UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS MATEMÁTICAS E DA NATUREZA
INSTITUTO DE GEOCIÊNCIAS
DEPARTAMENTO DE METEOROLOGIA

DANIEL PINHEIRO ORLANDI

APLICAÇÃO DE DEEP LEARNING PARA AUXÍLIO NA PREVISÃO DE TEMPO DE CURTO PRAZO

RIO DE JANEIRO
2018

DANIEL PINHEIRO ORLANDI

APLICAÇÃO DE DEEP LEARNING PARA AUXÍLIO NA PREVISÃO DE TEMPO DE CURTO PRAZO

Monografia apresentada como requisito para a obtenção do grau de Bacharel em Meteorologia, pela Universidade Federal do Rio de Janeiro.

Orientadora: Profª. Drª. Fernanda Cerqueira Vasconcellos
Co-orientador: Me. Eduardo Charles Vasconcellos

RIO DE JANEIRO

2018

APLICAÇÃO DE DEEP LEARNING PARA AUXILIO NA PREVISÃO DE TEMPO DE CURTO PRAZO

Daniel Pinheiro Orlandi

Monografia submetida ao corpo docente do Departamento de Meteorologia da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários à obtenção de grau de Bacharel em Meteorologia.

Avaliada por:

_____

Prof.º D. Sc. Fernanda Cerqueira Vasconcellos - IGEO/UFRJ
(Orientador)

_____

M. Sc. Eduardo Charles Vasconcellos- Universidade Federal Fluminense
(Orientador)

_____

Prof.º D. Sc. Wallace Figueiredo Menezes – IGEO/UFRJ

_____

Prof.º D. Sc. Esteban Walter Gonzales Clua – Intituto de Computação/UFF

Rio de Janeiro, RJ
Fevereiro, 2018

AGRADECIMENTOS

Agradeço primeiramente à minha mãe, pois sem todo o tempo e esforço em mim investido por ela, não estaria aqui. Agradeço ao meu pai, por sempre me apoiar. Agradeço ao meu outro pai, Marcelo, que também me educou e apoiou como se eu fosse seu filho. Agradeço também à Katri, meu amor, minha vida, pela paciência nos momentos difíceis e pelo apoio e carinho, nas horas certas. Agradeço à minha família, pois sem eles, não seria o que sou hoje. Agradeço também aos amigos, irmãos que a vida me deu e que tornamram esses anos de graduação inesquecíveis. Agradeço à Universidade Federal do Rio de Janeiro, por muito tempo meu segundo lar, e que mesmo com poucos recursos se destaca como uma instituição de exclência em ensino e pesquisa. Agradeço aos meus orientadores, por terem me cobrado quando era preciso e por terem acreditado que eu era capaz. Agradeço à banca avaliadora, por aceitar avaliar o meu trabalho E por último, agradeço ao programa PIBIC, por ter subsidiado a pesquisa que deu origem a esse trabalho.

SUMÁRIO

LISTA DE SIGLAS E ABREVIATURAS

ADAM - *Adaptative Moment Estimation*

AM - Aprendizado de Máquina

BASH - *Bourne-Again Shell*

BPTT - Propagação Retrógrada no Tempo, do inglês *Back Propagation Through Time*

CPTEC - Centro de Previsão do Tempo e Estudos Climáticos

CPU - Unidade de Processamento Central, do inglês *Central Processing Unity*

DBN - *Deep Belief Network*

DL - Aprendizado Profundo, do inglês *Deep Learning*

DSA - Divisão de Satélites Ambientais

ENIAC - *Eletronic Numerical Integrator*

EQM - Erro Quadrático Médio

GPCC - *Global Precipitation Climatology Centre*

GPU - *Graphics Processing Unit*

GrADS - *Grid Analysis and Display System*

GTS - *Global Telecommunication System*

IAF - Índice de Alarme Falso

IR - Canal Infravermelho*, do inglês Infrared*

ISC - Índice de Sucesso Crítico

LSTM - *Long Short-Term Memory*

MNT - Modelagem Numérica de Tempo

MO - Canal de Micro-Ondas

MSE - *Mean Squared Error* (vide EQM)

POD - Probabilidade de Detecção

PVI - Problema de Valor Inicial

RBM - *Restricted Bolstzmann Machine*

RNA - Rede Neural Artificial

RNN - Rede Neural Recorrente, do inglês *Recurrent Neural Network*

RNP - Rede Neural Profunda

ROVER - Real-*time Optical flow by Variational methods for Echoes of Radar*

SSIM - Índice de Similaridade Estrutural, do inglês Structural Similarity Index

SWIRLS - *Short-range Warning of Intense Rainstorms in Localized Systems*

TB - Temperatura de Brilho

VIS - Canal Visível

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

APLICAÇÃO DE DEEP LEARNING PARA AUXÍLIO NA PREVISÃO DE CURTO PRAZO

Daniel Pinheiro Orlandi

Fevereiro/2018

Orientadores: Profª. Drª Fernanda Cerqueira Vasconcellos

Me. Eduardo Charles Vasconcellos

Eventos de chuva intensa são caracterizados por um alto valor na relação volume de precipitação por tempo e sua previsibilidade é de grande importância para sociedade. Tais eventos podem causar danos a propriedades, impactos econômicos e até mesmo perda de vidas. Os sistemas atmosféricos responsáveis por causar precipitação intensa possuem baixa previsibilidade, devido as suas escalas temporal (minutos a horas) e espacial (centenas a milhares de metros). Na área computacional, a previsão do tempo é dominada por modelos numéricos de previsão do tempo, e diversas estratégias têm sido adotadas para melhorar a sua capacidade de previsão. Contudo, os métodos tradicionais de previsão apresentam dificuldades ao prever esses eventos. Por isso, nesta monografia, aplicamos o *Deep Learning* como método alternativo na previsão de curto prazo. Utilizamos imagens de satélite, disponíveis na página do DSA/CPTEC, para treinar um modelo a partir da rede neural profunda PredNet e realizar previsões para 15, 30, 45, 60 e 75 minutos no futuro. Ao todo, utilizamos 108903 imagens, do canal infravermelho, do satélite METEOSAT-7, divididas em 3 conjuntos: treinamento, teste e validação. A fim de avaliar os resultados, utilizamos o Erro Quadrático Médio (EQM) e o Índice de Similaridade Estrutural (SSIM) para comparar as imagens geradas pelo modelo à imagem de referência. O modelo treinado com as imagens de satélite obteve uma boa resposta somente para as previsões de 15 minutos. Por isso, utilizou-se um outro modelo, treinado por Lotter et al.(2016) para outra tarefa, no mesmo conjunto de treinamento. O segundo modelo obteve bons resultados para todas as previsões (EQM: 0,000037 e SSIM: 0,93), sendo as previsões de 15 min as que apresentaram melhores respostas. Devido aos bons resultados apresentados e ao baixo custo, essa ferramenta mostra-se promissora para o auxílio da previsão de curto prazo.

ABSTRACT

Applying Deep Learning to improve nowcasting

Daniel Pinheiro Orlandi

February/2018

Orientadores: Prof. Dr. Fernanda Cerqueira Vasconcellos

Me Eduardo Charles Vasconcellos

Intense precipitation events are characterized by high values in precipitation volume over time ratio, and its predictability are of great importance to society. Such events can cause property damages, economical impacts, and jeopardize human lifes.The atmospheric systems responsible for causing heavy precipitation have low-predictability, due to its spacial (from meters to kilometers) and temporal scales (from minutes to hours). In computer science, weather forecasts are mostly made by numerical weather models, and many strategies have been adopted, in such way to increase model's forecast habilities. However, these models still present lack of acuracy when predicting intense precipitation events. Hence, in this research, we applied Deep Learning as alternative method to improve nowcasting. By using the satellite imagery library from DSA/CPTEC, we trained a model, using a deep neural network called PredNet to forecast 15, 30, 45, and 60 minutes in the future. Altogether, we used 108903 images divided in three sets: training, test, and validation. The results were analyzed using Mean Squared Error and Structural Similarity Index scores, to compare the image generated by the model to a ground-truth image. The model trained with the infrared satellite images was only capable of forecast, with good hability, 15 minutes in the future. Hence, we used another model, trained by Lotter et al. (2016), to execute another task, on the same training set. The second model achieved good results in all forecasts (MSE: 0,000037 e SSIM: 0,93), with the 15 minutes forecasts reaching the best results. Due to the performance achieved by the models, and the low computational cost when running the model, the Deep Learning shows to be a promising tool in improving nowcasting.

1 Introdução

Eventos de chuva intensa são caracterizados por um alto valor na relação volume de precipitação por tempo e sua previsibilidade é de grande importância para sociedade. Diversas cidades brasileiras sofrem recorrentemente com episódios de chuva intensa. Tais eventos podem causar danos a propriedades, impactos econômicos e até mesmo perda de vidas (HASSAN; BARCELLOS; DA SILVA, 2017).

Os sistemas atmosféricos responsáveis por causar precipitação intensa possuem baixa previsibilidade, devido as suas escalas espacial e temporal. A maior parte das nuvens que causam precipitação possuem um tamanho menor que a resolução espacial da maioria dos modelos atmosféricos operacionais utilizados nos centros de previsão do tempo (BOUCHER et al., 2013; CHAN et al., 2014). Além disso, uma nuvem cumulonimbus pode desenvolver-se, atingir o estágio maduro (nuvem bigorna), precipitar e se dissipar dentro de uma hora (MAPES et al., 2006). Esse tempo é inferior à saída do campo de precipitação de um modelo operacional, por exemplo, que é acumulada em horas. A bem da verdade, a saída de um modelo pode ser ajustada para um tempo menor que uma hora. Entretanto, o custo computacional e, portanto, operacional desse modelo seria inviável. Ademais, há as limitações inerentes aos próprios modelos. A atmosfera é um sistema caótico, com fenômenos que ocorrem em diversas escalas espaciais e temporais. Além de ser um sistema em cascata cujos subsistemas constituintes podem não ser corretamente simulados, ou pior, podem nunca serem conhecidos (BAUER; THORPE; BRUNET, 2015).

Na área computacional, a previsão do tempo é dominada por modelos numéricos, e diversas estratégias têm sido adotadas para melhorar a sua capacidade de previsão. Nascimento (2005) sugere que para melhorar a previsão do tempo, incluindo tempo severo, devemos: (i) conhecer melhor os ambientes atmosféricos sinóticos e de mesoescala propícios ao desenvolvimento de sistemas favoráveis à precipitação; (ii) estudar procedimentos que maximizem a extração de informações relevantes dos dados observados e de modelos de mesoescala que identifiquem esses ambientes, visando uma implementação operacional.

Este trabalho tem como motivação o item ii da estratégia proposta por Nascimento (2005) e a vasta coletânea de imagens atmosféricas de satélites disponíveis. O

objetivo geral é criar um modelo, utilizando imagens de satélite e técnicas de aprendizado de máquina (AM -*Machine Learning*, em inglês), para realizar previsão de curto prazo. Com isso, pretende-se criar uma ferramenta de *nowcasting* para auxiliar os previsores. Para alcançar esse objetivo geral, os seguintes objetivos específicos são necessários:

- Usar uma sequência de imagens de satélite para prever uma sequência fixa de imagens no futuro. Em um ambiente operacional, as imagens são obtidas a cada 15 minutos, ou menos dependendo do satélite utilizado. Nesse estudo, as imagens serão utilizadas para fazer previsões de 15, 30, 45, 60 e 75 minutos, após sua aquisição. Do ponto de vista do AM este problema pode ser tratado como uma previsão espaço-temporal de sequências (SHI et al., 2015). As imagens de satélites utilizadas são disponibilizadas gratuitamente pelo Centro de Previsão do Tempo e Estudos Climáticos (CPTEC).
- Utilizar o *DeepLearning* (DL), ou aprendizado profundo, um ramo do AM, para treinar um modelo com essas imagens de satélite. Por beneficiar-se da aceleração por *Graphics Processing Unit* (GPU - placa de vídeo), um modelo criado utilizando *DL* possui um custo computacional muito baixo. Podendo ser treinado em qualquer computador, até mesmo um computador doméstico, dotado de uma GPU (GOODFELLOW; BENGIO; COURVILLE, 2016; HEYE; VENKATESAN; CAIN, 2017). A rede utilizada para construção desse modelo será a PredNet (LOTTER; KREIMAN; COX, 2016).
- Dominar o uso das ferramentas Keras e Tensorflow, aqui utilizadas para treinar a rede neural profunda (RNP).
- Avaliar o desempenho do modelo e a acurácia da previsão.

O modelo aqui treinado, pode se tornar uma alternativa de baixo custo à implementação tradicional de um modelo numérico de previsão, capaz de prever um evento de precipitação.

Esta monografia está dividida da seguinte forma: o Capítulo 1 apresenta a motivação e objetivos deste estudo; o Capítulo 2 apresenta uma breve revisão da modelagem numérica de tempo (MNT) e seus desafios, assim como uma revisão do DL e sua aplicação na previsão da precipitação; o Capítulo 3 apresenta uma introdução aos principais conceitos e arquiteturas do DL, assim como uma descrição detalhada da rede neural utilizada; o Capítulo 4 expõe os *softwares* dados utilizados, apresenta uma descrição detalhada dos processos de aquisição, escolha e pré-processamento

dos dados, bem como uma descrição detalhada do processo de treinamento e das métricas estatísticas usadas; no Capítulo 5, os resultados obtidos são expostos; no Capítulo 6 as conclusões são apresentadas.

2 Revisão Bibliográfica

Este Capítulo apresenta o desenvolvimento, avanços e desafios encontrados na MNT ao longo das décadas, assim com uma revisão do estado da arte da aplicação de DL na previsão da precipitação.

2.1 Desenvolvimento da MNT

A previsão numérica de tempo constitui um importante avanço tecnológico e científico, pois é uma área multidisciplinar, envolvendo os campos da física, computação e química. É também produto do acúmulo de conhecimento científico resultante das inúmeras pesquisas realizadas nesse campo da Meteorologia (BAUER *et al*., 2015).

Inicialmente foi proposto que, conhecendo as leis físicas que governam a atmosfera, seria possível prever seu estado futuro (ABBE, 1901; BJERKNES; VOLKEN; BRÖNNIMANN, 1904). Foi assim que em 1922, Lewis F. Richardson ao publicar o livro intitulado "*Weather Prediction by Numerical Process* "propôs, que a previsão de mudanças na circulação atmosférica poderia ser realizada através da integração das equações de Navier-Stokes em um plano giratório (RICHARDSON, 1922;KALNAY, 2002; KIMURA, 2002; STENSRUD; YUSSOUF, 2007). Ou seja, tratando a atmosfera como um problema de valor inicial (PVI), integrando a equação da continuidade, a equação do movimento, a primeira lei da termodinâmica e assumindo o estado atual da atmosfera como condição inicial, seria possível prever seu estado futuro. Tal proposição, no início do século XX, era audaciosa, pois não havia rotinas de observações atmosféricas. Também sequer havia certeza de que a atmosfera possuía qualquer grau de previsibilidade, por conta da natureza caótica inerente aos fenômenos atmosféricos (BAUER; THORPE; BRUNET, 2015).

Utilizando em conjunto as equações anteriomente citadas, obtem-se uma poderosa ferramenta, capaz de descrever o comportamento das principais varáveis meteorológicas (KALNAY, 2002). A partir deste conjunto de equações, a variação no tempo espaço de variáveis como temperatura, pressão atmosf´erica, velocidade do vento e densidadedo ar pode ser calculada [No entanto, Richardson (1920), com seu modelo com 4 níveis na vertical e um sistema de grade horizontal de 200km, que tentava prever evoluções na pressão, velocidade do vento e densidade,

não foi capaz de fazer uma previsão para as 6h seguintes (RICHARDSON, 1922; KALNAY, 2002; KIMURA, 2002). Seus resultados obtiveram valores de pressão muito maiores que os observados na natureza, fazendo com que o modelo não fosse implementado operacionalmente à época (KIMURA, 2002). As previsões feitas por Richardson, todas à mão, e seu modelo, foram deixadas de lado. Até que, em 1950, com a ajuda do primeiro computador, o *Eletronic Numerical Integrator* (ENIAC), Charney, Fjöertoft e Von Neuman, conseguiram prever, com sucesso o campo de pressão em 500 hPa para as 24h seguintes. Os resultados obtidos, com a utilização do primeiro computador, trouxeram grandes avanços para o campo da previsão numérica de tempo (CHARNEY; FJÖRTOFT; NEUMANN, 1950; KALNAY, 2002; KIMURA, 2002; LORENZ, 2006; STENSRUD; YUSSOUF, 2007).

Atualmente, dezenas de centros de pesquisa e previsão atmosférica pelo mundo, resolvem diariamente, um sistema de equações diferenciais não lineares com meio bilhão de pontos de grade por passo de tempo. Levando em consideração processos termodinâmicos, dinâmicos, radiativos e químicos, em escalas espaciais e temporais que variam de dezenas de metros até milhares de quilômetros. Essas previsões também são utilizadas para qualidade do ar e para fins hidrológicos (BAUER; THORPE; BRUNET, 2015; CHEN et al., 2011; CLOKE et al., 2013; ZHANG et al., 2012).

2.2 Avanços na Previsão do Tempo

Os avanços científicos e tecnológicos no campo da modelagem, nos últimos anos, levaram a um aumento na capacidade de previsão dos modelos meteorológicos (BAUER; THORPE; BRUNET, 2015). Um desses avanços, são os algoritmos de assimilação de dados, que fornecem as condições iniciais para as previsões, em 4 dimensões (RABIER et al., 2000). Estas dimensões, vão da superfície até a mesosfera, em uma escala temporal que varia de minutos a horas. Os algoritmos de assimilação, também utilizam dados de estações de superfície e radiossondas, radiâncias em diversos comprimentos de onda, umidade e velocidade do vento em superfície, todos estes obtidos por satélites (GÉRARD; SAUNDERS, 1999; MCNALLY et al., 2000). Outros avanços científicos incluem: o acoplamento oceano-atmosfera (JANSSEN et al., 2002); maior resolução vertical na estratosfera e camada limite (BRETHERTON et al., 1999; SIMMONS et al., 1999); melhor

representação de nuvens e convecção (GREGORY et al., 2000); novos esquemas de radiação de onda longa (GREGORY et al., 2000; JAKOB; KLEIN, 2000), e aumento significativo na resolução horizontal (BAUER; THORPE; BRUNET, 2015; SIMMONS, 2003).

