-identity relation to distinguish simple syntactic forms. *Minds and Machines*, 16(2):107–139, 2006.

Thomas R Shultz, Jean-Philippe Thivierge, and Debra Titone. Generalization in a model of infant sensitivity to syntactic variation. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, pages 2009–20014, 2005.

H. Siegelmann and E. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, pages Volume 50, Issue 1, pp. 132–150, 1995.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. Clutrr: A diagnostic benchmark for inductive reasoning from text, 2019.

Koustuv Sinha, Shagun Sodhani, Joelle Pineau, and William L. Hamilton. Evaluating logical generalization in graph neural networks, 2020.

Paul Smolensky. The constituent structure of connectionist mental states: A reply to fodor and pylyshyn. *Southern Journal of Philosophy*, 26(Supplement):137–161, 1987.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

David Sussillo. Random walks: Training very deep nonlinear feed-forward networks with smart initialization. *CoRR*, abs/1412.6558, 2014. URL http://arxiv.org/abs/1412.6558.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. URL http://arxiv.org/abs/1312.6199.

Daniel Tanneberg, Elmar Rueckert, and Jan Peters. Learning algorithmic solutions to symbolic planning tasks with a neural computer, 2020. URL https://openreview.net/forum?id=S1gNc3NtvB.

Andrew Trask, Felix Hill, Scott E Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. In *Advances in Neural Information Processing Systems*, pages 8046–8055, 2018.

Ivan Vankov and Jeffrey S. Bowers. Out-of-the box neural networks can support combinatorial generalization. *CoRR*, abs/1903.12354, 2019. URL http://arxiv.org/abs/1903.12354.

Ivan I. Vankov and Jeffrey S. Bowers. Training neural networks to encode symbols enables combinatorial generalization. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 375(1791):20190309, 2020. doi: 10.1098/rstb.2019.0309. URL https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2019.0309.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Marius Vilcu and Robert F Hadley. Generalization in simple recurrent networks. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 23:1072–1077, 2001.

Marius Vilcu and Robert F Hadley. Two apparent 'counterexamples' to marcus: A closer look. *Minds and Machines*, 15(3-4):359–382, 2005.

C. von Ehrenfels. *Über "Gestaltqualitäten"*. Reisland, 1890. URL https://books.google.co.uk/books?id=TqcGoAEACAAJ.

A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989. doi: 10.1109/29.21701.

Yaqing Wang and Quanming Yao. Few-shot learning: A survey. *CoRR*, abs/1904.05046, 2019. URL http://arxiv.org/abs/1904.05046.

Tillman Weyde and Radha Kopparti. Feed-forward neural networks need inductive bias to learn equality relations. *https://arxiv.org/abs/1812.01662*, 2018.

Peter M Williams. Bayesian regularization and pruning using a laplace prior. *Neural computation*, 7(1):117–143, 1995.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge, 2018.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter Battaglia. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HkxaFoC9KQ.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014. URL http://arxiv.org/abs/1409.2329.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018. URL http://arxiv.org/abs/1812.08434.