Graças a esses avanços, houve consequentemente, um aumento na acurácia das previsões. O grau de acerto das previsões de curto prazo aumentou um dia por década nos últimos 40 anos. Por exemplo, a previsão para 6 dias, tem a mesma capacidade de acerto que as previsões de 3 dias há 40 anos atrás (para a altura geopotêncial em 500hPA). O aumento na assertividade das previsões também pode ser observado para as outras variáveis (BAUER; THORPE; BRUNET, 2015; MAGNUSSON et al., 2014; SIMMONS, 2003). A previsibilidade de ambos os hemisférios aumentou nos últimos anos: 20% para o hemisfério sul e 5% para o hemisfério norte. Isso ocorre principalmente devido a utilização de dados fornecidos por satélite e técnicas de assimilação de dados mais efetivas (ENGLHISH et al., 2000; SIMMONS; HOLLINGSWORTH, 2002). É importante salientar que, na última década, os erros nas previsões de todos os modelos diminuíram e durante os últimos 30 anos, a capacidade de previsão dos modelos numéricos vem aumentando continuamente (HAIDEN et al., 2016).

2.3 Previsão da Precipitação

A precipitação é uma das variáveis meteorológicas mais difíceis de prever (ZHANG; ODINS; NIELSEN-GAMMON, 2006), sendo difícil prever sua localização, forma e intensidade. Cataldi et al. (2007), verificaram que o modelo regional Eta subestimou a precipitação, em uma previsão dez dias à frente, aplicada as bacias dos rios Iguaçu, Paraná e Paranaíba. As previsões foram realizadas semanalmente, iniciando-se as quartas-feiras às (12:00Z), para os anos de 1996 a 2001. De acordo com McBride e Eberth (2000), a localização e forma da precipitação podem ser corretamente previstas, mas a magnitude não. Em eventos extremos (>100mm/dia), a intensidade da tempestade é a maior fonte de erros (CUO; PAGANO; WANG, 2011). Contudo, a precipitação é a variável fornecida pela previsão mais utilizada pelo público em geral (LAZO; MORSS; DEMUTH, 2009). Portanto, a previsão de eventos de precipitação, continua sendo um grande desafio na meteorologia operacional (SUKOVICH et al., 2014).

Essa variável é altamente dependente de fenômenos de escala sub-grade, como por exemplo: convecção profunda e formação de gotas dentro de uma nuvem. Por esta razão, a correta previsão da precipitação é dependente de parametrizações (ARAKAWA, 2004; LOWREY; YANG, 2008; RANDALL, 2013). Essas parametrizações carregam consigo grandes fontes de erros, devido à nossa compreensão limitada dos processos físicos necessários para a formação da precipitação (LOWREY; YANG, 2008; YANO, 2016).

2.3.1 Previsão de curto-prazo (Nowcasting)

A previsão de curto prazo ou *Nowcasting* consiste de descrição das condições atuais de tempo, junto com previsões por extrapolação, para períodos de até 6h (WORLD METEOROLOGICAL ORGANIZATION, 2017). Nessa curta escala de tempo, um previsor, em posse de imagens de radar, satélite e dados observacionais, pode realizar uma previsão razoável para as próximas horas. Ele pode acompanhar a evolução dos fenômenos de pequena escala presentes em uma pequena área ou cidade. Portanto, o *Nowcasting* é uma ferramenta de extrema importância na divulgação dos impactos de sistemas atmosféricos para o público, incluindo: ciclones tropicais, tempestades e tornados, que podem causar enchentes e ventos fortes (WORLD METEOROLOGICAL ORGANIZATION, 2017).

2.3.2 Previsão de curto prazo de precipitação

A extrapolação de ecos de radar tem sido a principal forma de se realizar previsões de curto prazo de precipitação. Isso porque o início, desenvolvimento e dissipação de sistemas convectivos, relacionados a uma quantidade significativa de precipitação, estão associados à ocorrência de linhas de convergência na camada limite (WILSON, 2006). Estas linhas podem ser identificadas de 3 maneiras: nos ecos do radar, através de algoritmos automatizados que identificam a refletividade e as assinaturas Doppler das velocidades associadas às linhas de convergência; identificação utilizando seres humanos, em posse de diversos conjuntos de dados (radar, satélite estações de superfície); análise automatizada dos campos de temperatura, pressão e gradientes de vorticidade, fornecidos por modelos numéricos de previsão (WILSON, 2006).

2.4 Desafios

Apesar dos avanços das últimas décadas, ainda há barreiras a serem vencidas pela previsão numérica de tempo. Os modelos atuais, mesmo os de alta resolução espacial (1km, por exemplo), são bons em prever a precipitação gerada por sistemas frontais. Todavia, ainda possuem grande dificuldade em representar fenômenos convectivos locais ou regionais, principalmente devido à dependência que a simulação desse tipo de fenômeno tem dos processos em escala sub-grade (CERLINI; EMANUEL; TODINI, 2005; GOLDING, 2000; KAUFMANN; SCHUBIGER; BINDER, 2003; OLSON; JUNKER; KORTY, 1995; RICHARD et al., 2003). Os fenômenos que ocorrem nessa escala como: convecção profunda, turbulência de pequena escala e precipitação, são altamente dependentes das parametrizações e das condições iniciais (LOWREY; YANG, 2008), o que dificulta muito a sua previsão. (CHING et al., 2014; SUKOVICH et al., 2014).

Parametrizações descrevem processos radiativos, convectivos e difusivos que ocorrem na atmosfera e na superfície terrestre em um grande espectro de escalas e não podem ser resolvidos diretamente pelos modelos de previsão de tempo (ARAKAWA, 1997; CHING et al., 2014). As parametrizações são limitadas pelo conhecimento atual dos processos físicos de pequena e microescala, necessários para calcular o impacto médio desses processos nos fluxos de momentum e de calor atmosféricos (BAUER; THORPE; BRUNET, 2015; YANO, 2016). Apesar da pequena escala ocorrências desses fenômenos (ARAKAWA, 2004; RANDALL, 2013), se comparados à grade de modelo, sua correta simulação é essencial para uma boa previsão (ARAKAWA, 1997). Processos como a convecção profunda, por exemplo, requerem um grau maior de parametrização, pois ocorrem em pequenas porções da grade do modelo (ARAKAWA, 2004).

O grau de parametrização e o nível de representação do processo físico variam para diferentes fenômenos (STENSRUD; YUSSOUF, 2007). Portanto, seu cálculo de forma errada ou a escolha incorreta do esquema parametrizações, pode levar à grandes erros na previsão. Esses erros, não só podem afetar as simulações de fenômenos de grande escala, como também, pode afetar, principalmente, a previsão de fenômenos de menor escala (BAUER; THORPE; BRUNET, 2015; RANDALL, 2013). Com o aumento da resolução espacial dos modelos,

gradualmente menos parametrizações serão utilizadas. No entanto, processos atmosféricos como a convecção, por exemplo, acontecem em uma ampla gama de escalas. Essas escalas vão desde grandes nuvens tropicais (ocupam áreas de 15 km ou mais), até pequenas nuvens convectivas, cujos processos internos não são resolvíveis nem em escalas de 1km (BAUER; THORPE; BRUNET, 2015; RANDALL, 2013). Portanto, apesar da possibilidade de se aumentar a resolução espacial dos modelos e da constante melhora nas parametrizações ao longo dos últimos anos, sempre haverá fenômenos em escalas de movimento em que sua resolução explicita não será possível (SUKOVICH et al., 2014).

Outro ponto a ser levado em consideração é que a previsão numérica de tempo é altamente dependente de supercomputadores. Com o aumento da resolução e da complexidade dos modelos, também aumenta o investimento necessário em *hardware* e no desenvolvimento de novos algoritmos (BAUER; THORPE; BRUNET, 2015). Atualmente os computadores mais potentes utilizados na previsão do tempo figuram entre os 20 mais potentes do mundo e executam cálculos na taxa de 10^{15} operações de ponto flutuante por segundo (BAUER; THORPE; BRUNET, 2015; SHALF; QUINLAN; JANSSEN, 2011). Todavia, esses sistemas consomem grandes quantidades de energia elétrica. Assim sendo, para que os grandes centros possam manter o consumo de energia dentro de um patamar aceitável operacionalmente, faz-se necessário o investimento em novos métodos numéricos. Métodos que possuam alta paralelização computacional e baixo consumo de eletricidade (MÜLLER; SCHEICHL, 2014). As limitações operacionais enfrentadas, em face do aumento da resolução horizontal dos modelos numéricos de previsão de tempo, abrem portas para novas técnicas computacionais que podem contribuir para melhores previsões de precipitação, incluindo eventos extremos de precipitação. Uma dessas técnicas é o AM.

2.5 Aprendizado de Máquina (AM)

AM é um campo multidisciplinar, envolvendo inteligência artificial, probabilidade, estatística, teoria da informação, filosofia, psicologia e neurobiologia. Desenvolveu-se a partir do estudo de algoritmos que tentam imitar a forma como o cérebro humano aprende (SOLANKI; DHANKAR, 2017). Em 1946, com o desenvolvimento do ENIAC, surgiu a ideia de que os processos de pensar e aprender podiam ser

recriados através de um algoritmo lógico. Em 1958, Frank Rosenblatt criou o Perceptron, um programa que é composto por uma rede de pontos, onde decisões simples (em cada ponto) são utilizadas para resolver problemas complexos (ROSENBLATT, 1958). Em 1967, o reconhecimento de padrões foi desenvolvido, utilizando o método conhecido como vizinhança próxima (COVER; HART, 1967). Em 1981, Gerold Dejong introduziu o aprendizado baseado em explicação, onde conhecimento anterior é fornecido através de exemplos de treinamento, utilizando aprendizado supervisionado (DEJONG, 1981). Desde então, os algoritmos de AM continuam evoluindo e são capazes de reconhecer padrões, aprender por experiência e extrair informações relevantes de um grande volume de dados. Um dos algoritmos utilizados no AM são as Redes Neurais Artificiais (RNA). As RNAs são sistemas adaptativos não-lineares de processamento de informação, que combinam numerosas unidades de processamento. Essas unidades possuem características como auto adaptação, auto-organização e aprendizado em tempo real e são baseadas nas redes neurais biológicas (DING et al., 2013).

Os recentes avanços na capacidade das RNAs em modelar fenômenos dinâmicos não-lineares, junto com o sucesso obtido em várias outras áreas, fez com que as RNAs ganhassem notoriedade na previsão do tempo (LITTA; MARY IDICULA; NAVEEN FRANCIS, 2012).

Rivolta et al. (2006) utilizaram imagens do satélite METEOSAT-7 junto com uma RNA, que extrapolava no tempo as radiâncias, obtidas com satélite, e depois as utilizava para estimar a precipitação. Para isso, eles usaram um algoritmo calibrado com radiâncias em micro-ondas (MO), para estimar radiâncias a partir de imagens no Infravermelho (IR – *Infrared*, em inglês). Ao comparar as estimativas de precipitação com a precipitação observada, obtiveram um erro quadrático médio (EQM) de 0,4 milímetros por hora.

Além disso, Litta et al. (2012) utilizaram uma RNA, com diferentes algoritmos, para treinar um modelo que fosse capaz de prever tempestades severas sobre a região de Kolkata, Índia. Para tal tarefa, foram utilizados os campos de pressão ao nível médio do mar, velocidade do vento e umidade relativa dos meses de abril e maio de 3 anos (2007-2009). Os dados foram obtidos do Departamento Meteorológico Indiano de Kolkata. Os resultados mostraram que o modelo utilizando o Levenberg

Marquardt *algorithm* foi capaz de prever bem as tempestades, em termos da queda súbita da temperatura e intensidade.

Saba et al. (2017) utilizaram um modelo de previsão do tempo híbrido, que combina duas arquiteturas de RNA. Treinaram-no com as séries temporais das seguintes variáveis: ponto de orvalho, temperatura máxima, temperatura mínima, temperatura média, umidade relativa média, precipitação e velocidade do vento. As séries temporais destas variáveis, foram utilizadas para prever às 48h seguintes. Feito isso, os autores obtiveram um coeficiente de correlação de 0,93, EQM de 159 e um índice de espalhamento de 0,69, para as variáveis utilizadas na previsão.

2.6 Deep Learning (DL)

Apesar de obter bons resultados, as técnicas convencionais de AM são limitadas em sua habilidade de processar dados em sua forma bruta. É necessária a criação de um extrator de características que transforma os valores brutos (exemplo, os valores RGB dos pixels em uma imagem) em um vetor de características. O vetor é então passado para um classificador, que detecta e classifica os padrões nos dados (LECUN; BENGIO; HINTON, 2015). Técnicas chamadas *Representation Learning* permitem que as máquinas sejam alimentadas com dados brutos e automaticamente identifiquem as características necessárias para a classificação ou detecção de padrões (BENGIO; COURVILLE; VINCENT, 2013; LECUN; BENGIO; HINTON, 2015). O DL nada mais é do que a aplicação das técnicas de *Representation Learning* com múltiplas camadas de processamento (WAN et al., 2014). As camadas são obtidas através da combinação de vários módulos simples e não lineares (GOODFELLOW; BENGIO; COURVILLE, 2016). As RNPs são as redes neurais utilizadas quando estamos trabalhando com DL. Uma RNA é dita profunda, quando a sua densidade de neurônios (módulos) por camada é muito alta (milhares de neurônios por camada). Depois de um número suficiente de transformações nos dados, dentro da RNP, funções complexas podem ser aprendidas por um algoritmo de aprendizado (LECUN et al., 1998; LI, 2015).

O DL provou ser muito eficiente em identificar automaticamente características em dados de altas dimensões (MNIH et al., 2013). As RNPs também tem vantagens sobre as RNAs (arquiteturas com um número consideravelmente menor de camadas e de neurônios por camada), já que possuem a habilidade de descobrir

características em dados esparsos, de maneira hierarquizada, automaticamente (BENGIO; COURVILLE; VINCENT, 2013). Portanto, as técnicas de DL tem aplicabilidade em muitos domínios da ciência, incluindo a Meteorologia (KLEIN; WOLF, 2015; GHADERI; SANANDAJI; GHADERI, 2017; HOSSAIN et al., 2015).

Narejo e Pasero (2017) implementaram uma RNP capaz de fazer previsões de curto prazo para as seguintes variáveis meteorológicas: temperatura, pressão e umidade, obtidas de uma estação automática. Baseada nas arquiteturas *Deep Belief Network* (DBN) e *Restricted Bolstzmann Machine* (RBM), a RNP foi treinada comas séries temporais destas variáveis. O modelo foi capaz fazer previsões das variáveis meteorológicas com um alto grau de acerto para a região na qual a estação estava localizada. Hossain et al. (2015) compararam a eficiência de uma RNA e uma RNP na previsão da temperatura do ar no noroeste do estado americano de Nevada. Para tal, os autores utilizaram dados históricos observados de temperatura do ar e concluíram que a RNP fornece melhores previsões do que as RNAs tradicionais.

Shi et al. (2015), em sua pesquisa, utilizaram imagens de radar para treinar uma RNP e, a partir dela, previram o desenvolvimento de tempestades através das sequências de imagens. As imagens geradas pela RNP eram ecos de radar. Para quantificar a qualidade da previsão, os autores utilizaram o *Short-range Warning of Intense Rainstorms in Localized Systems* (SWIRLS), sistema de alerta utilizado pelo centro meteorológico de Hong Kong. Os autores obtiveram uma correlação de 0,908, probabilidade de detecção (POD) de 0,660, índice de alarme falso (IAF) de 0,195, índice de sucesso crítico (ISC) de 0,577 e EQM de 1,420. Todos esses índices foram melhores que os obtidos com o algoritmo Real-*time Optical flow by Variational methods for Echoes of Radar* (ROVER), parte do SWIRLS, utilizado operacionalmente pelo observatório de Hong Kong, para nowcasting. Heye et al. (2017), aplicando a mesma técnica que Shi *et al.* (2015), realizaram uma previsão de curto período de tempo, com uma resolução temporal na escala de minutos, utilizando uma RNP. A RNP foi treinada com a ajuda da aceleração de GPU, com um custo computacional muito inferior e com acurácia melhor que a de uma previsão de persistência.

3 *Deep Learning* e a Rede PredNet

Um sistema meteorológico em determinada região pode ser representado por uma grade MxN de valores P, onde P é mensurado em cada ponto da grade. Se o estado do sistema pode ser definido, em determinado instante de tempo, por um conjunto de valores P, podemos representar tal estado por um tensor X tal que $X \in R^{P \times M \times N}$, onde R representa o domínio das características observadas. Segundo Shi et al (2015), utilizando uma série temporal de observações J = $X_1, X_2 \ldots X_t$, podemos prever, com auxílio de DL, estados futuros do sistema $X_{t+1}, X_{t+2}, \ldots, X_{t+n}$.

Neste estudo, consideramos que o parâmetro P é a temperatura de brilho (TB) obtida através de imagens de satélite. Se as sequências de imagens forem separadas em subconjuntos de treinamento, validação e teste, o problema torna-se uma previsão espaço-temporal e podemos usar o DL para resolvê-lo (SHI et al., 2015).

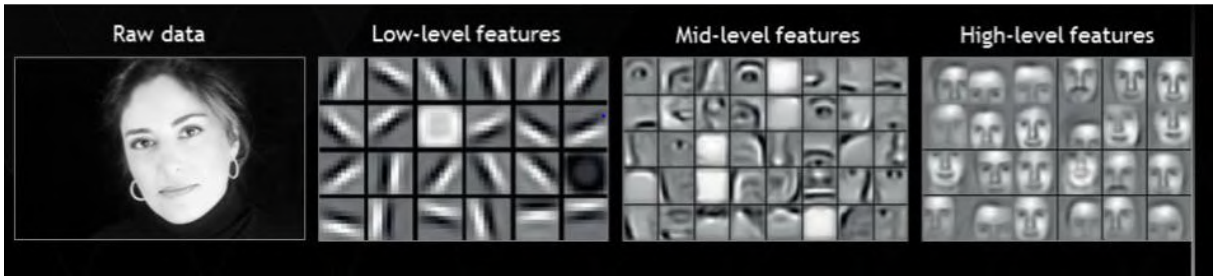
Nosso modelo de previsão será gerado a partir de uma rede chamada PredNet (LOTTER; KREIMAN; COX, 2016), que faz o uso de operações de convolução (vide Seção 3.2) aliadas a uma célula *Long Short-term Memory* (LSTM) em uma Rede Neural Recorrente (RNN, do inglês *Recurrent Neural Network*) (vide Seção 3.3), para fazer uma previsão espaço-temporal. A Rede PredNet será apresentada na Seção 3.5.

3.1 Aprendizado por representação

Os bons resultados obtidos ao empregar o DL em tarefas como classificação de imagens, reconhecimento de fala e etc, advêm do emprego do aprendizado por representação (GOODFELLOW; BENGIO; COURVILLE, 2016; LOTTER; KREIMAN; COX, 2016). Aprendizado por representação é o processo de extrair características relevantes de dados brutos e não pré tratados (BENGIO; COURVILLE; VINCENT, 2013; LECUN; BENGIO; HINTON, 2015). A imagem à esquerda na Figura 3.1 mostra um exemplo de dado bruto. Uma imagem pode ser representada por diferentes características, tais como um vetor de valores de intensidade de pixel, ou ainda, de uma forma mais abstrata, tal como um conjunto de arestas, regiões com um formato particular, etc. Uma RNP é capaz de extrair essas características e utilizá-las em tarefas como reconhecimento de expressões, reconhecimento facial, classificação de objetos, etc (BENGIO; COURVILLE; VINCENT, 2013;

GOODFELLOW; BENGIO; COURVILLE, 2016). As diferentes características extraídas de uma mesma imagem, por uma RNP, podem ser observadas nas outras imagens da Figura 3.1.

Figura 3.1: Características aprendidas por uma RNP convolutiva, em um treinamento, a partir de um dado bruto não classificado.



Fonte: https://www.analyticsvidhya.com/blog/2017/04/comparison-between-deep-learning-machine-learning/

Os métodos de DL utilizam diversas camadas de processamento, e visam a extração de características (*features*) de alto nível (Figura 3.1 – direita), a partir da aplicação do aprendizado por representação (LECUN; BENGIO; HINTON, 2015). Essas camadas são compostas de transformações (funções) lineares e não lineares, que automaticamente extraem as características relevantes para as tarefas de detecção ou classificação (GOODFELLOW; BENGIO; COURVILLE, 2016; LECUN; BENGIO; HINTON, 2015). As características extraídas ficam cada vez menos abstratas a medida que a informação flui pelas camadas de uma RNP (Figura 3.1 – centro esquerda; centro direita), possibilitando assim que uma máquina seja capaz realizar tarefas como classificação, detecção, etc (HINTON; OSINDERO; TEH, 2006).

3.2 Convolução

Extrair características de alto nível de dados brutos não é fácil (GOODFELLOW; BENGIO; COURVILLE, 2016). Redes Neurais Convolutivas tem se mostrado eficientes na extração de características a partir de dados com uma estrutura topológica. Ou seja, são capazes de manter essa estrutura conservada, mesmo após as diversas transformações que o mesmo sofre dentro de uma RNA. Segundo Dumolin e Visin (2016), imagens, sons e outros tipos similares de dados possuem essa estrutura topológica, ou seja, compartilham as seguintes propriedades: são armazenados na forma de matrizes multidimensionais (tensores); possuem um ou mais eixos cuja estrutura de organização é importante (ex: os eixos de altura e largura de uma imagem, o eixo do tempo em um dado de som); possuem canais, que são eixos utilizados para acessar diferentes características dos dados (ex: os

canais RGB de uma imagem colorida, ou os canais direito e esquerdo de um arquivo de áudio estéreo) (OH et al., 2001).

Em tarefas como visão computacional e reconhecimento de fala, a conservação da estrutura topológica presente nos dados é útil para a execução da tarefa, pois facilita o processo de extração de características (DUMOULIN; VISIN, 2016; GOODFELLOW; BENGIO; COURVILLE, 2016; LECUN; BENGIO; HINTON, 2015). Em uma operação de convolução, a RNP é capaz de preservar essas estruturas, enquanto extrai as características. Portanto, apresenta bons resultados ao executar o reconhecimento de fala ou a classificação dos elementos presentes em uma figura (DUMOULIN; VISIN, 2016; LECUN; BOTTOU; BENGIO, 1997).

Uma convolução é uma transformação linear, esparsa (apenas algumas unidades de entrada contribuem para dada saída), capaz de reutilizar seus parâmetros, ou seja, os mesmos pesos são reaplicados em outras camadas da RNP (LECUN et al., 1998). Em visão computacional, durante a operação de convolução, uma matriz, chamada de matriz de convolução ou *Kernel*, “varre” uma matriz de entrada, produzindo uma nova matriz de saída (DUMOULIN; VISIN, 2016; GOODFELLOW; BENGIO; COURVILLE, 2016). A matriz de entrada é conhecida como mapa de característica de entrada, ou *input feature map*. Que pode representar qualquer tipo de dado ou sinal, por exemplo, o canal azul de uma imagem RGB. Os valores dessa matriz seriam as intensidades dos pixels em cada ponto da imagem (DUMOULIN; VISIN, 2016; GOODFELLOW; BENGIO; COURVILLE, 2016).A matriz produzida ao final da convolução é conhecida como mapa de características de saída, ou *output feature map* e armazena as características extraídas na operação de convolução (DUMOULIN; VISIN, 2016). Informações detalhadas da aritmética da convolução podem ser encontradas no Apêndice A.

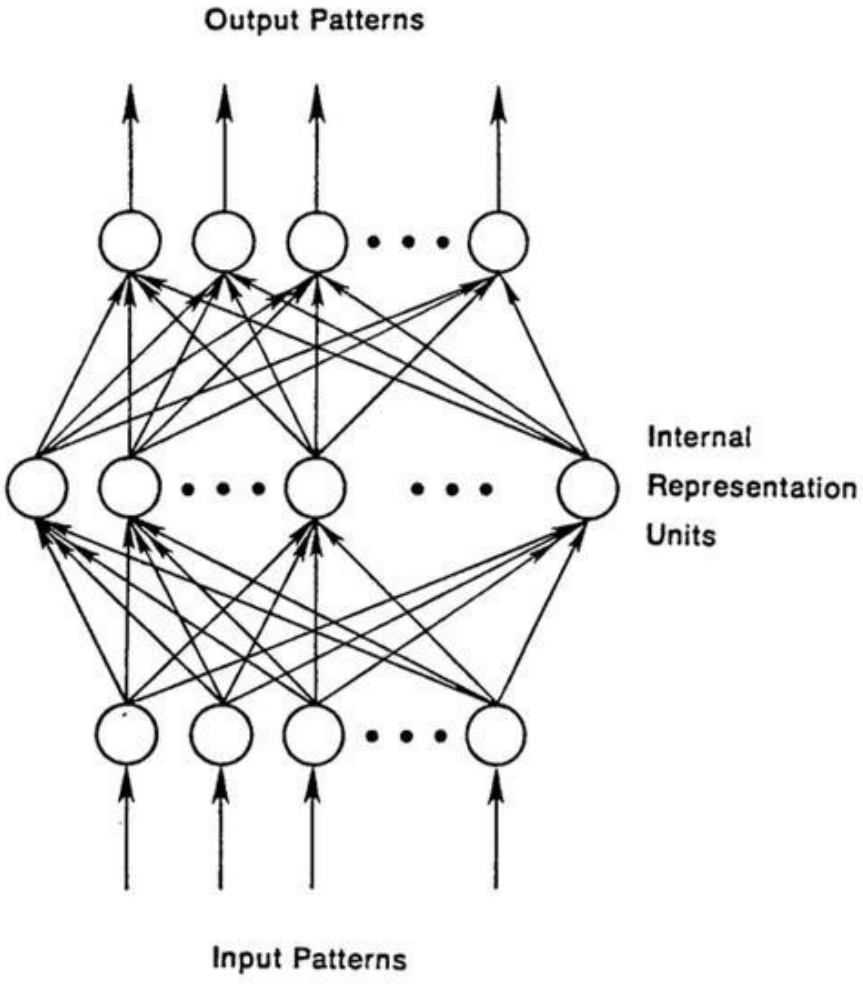
3.3 Long Short-Term Memory Networks (LSTM)

Para compreender a arquitetura LSTM é necessário dar alguns passos atrás e entender a arquitetura de rede RNN e suas limitações. Em DL, quando estamos trabalhando com dados sequenciais, ou quando queremos fazer algum tipo de previsão, a partir dos dados, utilizamos as RNNs (LECUN; BENGIO; HINTON, 2015; LOTTER; KREIMAN; COX, 2016; SHI et al., 2015). O guia intitulado“*A beginner’s guide to Recurrent Networks and LSTMs"* (DEEPLEARNING4J, 2017) contém mais

detalhes sobre RNNs, redes *feedforward* e células LSTM, ajudando na confecção dessa Seção.

RNNs são diferentes das RNPs tradicionais (redes *feedforward*). Em uma rede *feedforward* (Figura 3.2) os dados de entrada são inseridos na rede, depois convertidos em uma saída. A saída poderia ser uma classificação, um nome que é associado ao dado de entrada bruto. Por exemplo, se os dados de entrada fossem imagens de animais, a saída seria o nome desse animal (sua classificação): gato, cachorro, etc. Nesse tipo de rede, o fluxo de informações se dá em apenas um sentido (da entrada para a saída) e as informações nunca retornam ao mesmo nó mais de uma vez. A rede é treinada com uma série de dados categorizados e a forma como um dado é classificado não necessariamente altera a forma como ela classificará o próximo dado. Ou seja, uma rede *feedforward* não preserva a noção de tempo e a única entrada que ela considera, é a atual.

Figura 3.2: Típica rede feedforward.

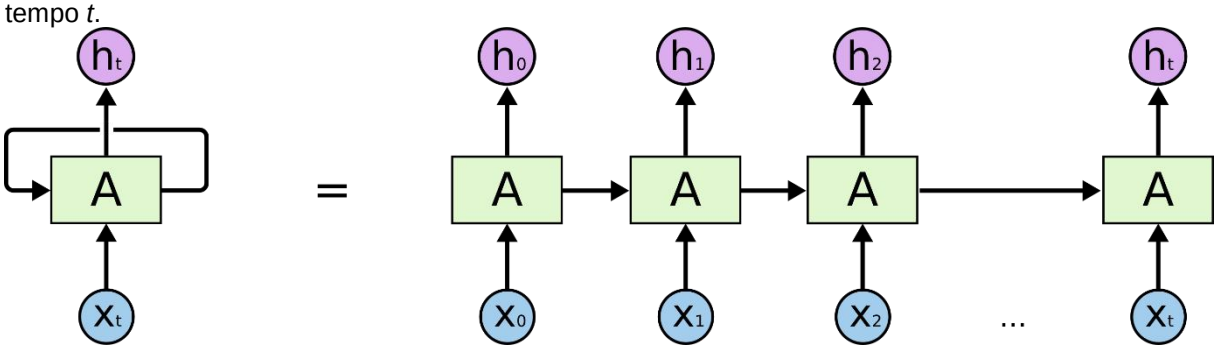


Fonte: https://deeplearning4j.org/lstm

Em uma rede *feedforward* o sinal de erro *E*, diferença entre a saída e o observado, propaga-se de cima para baixo (Figura 3.2), a partir da saída da rede, percorrendo os pesos de cada camada, até a base da rede, onde o dado de entrada é inserido. Esse mecanismo é conhecido como *backpropagation,* ou propagação retrógrada (LECUN; BENGIO; HINTON, 2015; RUMELHART; HINTON; WILLIAMS, 1986). A cada um desses pesos *W* é atribuída uma derivada parcial $\frac{-\partial E}{\partial W}$ (gradiente), que computa a contribuição de cada *W* para o erro final. Essas derivadas são então utilizadas pela regra de aprendizado, para minimizar o erro final através do ajuste dos valores atribuídos a cada *W* na rede. Esse processo é conhecido como método do declínio do gradiente ou *gradient descent* (GOODFELLOW; BENGIO; COURVILLE, 2016).

As redes RNN, ao contrário das *feedforward*, são capazes de armazenar informações por um período de tempo suficientemente longo, graças ao mecanismo de propagação retrógrada no tempo (*backpropagation through time* ou BPTT) (HOCHREITER; SCHMIDHUBER, 1997). A BPTT é uma versão adaptada da *backpropagation*, definida por uma série de cálculos ordenados que conectam um passo de tempo ao próximo (PASCANU; MIKOLOV; BENGIO, 2012; WERBOS, 1990). Isso significa que a rede não recebe como entrada apenas o dado atual, mas também o que aprendeu no passado (saídas anteriores). Ou seja, a decisão tomada pela rede no instante *t-1* afeta a decisão que a mesma irá tomar no passo de tempo seguinte *t,* e assim por diante*.* Portanto, a informação dentro da rede é reutilizada através de *loops*. Na Figura 3.3, o diagrama antes da igualdade representa a BPTT em uma RNN. Depois da igualdade (Figura 3.3 - direita), temos uma representação equivalente da RNN, onde cada elemento é uma cópia do elemento anterior. Cada um desses elementos transmite a informação para o elemento seguinte na sequência, ou seja, reutilizando as informações dos passos de tempo anteriores.

Figura 3.3: Desmembramento de uma RNN. Cada uma transmitindo a informação para seu sucessor. X_t representa o dado de entrada, *A é* a unidade oculta e h_t representa a saída em dado instante de tempo *t*.



Fonte: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Graças à capacidade que as RNNs têm de armazenar informações do passado, é dito que elas possuem “memória”. Isso tem um objetivo: há informação na sequência de dados em si. As informações contidas nos frames anteriores de um vídeo, por exemplo, ajudam a informar o estado do frame atual.

As informações a respeito das decisões tomadas anteriormente pela rede são armazenadas no que chamamos de unidades ocultas (letra A Figura 3.3), que são capazes de preservar informações de decisões tomadas muitos passos de tempo atrás, chamadas de dependências de longo termo (*long-term dependencies*). As unidades ocultas funcionam como apresentado na Equação 3.1: o estado oculto em determinado tempo *t* é A_t, que é função da entrada X_t, multiplicada por um peso *W* e depois somado ao estado oculto do passo de tempo anterior A_{t-1}, multiplicado por sua própria matriz de transição U (contém as probabilidades de transição de A_{t-1}; similar à uma cadeia de Markov). Os pesos funcionam como filtros que determinam o quanto a entrada passada é importante para a próxima saída. A operação é então submetida a uma função φ que pode ser uma função logística ou uma tangente hiperbólica, dependendo do método utilizado na aglutinação de valores muito grandes ou muito pequenos em um espaço lógico, que trunca os gradientes, tornando-os tratáveis (DEEPLEARNING4J, 2017; OLAH, 2015). O erro gerado por essas unidades será reutilizado, através da propagação retrógrada, para ajustar os pesos *W* até que o erro não diminua mais durante o processo de treinamento.

Equação3.1

$$A_t=\varphi(WX_t+UA_{t-1})$$

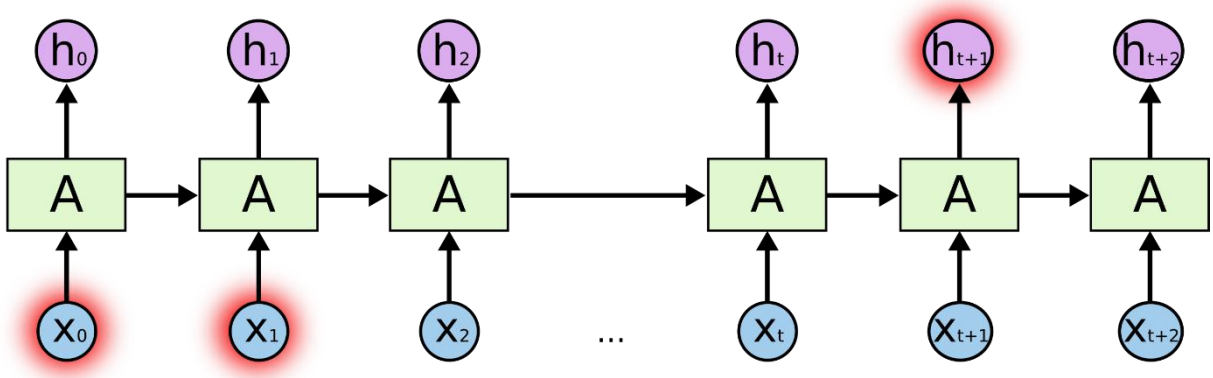
Fonte: https://deeplearning4j.org/lstm

3.3.1 Célula LSTM

A grande vantagem das RNNs é sua capacidade de usar as dependências de longo termo (DU; XU, 2017). No entanto, em alguns casos, o intervalo entre a informação relevante e o espaço em que ela é necessária é muito grande. Considere por exemplo, um modelo de linguagem tentando prever a última palavra da frase: “Eu moro na Alemanha, eu falo alemão”. A informação mais recente (verbo “falo”) sugere que a próxima palavra será o nome de um idioma, mas se quisermos prever corretamente o idioma precisaremos do contexto “Alemanha” que está mais para trás na linha temporal da frase (Figura 3.4). Em uma aplicação real, é provável que o espaço entre as informações, aumente muito, debilitando assim a capacidade da RNN de conectar as informações (OLAH, 2015).

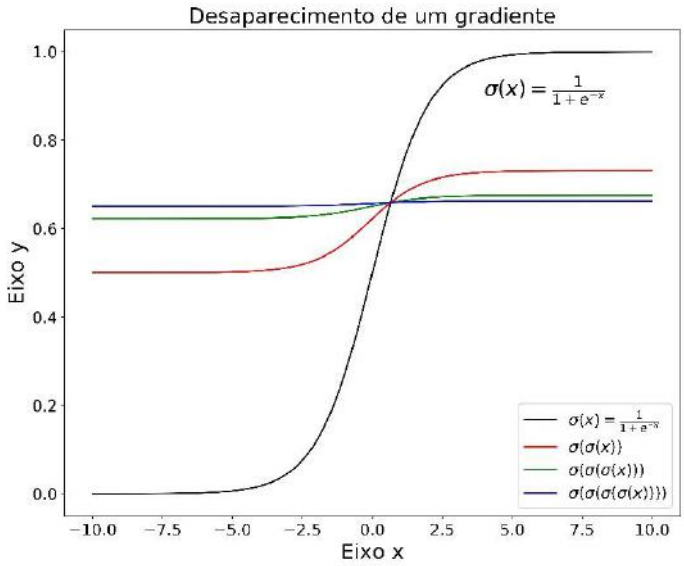
A dificuldade de um modelo de correlacionar eventos temporalmente distantes entre si, é conhecido como dissipação de gradientes ou *vanishing gradientes* (PASCANU; MIKOLOV; BENGIO, 2012). Os gradientes representam a variação dos pesos *W* em função de uma mudança no erro. Se não soubermos o comportamento do gradiente, não poderemos regular os pesos da rede, que para de aprender, precisa ajustá-los. Isso ocorre porque as camadas e os passos de tempo de uma RNN relacionam-se através de operações de multiplicação, fazendo com que sejam suscetíveis acrescimento/decrescimento explosivos. A Figura 3.5 mostra os efeitos da aplicação repetida de uma função sigmoide. Observa-se que a partir da quarta aplicação desta função (Figura 3.5 – linha em azul claro), a curva não é mais detectável, exemplificando o caso de um gradiente que desaparece.

Figura 3.4: Diagrama mostrando um intervalo longo entre as entradas necessárias, X_0 e X_1, para a previsão correta de h_{t+1}, ou seja, uma dependência de longo termo.



Fonte: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Figura 3.5: Desaparecimento de um gradiente, a partir da aplicação repetida de uma função sigmoide.



Em 1997, a Célula LSTM, foi criada por Sepp Hochreiter e Jurgen Schmidhuber para resolver o problema da explosão ou desaparecimento de gradientes nas RNNs padrão (HOCHREITER; SCHMIDHUBER, 1997). A arquitetura LSTM é capaz de preservar o erro, mantendo-o constante, que é então propagado retrogradamente para muitos passos de tempo anteriores. O que torna possível a conexão entre causas e efeitos separados por um grande número de passos de tempo.

As Células LSTMs conseguem armazenar informações fora do fluxo normal da rede recorrente, em uma célula. A informação pode ser guardada, lida ou escrita a partir de uma dessas células. Essas unidades decidem o que armazenar e quando permitir a leitura, escrita ou a eliminação das informações nelas contidas, através de portas que abrem e fecham. Cada uma dessas portas funciona através da multiplicação da informação por uma função sigmoide (também conhecida como função logística). Ao

multiplicar a informação por essa função, retém-se apenas a informação no intervalo entre zero e um (informação relevante). As portas atuam nos sinais que recebem, bloqueando (sinais fora do intervalo [0,1]) ou permitindo a passagem da informação, baseada em sua relevância (sinais dentro do intervalo [0,1]). A informação é filtrada com o conjunto de pesos contidos nas próprias células, que são ajustados através do processo de aprendizagem da rede recorrente. Portanto, a células LSTM aprendem quando deixar a informação entrar, sair ou ser deletada, através de um processo interativo de palpites, propagando retrogradamente o erro, através da BPTT e ajustando os pesos através do declínio de gradientes. Informações detalhadas sobre o funcionamento da célula LSTM e seus portões podem ser encontradas no Apêndice B.

3.4 Processo de aprendizado

O processo de aprendizado de uma rede neural nada mais é que um problema de otimização (GOODFELLOW; BENGIO; COURVILLE, 2016). Ele é implementado através do processo interativo conhecido como treinamento (DING et al., 2013). Onde um algoritmo é encarregado de ajustar os pesos em uma rede, de forma que o erro produzido, ao final de cada ciclo dentro do treinamento, seja minimizado. O processo de aprendizado das RNAs pode ser dividido em: supervisionado, não supervisionado e aprendizado por reforço. O aprendizado supervisionado se dá quando há uma comparação direta entre a saída da rede e a saída esperada (alvo), onde cada exemplo no conjunto de treinamento está associado à uma classe pré-definida (GOODFELLOW; BENGIO; COURVILLE, 2016; SOLANKI; DHANKAR, 2017). No aprendizado não supervisionado, o conjunto de treinamento não está classificado (dado bruto) e a rede aprende sozinha as características mais importantes do conjunto de dados (GOODFELLOW; BENGIO; COURVILLE, 2016; LE et al., 2011; SOLANKI; DHANKAR, 2017). O aprendizado por reforço é um caso especial do aprendizado supervisionado onde a rede aprende por tentativa e erro a maximizar seus resultados através de uma recompensa (um sinal, representando o nível de otimização necessário, que é propagado pela RNA ou RNP (MNIH et al., 2013; SOLANKI; DHANKAR, 2017). Na arquitetura PredNet, utilizamos o aprendizado não supervisionado, pois a rede aprende sozinha a localizar os sitemas meteorológicos, assim como sua movimentação.

O algoritmo responsável pela gradual redução do erro, também conhecido como otimizador, é a essência do processo de aprendizado. Geralmente, os otimizadores são baseados no método do declínio do gradiente, no entanto, existem diversos otimizadores baseados em diferentes métodos de aprendizado (DING et al., 2013). A arquitetura PredNet utiliza como otimizador o algoritmo conhecido como ADAM, uma abreviação para *Adaptive Moment Estimation*. ADAM é um algoritmo de otimização estocástica, baseado no método do gradiente, que utiliza apenas gradientes de primeira ordem[1]. Esse método é computacionalmente eficiente e requer pouca memória, tornando-o um ótimo candidato para problemas que possuam muitos parâmetros ou um conjunto de dados muito grande (KINGMA; BA, 2014; LOTTER; KREIMAN; COX, 2016).

3.5 A rede PredNet

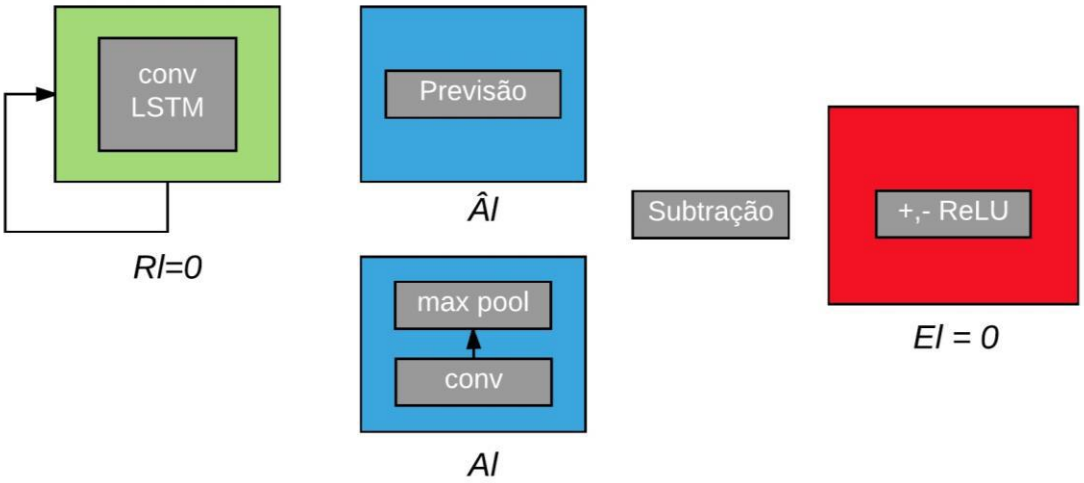
A rede PredNet consiste de uma série de módulos, que podem ser sobrepostos. Os módulos tentam fazer previsões locais das entradas (frames). O algoritmo baseia-se na seguinte ideia: para que seja possível prever a próxima imagem em uma sequência temporal de imagens, é preciso um modelo que seja capaz de representar as mudanças dinâmicas que as estruturas presentes nessa imagem sofrem ao longo do tempo. A rede PredNet tenta prever continuamente os próximos frames de um vídeo, ou sequência de imagens. Para isso, utiliza uma rede convolutiva recorrente, que nada mais é que a combinação da arquitetura convolutiva e da arquitetura RNN com células LSTM (LOTTER; KREIMAN; COX, 2016). Os desenvolvedores da rede utilizada nesse estudo inspiraram-se no conceito *predictive coding*, que tem sua origem na neurociência. O *predictive coding* atesta que o cérebro humano está constantemente fazendo previsões com base nos estímulos sensoriais que recebe (LOTTER; KREIMAN; COX, 2016).

3.5.1 Passo a passo da rede PredNet

Considerando uma rede de uma camada (Figura 3.6), no instante t=0, ainda não houve a ingestão dos dados (imagens de satélite). Portanto, o erro, $E_{l,}$ e a camada convolutiva recorrente, R_l, são inicializadas com valor zero.

[1] Gradientes de primeira ordem nada mais são do que as derivadas de primeira ordem calculadas no método do gradiente, para mais informações vide Seção 3.3.

Figura 3.6: Diagrama do processo de inicialização da rede, t=0, com $El = R_l = 0$.



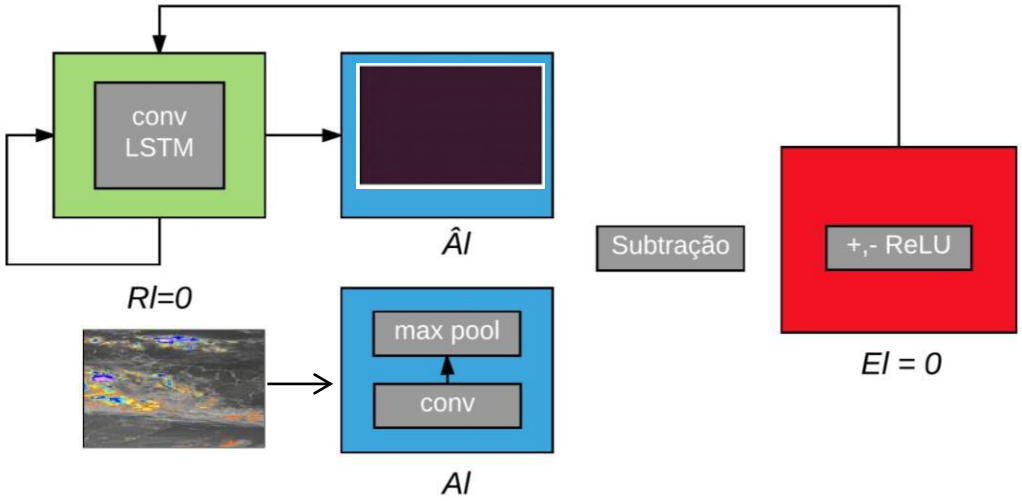
Fonte: Lotter et al.(2016)

No instante t=1, E_l é atualizado primeiro (ainda zero) e seu valor utilizado para atualizar os pesos da camada R_l, que ainda continuam a valer zero. R_l é então ativada e produz uma previsão inicial $\hat{A}_l$. O resultado da convolução $\hat{A}_l$ é um frame todo escuro, pois a camada E_l, utilizada para atualizar R_l, ainda vale zero (Figura 3.7). Depois, a primeira imagem real A_l, é inserida na rede e passa por uma convolução e um *max pooling*[2] (Figura 3.7). A_l e $\hat{A}_l$ são subtraídos e então transferidos para a camada responsável por calcular o erro (Figura 3.8). Depois de calculado o erro é dividido em duas populações retificadas (entre -1 e 1), uma positiva e a outra negativa, dando origem a E_{l+1} (Figura 3.8) (LOTTER; KREIMAN; COX, 2016).

No instante t=2, E_{l+1} é utilizado para atualizar os pesos da camada R_l. Depois de atualizada, a camada produz uma nova previsão $\hat{A}_{l+1}$. Nesse momento, a próxima imagem na sequência, A_{l+1}, é inserida na rede, subtraída de $\hat{A}_{l+1}$ e seu resultado é encaminhado para a camada E_l, que calcula E_{l+2} e assim por diante (Figura 3.9).

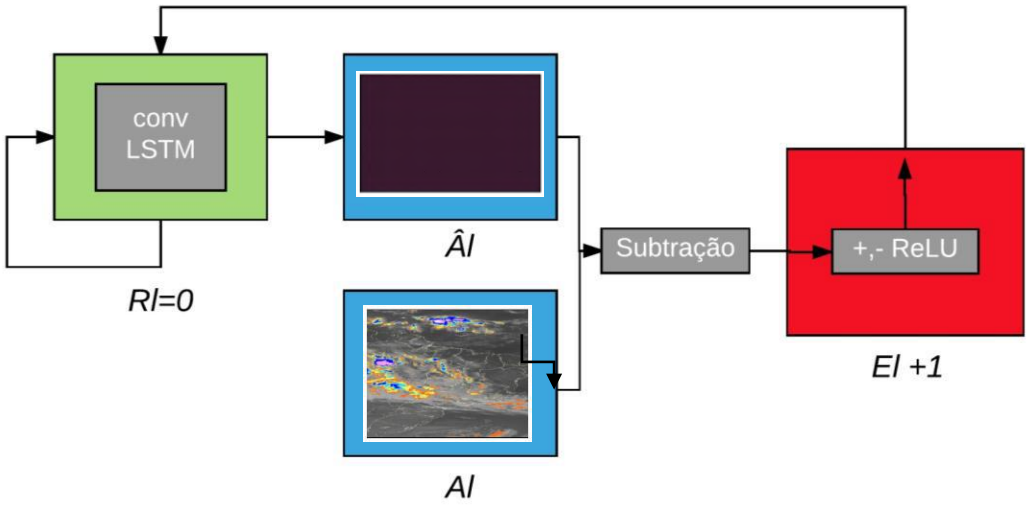
[2] O *pooling* ou agrupamento é similar a convolução. A diferença, no *pooling*, é que o *kernel* ao varrer o mapa de entrada computa uma função matemática. Essa função pode ser o valor máximo (*max pooling*), valor médio (*mean pooling*), valor mínimo (*min pooling*), etc. Informações detalhadas sobre a aritmética do pooling podem ser encontradas no Apêndice A.1.

Figura 3.7: Diagrama do processo de funcionamento da PredNet em *t=1.*



Fonte: Lotter et al.(2016)

Figura 3.8: Diagrama do fluxo de informações na rede PredNet ainda em t=1.

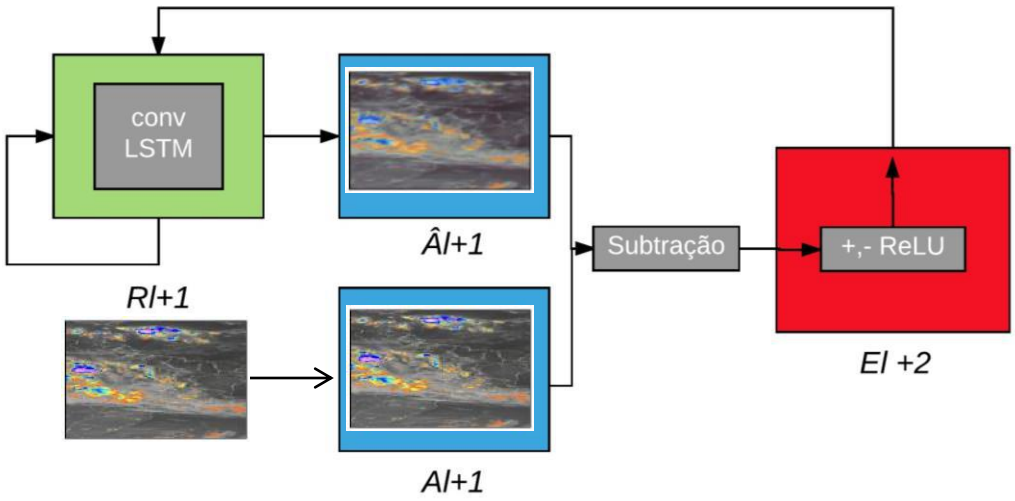


Fonte: Lotter et al.(2016)

A rede PredNet, é organizada de forma que seu “lado direito” (camadas A_l *e* E_l) é equivalente à uma rede convolutiva profunda padrão, enquanto seu “lado esquerdo” (camadas R_l)funciona como uma *Generative Adversarial Network*, ou uma rede desconvolutiva[3] (LOTTER; KREIMAN; COX, 2016). O modelo é treinado de forma que a soma ponderada dos valores de erro obtidos, seja minimizada. A Figura 3.10, mostra a diagramação completa da rede PredNet (LOTTER; KREIMAN; COX, 2016).

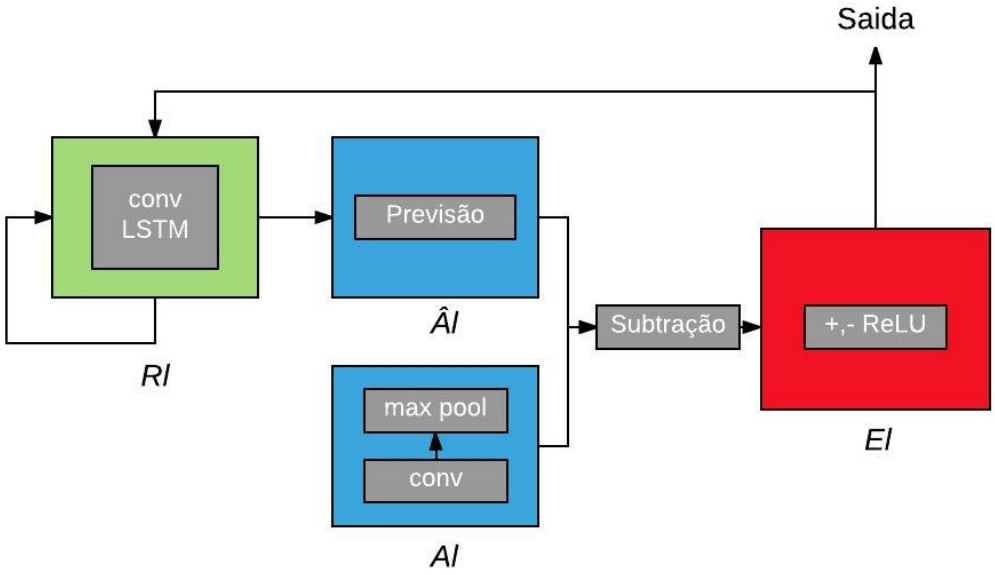
[3] Uma rede desconvolutiva realiza o processo contrário ao de uma convolução. Ou seja, recebe um mapa de características, saído de uma convolução e o transforma novamente em um sinal. No caso da PredNet transforma o mapa de características em uma imagem.

Figura 3.9: Diagrama do fluxo de informações em *t=2*. O processo se repete para t=3, t=4, etc.



Fonte: Lotter et al.(2016)

Figura 3.10: Estrutura completa da rede PredNet, contendo todos os seus módulos.



Fonte: Lotter et al.(2016)

4 Treinamento de uma RNP

O treinamento de uma RNP pode ser executado de duas formas: *Batch Training* ou *Online Training*. A principal diferença entre as duas formas de treinamento é que, no *online training*, os pesos e o viés são ajustados ao final de cada ciclo (instância). Já no *Batch Training*, os ajustes nos pesos são acumulados e só são ajustados após um determinado número de instâncias, também conhecido como épocas ou *epoch*.

Em arquiteturas que utilizam o método do gradiente é comum usar o *online training*, já que este se mostra mais eficiente e mais rápido no processo de aprendizado (WILSON; MARTINEZ, 2003). Isso ocorre porque este tipo de treinamento é capaz de usar o gradiente local, de cada instância de treinamento, para determinar qual o melhor ajuste a ser feito. Esses gradientes podem conter ruídos ou até mesmo serem contraditórios, mas na média, ajustam-se em direção ao gradiente verdadeiro. Como no *online training* os pesos são atualizados a cada instância, o método é capaz de acompanhar as variações do gradiente ao longo de cada época, o que permite uma taxa de aprendizado maior (ordem de grandeza com que o erro é atualizada) (WILSON; MARTINEZ, 2003). Nessa forma de treinamento, é necessário que o conjunto de treinamento seja dividido em partes menores, o que assegura um processo de treinamento mais eficiente computacionalmente. A arquitetura PredNet utiliza o *online training*.

O treinamento foi realizado em um *Desktop*, equipado com uma unidade de processamento central (CPU, do inglês *Central Processing Unit*) Intel 4790k, uma GPU NVIDIA GeForce GTX 980 Ti e 11Gb de memória RAM. A CPU conta com 4 núcleos, já a GPU, possui 2816 núcleos e utiliza a arquitetura NVIDIA Maxwell^{tm}. Os *softwares* utilizados nesse estudo utilizam a aceleração de GPU, onde a mesma executa as funções mais intensivas computacionalmente. O que faz com que o processo de treinamento seja consideravelmente mais rápido do que quando feito em CPU (HEYE; VENKATESAN; CAIN, 2017).

4.1 Produtos

Foram utilizados os seguintes produtos no estudo:

- *Global Precipitation Climatology Centre (GPCC) Full Data Daily Version 1.0 at 1.0° – Daily Land-Surface Precipitation from Rain-Gauges built on GTS-based and Historic Data*: O produto fornece uma estimativa do acumulado diário de

precipitação em uma grade com resolução espacial de 1° de latitude por 1° de longitude. O dado é baseado em dados de pluviômetros históricos e dados fornecidos pelo *Global Telecommunication System* (GTS). Além do acumulado diário de precipitação, em mm dia^{-1}, fornece também o desvio padrão em mm dia^{-1}, o erro kriging na interpolação em % e o número de medidas por célula de grade, fornecidos em um arquivo netcdf (SCHAMM et al., 2014).

- Imagens do satélite geoestacionário METEOSAT-7: As imagens no canal IR realçado foram obtidas na página do CPTEC, oriundos da Divisão de Satélites Ambientais (DSA). Possuem 1151x1323 pixels, com a latitude variando de 5°N a 35°S e longitude variando de 70°W a 20°W e resolução temporal de 15 minutos. Cada pixel corresponde a um quadrado de 5X5 km.

4.2 Softwares

Foram utilizados os seguintes *softwares* no estudo:

- GrADS: O *Grid Analysis and Display System* (GrADS) é uma ferramenta interativa que possibilita o fácil acesso, manipulação e visualização de dados em geociências. O GrADS possui dois modelos capazes de analisar tanto dados de estações, quanto dados em ponto de grade, suportando diversos formatos de dados, incluindo binário, GRIB, NetCDF, HDF 4 e 5 e BUFR (OPENGRADS, 2017).
- *Shell Script*: É uma linguagem de script usada em diversos sistemas operacionais, com diferentes versões, dependendo do interpretador de comandos utilizado. O interpretador disponível na maior parte das distribuições GNU/Linux é o *Bourne-Again Shell* (bash), nomeado assim em homenagem ao seu criador: Bourne Shell (CONTRIBUIDORES DA WIKIPÉDIA, 2017a, 2017b).
- *Python2.7*: Poderosa linguagem de programação, orientada ao objeto, comparável a Perl, Ruby, Scheme e Java. É uma linguagem de fácil aprendizado, possuindo extensas bibliotecas aplicáveis em diversos domínios da ciência e ciência da computação, além de ser de fácil implementação em

diversas plataformas (BEGINNERSGUIDE/OVERVIEW - PYTHON WIKI, 2017).

- *Google Tensorflow*: TensorFlow™ é uma biblioteca de software *open source* para cálculo numérico. A arquitetura flexível do TensorFlow permite a implementação em uma CPUs ou GPUs em um *desktop*, servidor ou dispositivo móvel com uma simples aplicação. Foi inicialmente desenvolvido para pesquisas envolvendo AM e RNPs, contudo o sistema é aplicável em uma gama problemas (ABADI et al., 2016).
- *Keras*: Keras é uma aplicação de alto nível, escrita em Python que utiliza o TensorFlow. Foi desenvolvida com o objetivo de possibilitar a experimentação rápida, sendo capaz de ir de “ideia” ao resultado com uma curva de aprendizado alta. Foi desenvolvido tendo em mente 4 princípios: ser amigável ao usuário, modularidade, fácil extensão e utiliza o Python como linguagem padrão (CHOLLET, 2016).

4.3 Imagens no IR

Um ponto importante é a escolha das imagens no canal IR. As radiâncias no IR não estão diretamente relacionadas com o processo de formação da precipitação (RIVOLTA et al., 2006). Entretanto, radiômetros nos comprimentos de onda do visível (VIS) e IR são calibrados utilizando dados provenientes de algoritmos baseados em radiômetros em Micro-ondas (MO) (DI MICHELE et al., 2005), fornecendo informações sobre a estrutura interna da nuvem. De modo que as imagens IR podem ser utilizadas no estudo da precipitação (RIVOLTA et al., 2006). Outro ponto positivo do imagens IR é a possibilidade de usar Temperaturas de Brilho (TB) mais frias para a detecção de núcleos convectivos. Estes núcleos, frios, estão relacionados à nuvens de grande desenvolvimento vertical, que por sua vez estão associadas à precipitação intensa (THOMAS; CORPETTI; MÉMIN, 2010). Um ponto negativo do uso de imagens IR é a dificuldade de detecção de sistemas mais rasos (nuvens com menor desenvolvimento vertical), pois estes sistemas estão mais próximos do solo (SHUKLA; KISHTAWAL; PAL, 2017).

Todavia, apesar da dificuldade de detecção de sistemas menos profundos, radiômetros IR são capazes de identificar, eficientemente, os sistemas convectivos de escala sinótica e de mesoescala causadores de grandes volumes de precipitação. Portanto, sendo úteis na previsão da precipitação. Por isso, as imagens

provenientes desses sensores foram utilizadas nesse estudo (DI MICHELE et al., 2005; RIVOLTA et al., 2006; SHUKLA; KISHTAWAL; PAL, 2017; THOMAS; CORPETTI; MÉMIN, 2010).

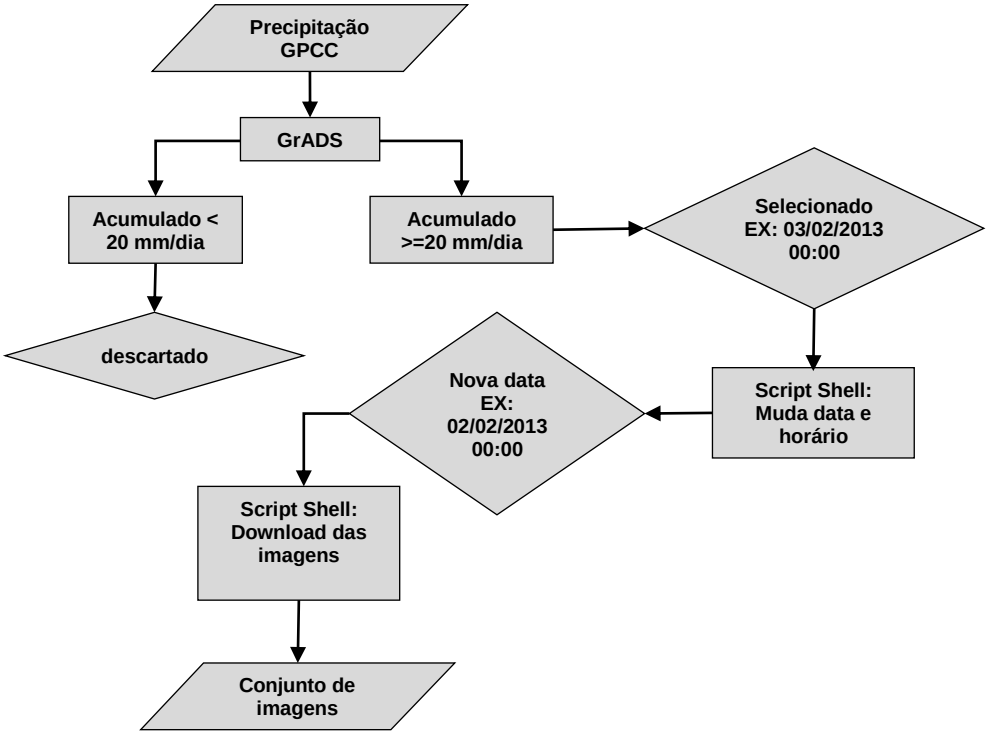
4.4 Escolha dos casos de precipitação e aquisição de imagens

Para compor o conjunto de dados, foi escolhido o período de 2011-2013 de acumulados diários de precipitação fornecidos pelo Global Precipitation Climatology Centre (GPCC). As imagens são das 24h anteriores ao horário em que a precipitação é acumulada. Ou seja, o GPCC acumula a precipitação todos os dias (00-00Z) e as imagens obtidas são das 24 horas anteriores. Por exemplo, se temos um acumulado de 50 mm no dia 03/02/2013 às 00Z, a aquisição das imagens começa no dia 02/03/2011 às 00Z e vai até às 00Z do dia 03/03/2013, totalizando 24 horas de imagens.

Os casos de precipitação foram selecionados a partir de um script criado no GrADS. O limiar escolhido foi de 20 milímetros por dia, em uma grade com latitude variando de 5°N a 35°S e longitude variando de 70°W a 20°W. Este limiar foi escolhido por dois motivos: para que fosse possível obter um número suficientemente grande de imagens, viabilizando o treinamento da rede; garantir que a rede seja capaz de aprender a dinâmica interna da atmosfera. Sendo assim, capaz de prever tanto eventos de precipitação considerados recorrentes e também eventos extremos, e consequentemente, mais raros (SHI et al., 2015).

Em posse dos referidos dias, utilizou-se um script em Shell para que fosse realizada a mudança das datas (para as 24 horas anteriores), e posterior download das imagens (IR), obtidas no endereço eletrônico do CPTEC. Ao todo foram coletadas 108903 imagens e o processo de aquisição das imagens é mostrado no fluxograma da Figura 4.1.

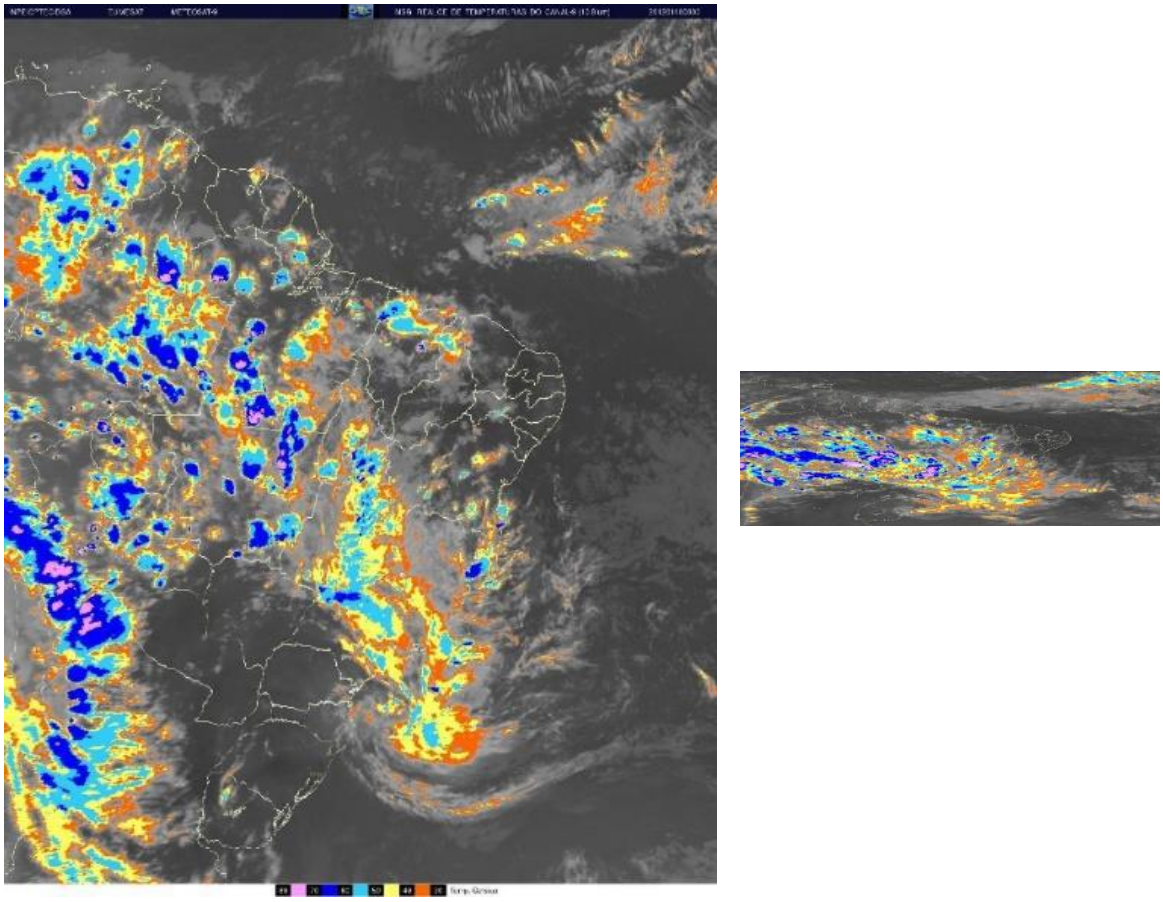
Figura 4.1: Fluxograma descrevendo o processo completo de pré-processamento dos dados.



4.5 Pré-processamento

Utilizando o Python, as imagens tiveram seus cabeçalhos e barra de cores retirados e foram redimensionadas de 1151x1323 pixels para 128x160 pixels (Figura 4.2). O redimensionamento foi necessário porque as imagens originais consumiam mais memória do que o disponível na máquina em que o treinamento foi realizado. Como todas as imagens são redimensionadas para um mesmo tamanho, a distorção sofrida por elas não prejudica o processo de aprendizado da rede. Após o corte e redimensionamento, as imagens foram divididas em 3 subconjuntos: conjunto de treinamento, conjunto de teste e conjunto de validação. A proporção da divisão pode ser vista na Tabela 4.1.

Figura 4.2: Esquerda, imagem em tamanho original (1151X1323), com cabeçalho (parte superior) e barra de cores (parte inferior da imagem). Direita, imagem redimensionada (128x160), cabeçalho e barra de cores removidos. A imagem na direita fica distorcida graças ao redimensionamento.



Fonte: CPTEC-INPE/DAS

Tabela 4.1: Total de imagens, número de dias, sequências de 15 dias, assim como a proporção de imagens contidas em cada subconjunto de dados.

Subconjunto	Número de Imagens	Número de dias	Número de blocos de 15 imagens	Proporção (%)
Treinamento:	88581	857	5485	80
Validação:	15302	161	1029	15
Teste:	5020	54	343	5
Total:	108903	1072	6857	100

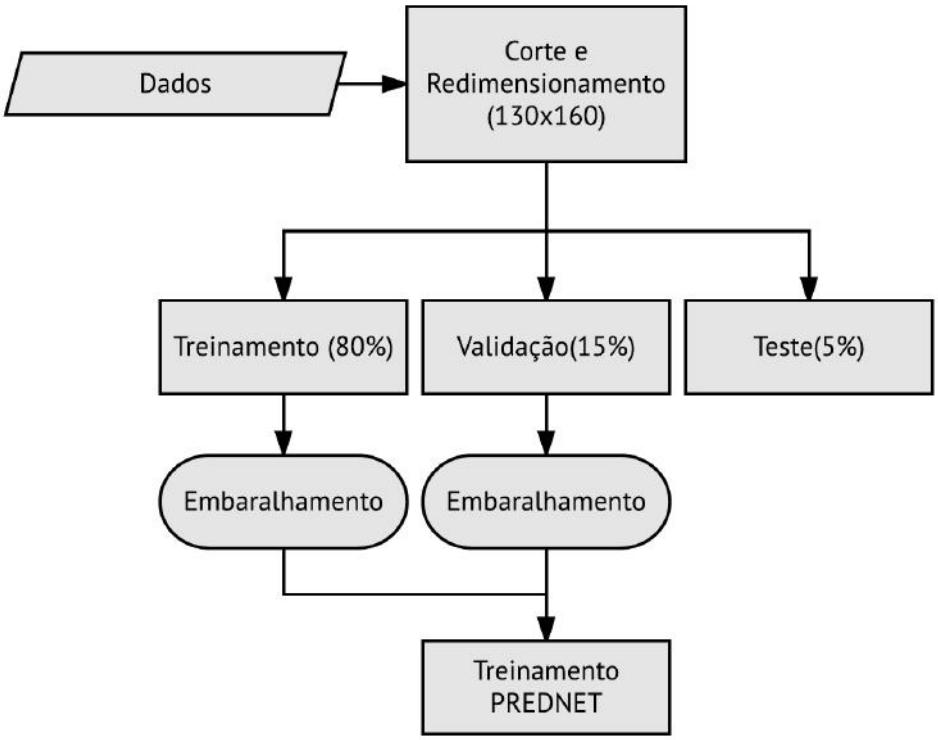
Após a divisão, as imagens precisam ser separadas em sequências de frames, que podem ter qualquer tamanho. Neste estudo, separamos as imagens em sequências de 15 frames consecutivos, de forma que no processo de treinamento tentamos prever o 10º, 11º, 12°, 13°, 14° frame, totalizando 75 minutos de previsão.

Em seguida, as sequências são submetidas a um processo de embaralhamento. O embaralhamento é feito de maneira que a próxima sequência de frames apresentada a rede para treinamento não seja consecutiva à sequência anterior. Ou seja, blocos de 15 frames não consecutivos (15 frames totalizam 225 minutos). Os

frames de cada bloco constituem uma sequência temporal, ou seja, o primeiro frame corresponde ao minuto 0, na sequência, e o décimo quinto, ao minuto 225. Essa sequência temporal nunca é desfeita durante o embaralhamento. O processo de embaralhamento cria descontinuidade nos subconjuntos de dados. As descontinuidades são importantes, pois fazem com que o modelo seja capaz de generalizar. O processo completo de pré-processamento está descrito no fluxograma da Figura 4.3.

Um modelo é dito generalista quando não se especializa em seu conjunto de treinamento (HEYE; VENKATESAN; CAIN, 2017; LOTTER; KREIMAN; COX, 2016; SHI et al., 2015), ou seja, é capaz de manter a acurácia da previsão mesmo para casos não vistos no treinamento.

Figura 4.3: Fluxograma com a metodologia completa empregada no pré-processamento das imagens. A linguagem de programação Python foi utilizada em todo o processo supracitado.



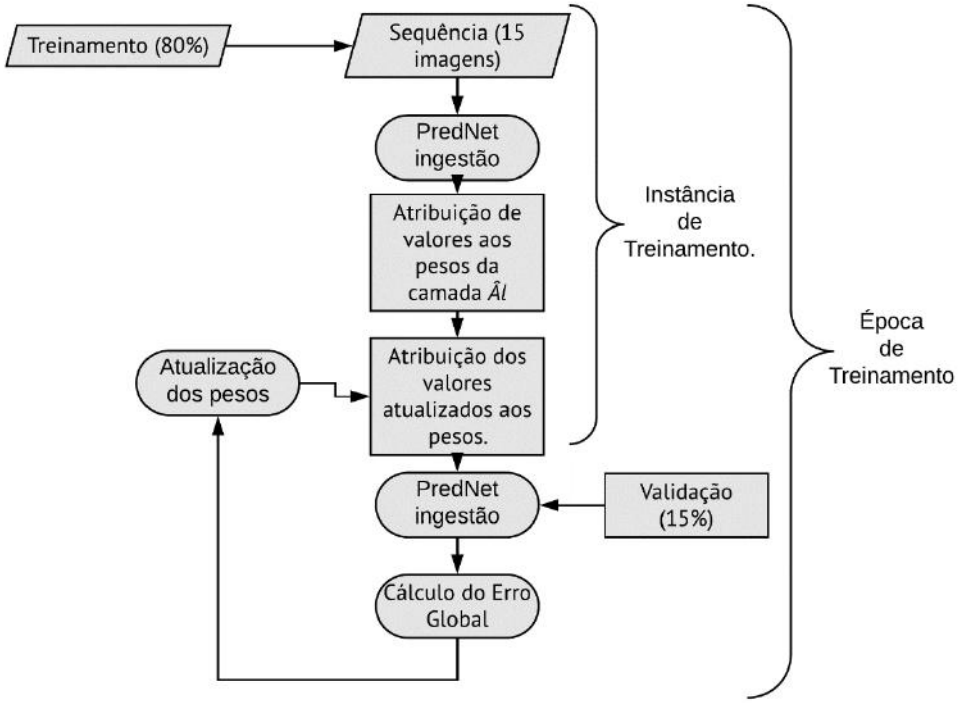
As descontinuidades introduzidas nos conjuntos de treinamento, validação e teste, através da não interseção (não há imagens repetidas em nenhum subconjunto) e do embaralhamento dos blocos de imagens, fazem com que a generalização seja alcançada durante o treinamento (LOTTER; KREIMAN; COX, 2016; SHI et al., 2015). Assim sendo, a rede “aprende” a dinâmica interna dos sistemas presentes

nos dados (SHI et al., 2015). No caso específico deste estudo, o modelo estaria aprendendo dinâmica interna da atmosfera terrestre.

4.6 Treinamento

O processo de treinamento de uma RNP como a PredNet é relativamente simples. A rede recebe como entrada 585 imagens por época de treinamento. As imagens são agrupadas em conjuntos de 195 imagens. Essa divisão é conhecida como *mini-batch*. Cada *mini-batch* possui 13 sequências de 15 *frames* ordenados, totalizando 195 imagens por *batch* e um total de 3 batches por época de treinamento. Em uma instância de treinamento, todas as sequências de *frames*, presentes em um batch (195 imagens ou 13 sequências de *frames*), percorrem todos os módulos da rede, seguindo o processo descrito na Seção 3.5.1. Após a passagem de todos os *batches* pelos módulos da rede, completa-se uma época do treinamento, ao todo 150 épocas de treinamento foram utilizadas. O erro global é calculado, interativamente, após a comparação de cada imagem gerada pela rede e a observação equivalente, durante as instâncias do treinamento. Este erro é retroprogagado e utilizado para ajustar os pesos em cada camada da RNP através do declínio do gradiente. O gradiente é gerado a partir do cálculo da taxa de variação do erro global, em relação ao valor atribuído aos pesos da RNP, em determinada época do treinamento. A partir da época 75 a taxa de aprendizado (*learning rate*) muda de 0,1 para 0,01, por instância. Isso significa que o algoritmo de otimização, ADAM, passa a diminuir o valor de E_l em 0,01 e não mais 0,1, fazendo com que os gradientes continuem estáveis e não explodam ou desapareçam. O processo de treinamento é descrito na Figura 3.9 e durou aproximadamente 10 horas. Uma vez treinado o modelo, tal processo não precisa ser repetido e o mesmo está pronto para operar.

Figura 4.4: Fluxograma descrevendo uma instância do processo de treinamento.



4.7 Análise dos Resultados

A presente Seção discute o método de avaliação empregado na avaliação dos resultados. Uma análise quantitativa eficaz dos resultados obtidos com modelos generativos (a rede PredNet é um modelo desse tipo) ainda é um problema não resolvido. Esse tipo de modelo é usado em uma ampla gama de aplicações e, dado a sua versatilidade, existe grande heterogeneidade na criação e avaliação desse tipo modelos (THEIS; OORD; BETHGE, 2015).

Lotter et al. (2016) utilizaram o EQM ou *Mean Squared Error* (MSE) e o índice de similaridade estrutural (SSIM) para avaliar a sua implementação da arquitetura PredNet. Seguiremos aqui a mesma metodologia.

4.7.1 Erro Quadrático Médio (EQM) ou *Mean Squared Error* (MSE)

O EQM é uma das métricas mais utilizadas na análise da qualidade de vídeos e imagens. Ele basicamente quantifica a diferença absoluta entre um sinal discreto e o original (DOSSELMANN; XUE, 2005; WANG et al., 2004). No nosso caso, a diferença entre imagem de referência e a imagem simulada pela rede, pixel à pixel. Portanto, valores de EQM baixos indicam que o modelo foi capaz de gerar uma previsão parecida com a imagem de referência (LOTTER; KREIMAN; COX, 2016). A formulação do EQM pode ser encontrada na Equação 4.1.

Equação 4.1:Descreve o EQM/MSE. x_i é a i-ézima amostra do sinal original, y_i é a i-ézima amostra do sinal atual e N é o número de amostras (sinal original e sinal atual). Embora trivial, o *MSE* é amplamente utilizado, graças a sua simplicidade.

$$\mathbf{EQM} = \frac{1}{N}\sum_{i=0}^{N-1}(x_i - y_i)^2$$

Fonte: Dosselmann e Xue (2005)

4.7.2 Índice de Similaridade Estrutural ou *Structural Similarity Index* (SSIM)

Apesar de sua simplicidade, o EQM é incapaz de detectar mudanças estruturais na imagem (artefatos, distorções e etc) (DOSSELMANN; XUE, 2005; WANG et al., 2004; WANG; BOVIK, 2009). Portanto, outros índices capazes de detectar mudanças estruturais entre duas imagens foram desenvolvidos. O SSIM foi desenvolvido por Wang et al. (2004) e está correlacionado com a forma como a visão humana percebe a qualidade de uma imagem (HORÉ; ZIOU, 2010; WANG et al., 2004). O SSIM mede a similaridade de um sinal, em relação a um sinal de referência utilizando três fatores: luminosidade, contraste e estrutura. O SSIM possui um intervalo de variação de -1 a 1. O valor 1 é obtido quando um sinal é igual ao sinal de referência (imagem é igual a imagem de referência), o valor -1 é obtido quando as imagens são completamente diferentes (WANG et al., 2004). Quanto mais próximo de 1, mais parecidas são as imagens. As Equações 4.2 e 4.3 descrevem o SSIM.

Equação 4.2: Formulação geral do SSIM, onde *f* representa a imagem atual e g a imagem ou sinal de referência.

$$SSIM(f,g) = l(f,g)c(f,g)s(f,g)$$

Fonte: Horé e Ziou (2010)

Equação 4.3: Descreve cada termo que compõe o SSIM. Onde *f* é a imagem gerada, *g* é a imagem de referência, *l* representa a função de comparação da luminosidade, μ representa a luminosidade média, *c* representa a função de comparação do contraste, σ é o desvio padrão, *s* representa a função de comparação estrutural e σ_{fg} é a covariância entre as duas imagens. As constantes C_1, C_2 e C_3 evitam a divisão por zero

$$l(f,g) = \frac{2\mu_f\mu_g + C_1}{\mu_f^2 + \mu_g^2 + C_1}$$

$$c(f,g) = \frac{2\sigma_f\sigma_g + C_2}{\sigma_f^2 + \sigma_g^2 + C_2}$$

$$s(f,g) = \frac{\sigma_{fg} + C_3}{\sigma_f\sigma_g + C_3}$$

Fonte: (HORÉ; ZIOU, 2010)

5 Resultados

Este Capítulo apresenta os resultados obtidos ao final do processo de treinamento. As imagens aqui apresentadas fazem parte do conjunto de testes. O resultado final do modelo é uma sequência de 5 imagens.

A partir do modelo resultante do treinamento da rede PredNet, foram geradas as previsões com imagens do conjunto de testes. Os resultados médios, obtidos em todo o conjunto de teste, podem ser observados no Gráfico 4.1. O gráfico da esquerda apresenta os valores de EQM (em inglês *MSE*) médios obtidos para cada previsão realizada pelo modelo, a partir do conjunto de testes. O gráfico da direita apresenta o mesmo cálculo, porém agora, feito para o SSIM. Pode ser observado, que os melhores resultados são obtidos na previsão de 15 min (EQM próximo a zero e SSIM próximo a 1). Observa-se também que as métricas decaem à medida que as previsões avançam no tempo.

A previsão mais curta que os modelos podem fazer é de 15 minutos. Esse tempo corresponde a amostragem mínima disponível do satélite METEOSAT (imagens disponíveis de 15 em 15 minutos). O tempo mínimo de previsão pode ser aumentado ou diminuído, dependendo apenas do satélite escolhido.

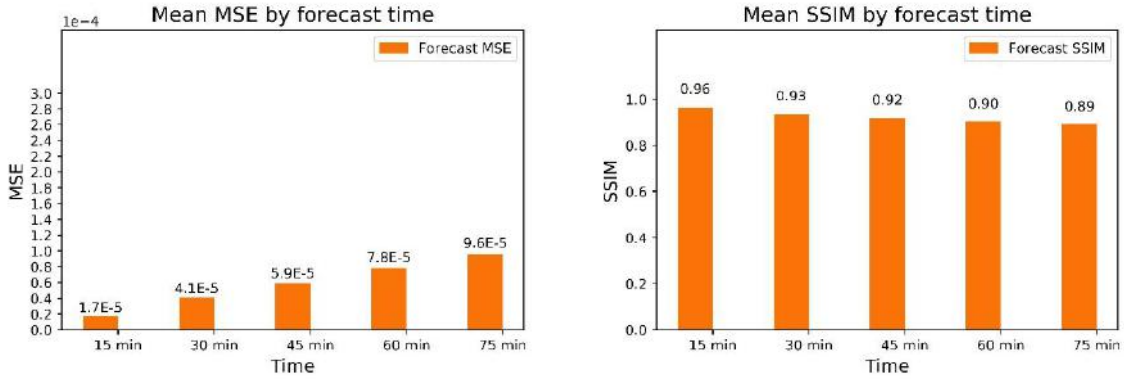


Gráfico 4.1: EQM médio para todas as previsões (esquerda) e SSIM para todas as previsões (direita), calculadas a partir do conjunto de testes.

Nas Seções a seguir, as previsões do dia 01/11/2010 gerada pelo modelo, assim como sua as métricas, associadas a qualidade dos resultados, são apresentadas como exemplo da ferramenta apresentada nesse trabalho.

5.2 Previsão para 15 minutos.

As imagens geradas pela rede são mais “enevoadas” (Figura 4.1- direita) do que as imagens de referência (Figura 4.1- esquerda). Contudo, representam bem a forma,

posicionamento e TBs (cor do pixel) das estruturas de nuvens, quando comparadas à observação. Pequenas diferenças são observadas nos núcleos mais frios; cores rosa e azul claro na imagem (Figura 4.1 - esquerda). Na imagem a seguir o EQM obtido foi de 2,33E-5. O SSIM manteve-se próximo a 1 (0,96).

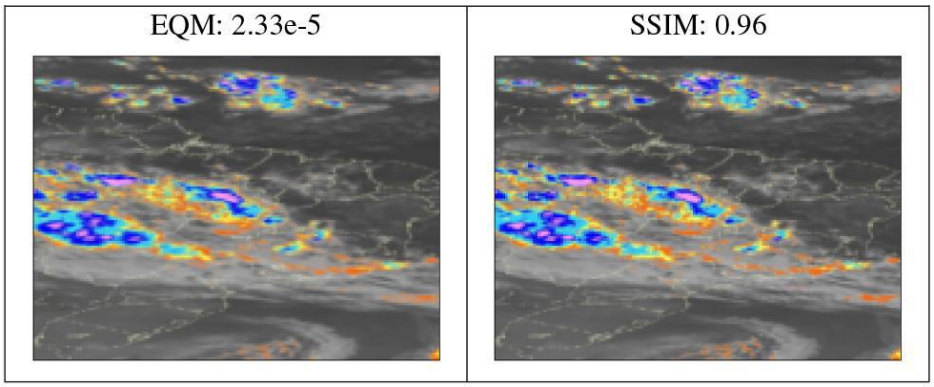


Figura 5.1: Resultado obtido a partir do treinamento, para o dia 01/11/2010 2:30Z. Previsto (esquerda), observação (direita).Previsão para 30 minutos.

Na previsão de 30 minutos (Figura 4.2), a rede passa a usar a previsão anterior (15 min) para fazer a previsão seguinte. O EQM continua próximo a zero, 5,61E-5. Apesar de baixos, os valores de EQM são significativamente maiores do que na previsão de 15 minutos (Figura 4.1). O SSIM manteve seu valor próximo a 1 (0,93). Entretanto, os valores obtidos são inferiores que os obtidos na previsão de 15 minutos (Figura 4.1).

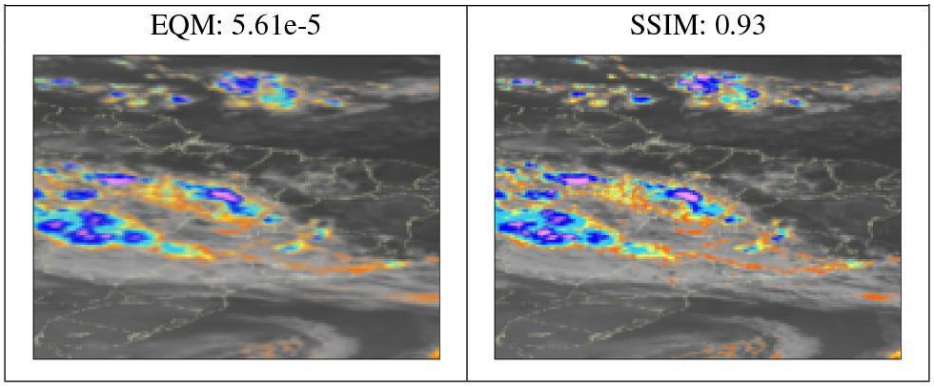


Figura 5.2: Resultado obtido a partir do treinamento (modelo A), para o dia 01/11/2010 2:45Z. Previsto (esquerda), observação (direita).

Podemos observar também que a previsão de 30 minutos (Figura 4.2 - esquerda) é mais “borrada” do que a previsão de 15 minutos (Figura 4.1- esquerda). Uma maior distorção na forma dos núcleos convectivos mais intensos, cores rosa e azul claro, é perceptível em ambas as imagens (Figura 4.2 - esquerda). Contudo, o posicionamento dos sistemas não é alterado (indicado também pelo SSIM, acima próximo a 1).

5.3 Previsão para 45 minutos

Na previsão de 45 minutos (Figura 4.3), as métricas do modelo apresentaram valores os seguintes valores: EQM: 8,03E-5 e SSIM: 0,92. A previsão mantém a tendência de perda de nitidez observada nas previsões anteriores (Figuras 4.1 e 4.2 - esquerda). A distorção nos núcleos menores é ainda maior, porém ainda é possível identifica-los (áreas em rosa) (Figura 4.3 - esquerda). Todavia, o posicionamento geral dos sistemas atmosféricos ainda é bem consistente ao compararmos com a imagem de referência (Figura 4.3 - direita).

O EQM continua próximo a zero, contudo, seu valor é maior que o observado na previsão anterior (Figura 4.2). O SSIM continua a apresentar valores próximos a 1 (Figura 4.3), mas acompanhando a tendência de diminuição observada nas previsões anteriores (Figuras 4.1 e 4.2). Portanto, apresentando um desempenho inferior, ao ser comparado com os índices obtidos nas previsões anteriores, indicando maiores diferenças estruturais na imagem prevista (Figura 4.3 - direita).

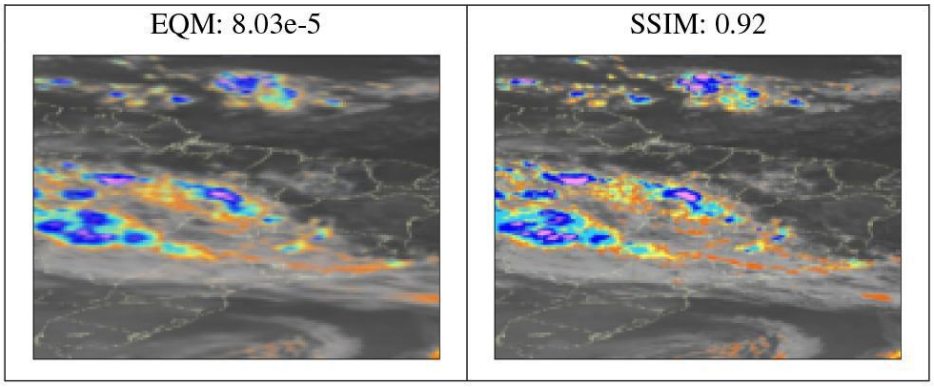


Figura 5.3: Resultado obtido a partir do treinamento, para o dia 01/11/2010 3:00Z. Previsto (esquerda), observação (direita).

5.4 Previsão para 60 minutos

Na previsão para 60 minutos (Figura 4.4 - esquerda), podemos observar que a tendência de diminuição de nitidez permanece. O formato dos núcleos mais frios (rosa), e dos sistemas menores, já não é mais delineado corretamente (Figura 4.4-esquerda), em comparação com a imagem de referência (Figura 4.4 - direita). Os fenômenos de maior escala ainda apresentam um posicionamento correto, ao se comparados a imagem de referência (Figura 4.4). Em geral, o posicionamento dos sistemas atmosféricos de maior escala, é melhor representado nessa previsão. O EQM (Figura 4.8) continua com um valor próximo a zero, 1,05E-4, sendo, contudo, uma ordem de grandeza maior que o obtido nas previsões anteriores (Figuras 4.1,

4.2, e 4.3) e sua tendência de degradação é ainda mais acentuada. O mesmo ocorre com o SSIM, 0,89, todavia, ainda próximo de 1, indicando grande similaridade entre a imagem gerada pelo modelo e a imagem de referência (Figura 4.4).

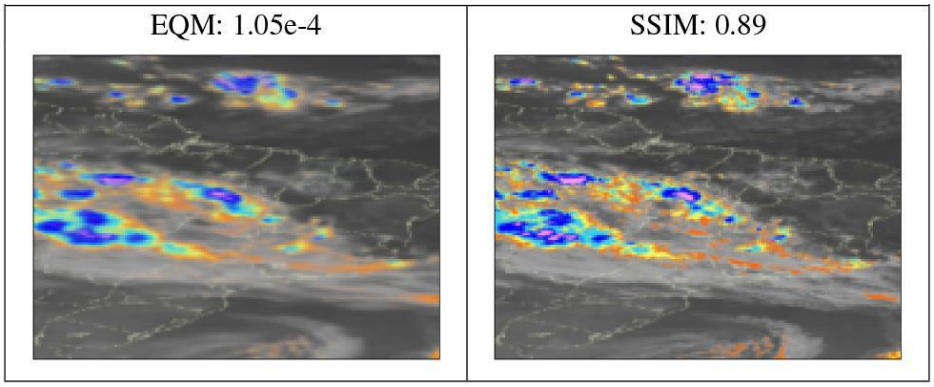


Figura 5.4: Resultado obtido para o dia 01/11/2010 3:15Z. Previsto (esquerda), observação (direita).

5.5 Previsão para 75 minutos

Na previsão para 75 minutos com o modelo (Figura 4.5), as tendências observadas nas previsões anteriores (15, 30, 45 e 60 minutos - Figuras 4.1, 4.2, 4.3 e 4.4, respectivamente) são ainda mais marcantes. Essa previsão obteve os valores mais baixos nas duas métricas utilizadas para avaliar as imagens.

Os valores são: EQM 1,24E-4, valor que aumentou 0,21E-4, em relação ao valor obtido na previsão de 60 minutos (Figura 4.4). O valor de SSIM de 0,87 (Figura 4.5) é o valor mais baixo de SSIM obtido, até o momento. Esse valor condiz com as alterações observadas na imagem gerada pelo modelo (Figura 4.5 - esquerda). Os sistemas atmosféricos, de maior escala, ainda são corretamente posicionados na imagem prevista (Figura 4.5 - esquerda). Já não é possível identificar os núcleos mais frios (rosa) e menores, que podem ser observados na imagem de referência (Figura 4.5 - direita). Entretanto, ainda seria possível utilizar essa imagem para auxílio na previsão.

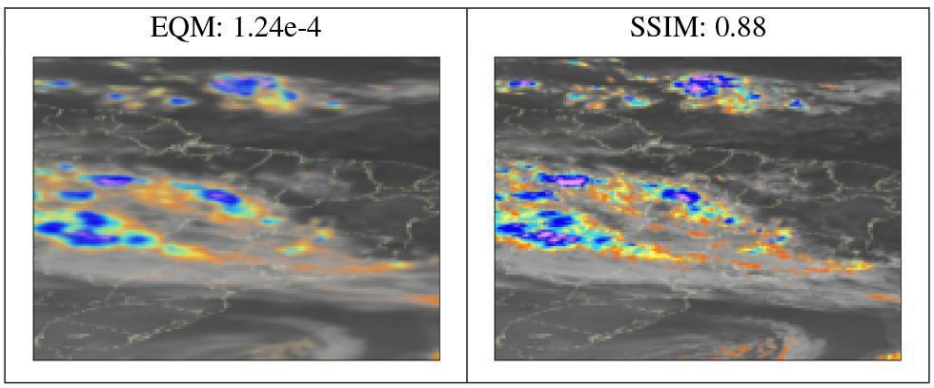


Figura 5.9: Resultado obtido a partir do modelo B, para o dia 01/11/2010 3:30Z. Previsto (esquerda), observação (direita).

6 Conclusões

O presente trabalho apresentou uma forma alternativa à previsão do tempo de curto prazo, utilizando DL para treinar um modelo com as imagens de satélite do IR realçado. Dois modelos foram utilizados: no primeiro, a rede foi treinada do zero (modelo A); no segundo, optou-se por utilizar o modelo gerado por Lotter et al. (2016), para prever frames de uma câmera montada em um carro (também treinados na rede PredNet). O modelo treinado por Lotter et al. (2016) (modelo B), apresentou resultados muito melhores do que o esperado ao utilizar o modelo A. Isso indica que durante o treinamento da arquitetura PredNet, com as imagens no IR, ocorreu um *Overfitting*. O *Overfitting* ocorre quando o intervalo entre os erros do processo de treinamento e os do processo de teste são muito grandes. Por isso, se a rede for treinada a partir de pesos obtidos em um treinamento anterior, com dados diferentes, possivelmente apresentará resultados superiores aos encontrados nesse estudo (processo conhecido como *fine tunning*). Ou seja, ainda é possível melhorar os resultados obtidos com o modelo A, que possivelmente apresentará resultados melhores que o modelo B, após o *finetuning*.

As métricas de avaliação geral, para o modelo B (todos os tempos de previsões analisados) apresentaram resultados satisfatórios, com EQM e SSIM médios de 0,000037, e 0,93, respectivamente. Os melhores resultados foram obtidos para a previsão de 15 minutos no futuro, em ambos os modelos.

Uma grande vantagem dessa nova ferramenta apresentada é o custo computacional baixo do modelo, se comparado ao de um supercomputador em um centro de previsão do tempo. Uma vez que a rede foi treinada, o modelo demora menos e 1 minuto para fazer a previsão. Além de ser uma alternativa mais barata, pois essa ferramenta pode ser utilizada dispondo somente de um computador e usando imagens de satélite, que são obtidas facilmente.

Finalmente conclui-se que devido aos bons resultados apresentados e ao baixo custo, essa ferramenta mostra-se promissora para o auxílio da previsão de curto prazo.

Referências Bibliográficas

ABADI, M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. **In: 12th USENIX Symposium on Operating Systems Design and Implementation**, Savannah, GA, USA, 2016. Disponível em: <http://arxiv.org/abs/1603.04467>

ABBE, C. The Physical Basis of Long-Range Weather Forecasts. **Monthly Weather Review**, v. 29, n. 12, p. 551–561, 1901. Disponível em: <http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1901)29%5B551c:TPBOLW%5D2.0.CO;2>

ARAKAWA, A. Adjustment Mechanisms in Atmospheric Models. **Journal of the Meteorological Society of Japan**, v. 75, n. 1B, p. 155–179, 1997.

ARAKAWA, A. The cumulus parameterization problem: Past, present, and future. **Journal of Climate**, v. 17, n. 13, p. 2493–2525, 2004.

B. KLEIN L. WOLF, Y. A. A Dynamic Convolution Layer for Short Range Weather Prediction. **In: The IEEE Conference on Computer Vision and Pattern Recognition**, Boston, MA, USA, 2015. p 4840-4848.

BAUER, P.; THORPE, A.; BRUNET, G. The quiet revolution of numerical weather prediction. **Nature**,v. 525, n. 7567, p. 47–55, 2015. Disponível em: <http://www.nature.com/doifinder/10.1038/nature14956>

BeginnersGuide/Overview - Python Wiki. 2017. Disponível em: <https://wiki.python.org/moin/BeginnersGuide/Overview>. Acesso em: 28 nov. 2017.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 35, n. 8, p. 1798–1828, 2013.

BJERKNES, V.; VOLKEN, E.; BRÖNNIMANN, S. The problem of weather prediction, considered from the viewpoints of mechanics and physics. **Meteorologische Zeitschrift**, v. 18, n. 6, p. 663–667, 1904.

BOUCHER, O. et al. Clouds and Aerosols. **Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change**, p. 571–657, 2013.BRETHERTON, C. S. et al. A GCSS boundary-layer cloud model intercomparison study of the first ASTEX Lagrangian experiment. **Boundary-Layer Meteorology**, v. 93, n. 3, p. 341–380, 1999.

BÜTEPAGE, J. et al. Deep representation learning for human motion prediction and classification. **In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, Honolulu, Hawaii , 2017. Disponível em: <http://arxiv.org/abs/1702.07486>

CATALDI, M. et al. Análise das Previsões de Precipitação Obtidas com a Utilização do Modelo Eta como Insumo para Modelos de Previsão Semanal de Vazão Natural. **Revista Brasileira de Recursos Hídricos**, v. 12, n. January 2007, p. 5–12, 2007. Disponível em: <https://www.abrh.org.br/sgcv3/UserFiles/Sumarios/226b3924d76b5ab925b5f74938b67d1b_ea017246934cc85146998705c618a548.pdf>

CERLINI, P. B.; EMANUEL, K. A.; TODINI, E. Orographic effects on convective precipitation and space-time rainfall variability: preliminary results. **Hydrology and Earth System Sciences Discussions**, v. 9, n. 4, p. 285–299, 2005. Disponível em: <https://hal.archives-ouvertes.fr/hal-00304830/>

CHAN, S. C. et al. The value of high-resolution Met Office regional climate models in the simulation of multihourly precipitation extremes. **Journal of Climate**, v. 27, n. 16, p. 6155–6174, 2014.

CHARNEY, J. G.; FJÖRTOFT, R.; NEUMANN, J. Von. Numerical Integration of the Barotropic Vorticity Equation. **Tellus**, v. 2, n. 4, p. 237–254, 1950. Disponível em: <https://www.tandfonline.com/doi/full/10.3402/tellusa.v2i4.8607>

CHEN, F. et al. The integrated WRF/urban modelling system: Development, evaluation, and applications to urban environmental problems. **International Journal of Climatology**, v. 31, n. 2, p. 273–288, 2011.

CHING, J. et al. Convectively Induced Secondary Circulations in Fine-Grid Mesoscale Numerical Weather Prediction Models. **Monthly Weather Review**, v. 142, n. 9, p. 3284–3302, 2014. Disponível em: <http://journals.ametsoc.org/doi/abs/10.1175/MWR-D-13-00318.1>

CHOLLET, F. et al.**Keras**.GitHub, 2015. Disponível em: < https://github.com/keras-team/keras>. Acesso em: 28 nov. 2017.

CLOKE, H. L. et al. Modelling climate impact on floods with ensemble climate projections. **Quarterly Journal of the Royal Meteorological Society**, v. 139, n. 671, p. 282–297, 2013.

CONTRIBUIDORES DA WIKIPÉDIA. **Shell script**. 2017a. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Shell_script&oldid=49752758>. Acesso em: 28 nov. 2017.

CONTRIBUIDORES DA WIKIPÉDIA. **Bash**. 2017b. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Bash&oldid=49752754>. Acesso em: 28 nov. 2017.

COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE Transactions on Information Theory**, v. 13, n. 1, p. 21–27, 1967. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1053964>

CUO, L.; PAGANO, T. C.; WANG, Q. J. A Review of Quantitative Precipitation Forecasts and Their Use in Short- to Medium-Range Streamflow Forecasting. **Journal of Hydrometeorology**, v. 12, n. 5, p. 713–728, 2011. Disponível em: <http://journals.ametsoc.org/doi/abs/10.1175/2011JHM1347.1>

DEJONG, G. Generalisations Based on Explanations. **In: Proceedings of the Seventh International Joint Conference on Artificial Intelligence**, Vancouver,B.C, Canada, 1981. p 67–69.

DI MICHELE, S. et al. Bayesian algorithm for microwave-based precipitation retrieval: Description and application to TMI measurements over ocean. **IEEE Transactions on Geoscience and Remote Sensing**, v. 43, n. 4, p. 778–791, 2005.

DEEPLEARNING4J: OPEN-SOURCE, DISTRIBUTED DEEP LEARNING FOR THE

JVM.**A Beginner's Guide to Recurrent Networks and LSTMs**, 2017. Disponível em: <https://deeplearning4j.org/lstm>. Acesso em: 6 dez. 2017.

DING, S. et al. Evolutionary artificial neural networks: A review. **Artificial Intelligence Review**, v. 39, n. 3, p. 251–260, 2013.

DOSSELMANN, R.; XUE, D. Y. Existing and emerging image quality metrics. **Canadian Conference on Electrical and Computer Engineering**, v. 2005, n. May, p. 1906–1913, 2005.

DU, J.; XU, Y. Hierarchical deep neural network for multivariate regression. **Pattern Recognition**, v. 63, n. June 2015, p. 149–157, 2017. Disponível em: <http://dx.doi.org/10.1016/j.patcog.2016.10.003>

DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning.**ArXiv e-prints,** p. 1–28, 2016. Disponível em: <http://arxiv.org/abs/1603.07285>

ENGLHISH, S. J. et al. A comparison of the impact of TOVS and ATOVS satellite sounding data on the accuracy of numerical weather forecasts. **Quarterly Journal of the Royal Meteorological Society**, v. 126, n. 569, p. 2911–2931, 2000.

GÉRARD, É.; SAUNDERS, R. W. Four-dimensional variational assimilation of special sensor microwave/imager total column water vapour in the ECMWF model. **Quarterly Journal of the Royal Meteorological Society**, v. 125, n. 560, p. 3077–3101, 1999. Disponível em: <http://dx.doi.org/10.1002/qj.49712556014>

GHADERI, A.; SANANDAJI, B. M.; GHADERI, F. Deep Forecast: Deep Learning-based Spatio-Temporal Forecasting. **In:Thirty-fourth International Conference on Machine Learning**, Sydney, Australia, 2017. Disponível em: <http://arxiv.org/abs/1707.08110>

GOLDING, B. W. Quantitative precipitation forecasting in the UK. **Journal of Hydrology**, v. 239, n. 1–4, p. 286–305, 2000.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Massachusets: MIT press,2016. 785 p. Disponível em: <http://www.deeplearningbook.org>

GREGORY, D. et al. Revision of convection, radiation and cloud schemes in the ECMWF Integrated Forecasting System. **Quarterly Journal of the Royal Meteorological Society**, v. 126, n. 566, p. 1685–1710, 2000.

HAIDEN, T. et al. Evaluation of ECMWF forecasts, including the 2016 resolution upgrade. **ECMWF Tech. Memo.**v. 792, n. December, 2016. Disponível em: <http://www.ecmwf.int/sites/default/files/elibrary/2015/15275-evaluation-ecmwf-forecasts-including-2014-2015-upgrades.pdf>

HASSAN, V. V.; BARCELLOS, P. da C. L.; DA SILVA, J. C. Método preditivo para acionamento das sirenes nas comunidades vulneráveis a escorregamentos no município de Duque de Caxias, Estado do Rio de Janeiro, Brasil. **Anuario do Instituto de Geociencias**, Rio de Janeiro, RJ, v. 40, n. 1, p. 127–134, 2017.

HEYE, A.; VENKATESAN, K.; CAIN, J. Precipitation Nowcasting: Leveraging Deep Recurrent Convolutional Neural Networks. **In:Proceedings - Cray User Group** Redmond, WA, 2017. Disponível em: <https://cug.org/proceedings/cug2017_proceedings/includes/files/pap155s2-ile1.pdf>

HINTON, G. E.; OSINDERO, S.; TEH, Y. W. A fast learning algorithm for deep belief nets. **Neural computation**,v. 18, n. 7, p. 1527–54, 2006. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/16764513>

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. **Neural Computation**,v. 9, n. 8, p. 1735–1780, 1997. Disponível em: <http://www7.informatik.tu-muenchen.de/~hochreit%5Cnhttp://www.idsia.ch/~juergen>

HORÉ, A.; ZIOU, D. Image quality metrics: PSNR vs. SSIM.**In: Proceedings - International Conference on Pattern Recognition**, Instanbul, Turkey, 2010.p. 2366–2369.

HOSSAIN, M. et al. Forecasting the weather of Nevada: A deep learning approach.**In:Proceedings of the International Joint Conference on Neural Networks**, Killarney, Ireland, 2015. p. 2–7.

JAKOB, C.; KLEIN, S. A. A parametrization of the effects of cloud and precipitation overlap for use in general-circulation models. **Quarterly Journal of the Royal Meteorological Society**, v. 126, n. 568, p. 2525–2544, 2000.

JANSSEN, P. A. E. M. et al. Impact and feedback of ocean waves on the atmosphere. **Atmosphere Ocean Interactions**, v. 1, n. August, p. 155–197, 2002.

KALNAY, E. **Atmospheric modeling, data assimilation and predictability**. [s.l.] : Cambridge University Press, 2002.

KAUFMANN, P.; SCHUBIGER, F.; BINDER, P. Precipitation forecasting by a mesoscale numerical weather prediction (NWP) model: eight years of experience. **Hydrology and Earth System Sciences**, v. 7, p. 812–832, 2003.

KIMURA, R. Numerical weather prediction. **Journal of Wind Engineering and Industrial Aerodynamics**, v. 90, n. 12–15, p. 1403–1414, 2002.

KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization.**In:the 3rd International Conference for Learning Representations**, San Diego, California, USA, 2014.p. 1–15. Disponível em: <http://arxiv.org/abs/1412.6980>

LAZO, J. K.; MORSS, R. E.; DEMUTH, J. L. 300 Billion Served. **Bulletin of the American Meteorological Society**, v. 90, n. 6, p. 785–798, 2009.

LE, Q. V. et al. Building high-level features using large scale unsupervised learning.**In: The IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'13)**, Vancouver, BC, Canada, 2013.p 8595–8598.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, 1998. p 2278–2323.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015. Disponível em: <http://arxiv.org/abs/1603.05691>

LECUN, Y.; BOTTOU, L.; BENGIO, Y. Reading Checks with Multilayer Graph Transformer Networks. **In:The IEEE International Conference on Acoustics, Speech, and Signal Processing**,1997.p. 151–154.

LI, G. Visual Saliency Based on Multiscale Deep Features. **In:The IEEE Conference on Computer Vision and Pattern Recognition**, Boston, MA, USA, 2015. p 5455-

5463.

LITTA, A.; MARY IDICULA, S.; NAVEEN FRANCIS, C. Artificial Neural Network Model for the Prediction of Thunderstorms over Kolkata. **International Journal of Computer Applications**, v. 50, n. 11, p. 50–55, 2012.

LORENZ, E. N. Reflections on the Conception, Birth, and Childhood of Numerical Weather Prediction. **Annual Review of Earth and Planetary Sciences**, v. 34, n. 1, p. 37–45, 2006. Disponível em: <http://www.annualreviews.org/doi/10.1146/annurev.earth.34.083105.102317>

LOTTER, W.; KREIMAN, G.; COX, D**.** Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning. **In:International Conference on Learning Representations,**Toulon, France, 2016**.**

LOWREY, M. R. K.; YANG, Z.L. Assessing the Capability of a Regional-Scale Weather Model to Simulate Extreme Precipitation Patterns and Flooding in Central Texas. **Weather and Forecasting**, v. 23, n. 6, p. 1102–1126, 2008. Disponível em: <http://journals.ametsoc.org/doi/abs/10.1175/2008WAF2006082.1%5Cnhttp://journals.ametsoc.org/doi/pdf/10.1175/2008WAF2006082.1>

MAGNUSSON, L. et al. Evaluation of Medium-Range Forecasts for Hurricane Sandy. **Monthly Weather Review**, v. 142, n. 5, p. 1962–1981, 2014. Disponível em: <http://journals.ametsoc.org/doi/abs/10.1175/MWR-D-13-00228.1>

MAPES, B. et al. The mesoscale convection life cycle: Building block or prototype for large-scale tropical waves? **Dynamics of Atmospheres and Oceans**, v. 42, n. 1–4, p. 3–29, 2006.

MCBRIDE, J. L; EBERT, E. E. Verification of quantitative precipitation forecasts from operational numerical weather prediction models over Australia. **Weather and Forecasting**, v. 15, n. 1, p. 103–121, 2000.

MCNALLY, A. P. et al. The use of TOW level-lb. **Quarterly Journal of the Royal Meteorological Society**, v.120, p. 689–724, 2000.

MNIH, V. et al. Playing Atari with Deep Reinforcement Learning. In: **Neural Information Processing SystemsDeep Learning Workshop**, Lake Tahoe, USA, 2013. p. 1–9. Disponível em: <http://arxiv.org/abs/1312.5602>

MÜLLER, E. H.; SCHEICHL, R. Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction. **Quarterly Journal of the Royal Meteorological Society**, v. 140, p. 2608–2624, 2014.

NAREJO, S.; PASERO, E. Meteonowcasting using Deep Learning Architecture. **International Journal of Advanced Computer Science and Applications**, v. 8, n. 8, p. 16–23, 2017.

NASCIMENTO, E. L. Previsão De Tempestades Severas Utilizando-Se Parâmetros Convectivos E Modelos De Mesoescala: Uma Estratégia Operacional Adotável No Brasil? **Revista Brasileira de Meteorologia**, v. 20, p. 121–140, 2005.

OH, B. M. et al. Image-based modeling and photo editing. **In:Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01**, Los Angeles, CA, USA, 2001. p. 433–442.

OLAH, C. **Understanding LSTM Networks -- colah's blog**. 2015. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: 6 dez. 2017.

OLSON, D. A.; JUNKER, N. W.; KORTY, B. Evaluation of 33 Years of Quantitative Precipitation Forecasting at the NMC. **Weather Forecast Branch, Meteorological Operations Division, National Meteorological Center**,Camp Springs, MD, USA, 1995.

OpenGrADS. 2017. Disponível em: <http://opengrads.org/>. Acesso em: 28 nov. 2017.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training Recurrent Neural Networks. **In: The 30th International Conference on Machine Learning, Atlanta, USA**, 2012. Disponível em: <http://arxiv.org/abs/1211.5063>

RABIER, F. et al. The ECMWF operational implementation of four-dimensional variational assimilation. I: Experimental results with simplified physics. **Quarterly Journal of the Royal Meteorological Society**, v. 126, n. 564, p. 1143–1170, 2000. Disponível em: <http://www.ingentaselect.com/rpsv/cgi-bin/cgi?ini=xref&body=linker&reqdoi=10.1256/smsqj.56414>

RANDALL, D. A. Beyond deadlock. **Geophysical Research Letters**, v. 40, n. 22, p. 5970–5976, 2013.

RICHARD, E. et al. Intercomparison of mesoscale meteorological models for precipitation forecasting. **Hydrology and Earth System Sciences**, v. 7, n. 6, p. 799–811, 2003.

RICHARDSON, L. F. **Weather Prediction by Numerical Process**. First ed. [s.l.] : Cambridge University Press, 1922.

RIVOLTA, G. et al. Artificial neural-network technique for precipitation nowcasting from satellite imagery. **Advances in Geosciences**, v. 7, p. 97–103, 2006. Disponível em: <http://www.adv-geosci.net/7/97/2006/>

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in The Brain. **Psychological Review**,v. 65, n. 6, p. 386–408, 1958. Disponívelem: <http://psycnet.apa.org/journals/rev/65/6/386.pdf%5Cnpapers://c53d1644-cd41-40df-912d-ee195b4a4c2b/Paper/p15420>

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, 1986.

SABA, T.; REHMAN, A.; ALGHAMDI, J. S. Weather forecasting based on hybrid neural model. **Applied Water Science**, v. 7, n. 7, p. 1–6, 2017. Disponível em: <http://link.springer.com/10.1007/s13201-017-0538-0>

SCHAMM, K. et al. Global gridded precipitation over land: A description of the new GPCC First Guess Daily product. **Earth System Science Data**, v. 6, n. 1, p. 49–60, 2014.

SHALF, J.; QUINLAN, D.; JANSSEN, C. Rethinking hardware-software codesign for exascale systems. **Computer**, v. 44, n. 11, p. 22–30, 2011.

SHI, X. et al. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. **In: Neural Information Processing Systems Conference**, Montreal, Canada, 2015. p. 1–12. Disponível em: <http://arxiv.org/abs/1506.04214>

SHUKLA, B. P.; KISHTAWAL, C. M.; PAL, P. K. Satellite-Based Nowcasting of Extreme Rainfall Events over Western Himalayan Region. **IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing**, v. 10, n. 5, p. 1681–1686, 2017.

SIMMONS, A. J.; HOLLINGSWORTH, A. Some aspects of the improvement in skill of numerical weather prediction. **Quarterly Journal of the Royal Meteorological Society**, v. 128, n. 580, p. 647–677, 2002. Disponível em: <http://doi.wiley.com/10.1256/003590002321042135>

SIMMONS, A. J. Observations, assimilation and the improvement of global weather prediction-some results from operational forecasting and ERA-40.**In: in Seminar on Recent developments in data assimilation for atmosphere and ocean**, Sinfield Park, Reading, 2003. p. 428–458.

SIMMONS, A J. et al. Stratospheric water vapour and tropical tropopause temperatures in {ECMWF} analyses and multi-year simulations. **Quarterly Journal of Royal Meteorological Society**, v. 125, p. 353–386, 1999.

SOLANKI, K.; DHANKAR, A. A review on Machine Learning Techniques. **International Journal of Advanced Research in Computer Science**, v. 8, n. 3, p. 778–782, 2017.

STENSRUD, D. J.; YUSSOUF, N. Reliable Probabilistic Quantitative Precipitation Forecasts from a Short-Range Ensemble Forecasting System. **Weather and Forecasting**, v. 22, n. 1, p. 3–17, 2007. Disponível em: <http://journals.ametsoc.org/doi/abs/10.1175/WAF968.1>

SUKOVICH, E. M. et al. Extreme Quantitative Precipitation Forecast Performance at the Weather Prediction Center from 2001 to 2011. **Weather and Forecasting**, v. 29, n. 4, p. 894–911, 2014. Disponível em: <http://dx.doi.org/10.1175/WAF-D-13-00061.1>

THEIS, L.; OORD, A. Van den; BETHGE, M. A note on the evaluation of generative models. **In: International Conference on Learning Representations**, San Juan, Puerto Rico, 2016. Disponível em: <http://arxiv.org/abs/1511.01844>

THOMAS, C.; CORPETTI, T.; MÉMIN, E. Data assimilation for convective-cell tracking on meteorological image sequences. **IEEE Transactions on Geoscience and Remote Sensing**, v. 48, n. 8, p. 3162–3177, 2010.

WAN, J. et al. Deep learning for content-based image retrieval: A comprehensive study. **In:Proceedings of the ACM International Conference on Multimedia**, Orlando, FL, USA, 2014. p 157–166.

WANG, Z. et al. Image quality assessment: From error visibility to structural similarity. **IEEE Transactions on Image Processing**, v. 13, n. 4, p. 600–612, 2004.

WANG, Z.; BOVIK, A. C. Mean squared error: Lot it or leave it? A new look at signal fidelity measures. **IEEE Signal Processing Magazine**, v. 26, n. 1, p. 98–117, 2009.

WERBOS, P. J. Backpropagation Through Time: What It Does and How to Do It. **Proceedings of the IEEE**, v. 78, n. 10, 1990. p 1550–1560.

WIKIPEDIA CONTRIBUTORS. **Pointwise product - Wikipedia**. 2017. Disponível em: <https://en.wikipedia.org/wiki/Pointwise_product>. Acesso em: 9 dez. 2017.

WILSON, D. R.; MARTINEZ, T. R. The general inefficiency of batch training for gradient descent learning. **Neural Networks**, v. 16, n. 10, p. 1429–1451, 2003.

WILSON, J. W. VERY SHORT PERIOD (0-6) FORECASTS OF THUNDERSTORMS **In: Proceedings of the Workshop on Warnings of Real-Time Hazards by Using Nowcasting Technology,** Sydney, Australia,2006.

WORLD METEOROLOGICAL ORGANIZATION. **Nowcasting**. 2017. Disponível em: <http://www.wmo.int/pages/prog/amp/pwsp/Nowcasting.html>. Acesso em: 21 nov. 2017.

YANO, J.-I. Subgrid-scale physical parameterization in atmospheric modeling: How can we make it consistent? **Journal of Physics A: Mathematical and Theoretical**, v. 49, n. 28, p. 284001, 2016.

ZHANG, F.; ODINS, A. M.; NIELSEN-GAMMON, J. W. Mesoscale Predictability of an Extreme Warm-Season Precipitation Event. **Weather and Forecasting**, v. 21, n. 2, p. 149–166, 2006.

ZHANG, Y. et al. Real-time air quality forecasting, part I: History, techniques, and current status. **Atmospheric Environment**, v. 60, p. 632–655, 2012. Disponível em: <http://dx.doi.org/10.1016/j.atmosenv.2012.06.031>

Apêndice A –Passo a passo da convolução

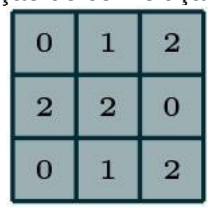
Com o intuito de manter a simplicidade, utilizaremos apenas um mapa características ao explicar a operação de convolução. Durante a convolução, o *Kernel* (Figura A-1), desliza através do mapa de características, matriz em azul (Figura A-2). A cada deslocamento, o *kernel* sobrepõe uma pequena área do mapa de entrada, de forma que cada elemento M_{ij}, da matriz de entrada, está relacionado a um elemento K_{ij}, no *kernel* (Figura A-3; esquerda). O produto $M_{ij} \times K_{ij}$ é calculado, produzindo uma nova matriz produto *P* (Figura A-3; centro). Em seguida, os elementos, P_{ij}, da matriz produto são somados, resultando em apenas um valor de saída, S_{ij}, por área encoberta pelo *kernel* (Figura A-3; direita). Cada elemento S_{ij}, será então utilizado para compor o mapa de características de saída, *S*, matriz em verde na Figura A-2. Depois, o *kernel* desliza novamente e o processo se repete.

Cada operação convolutiva gera um de mapa de característica de saída (*output feature maps*), que pode ser reutilizado, uma nova convolução. Esse processo pode ser repetido indefinidamente e gera um novo mapa de saída, com representações um pouco mais abstratas que as contidas no mapa de saída anterior (BÜTEPAGE et al., 2017; LECUN; BENGIO; HINTON, 2015).

A convolução mostrada na Figura A-2 é uma instância de uma convolução bidimensional, no entanto, o processo pode ser generalizado para N dimensões (DUMOULIN; VISIN, 2016; GOODFELLOW; BENGIO; COURVILLE, 2016). Em uma convolução tridimensional, o *kernel* utilizado é um cuboide que desliza através do volume (altura, largura e profundidade) do mapa de entrada (DUMOULIN; VISIN, 2016).

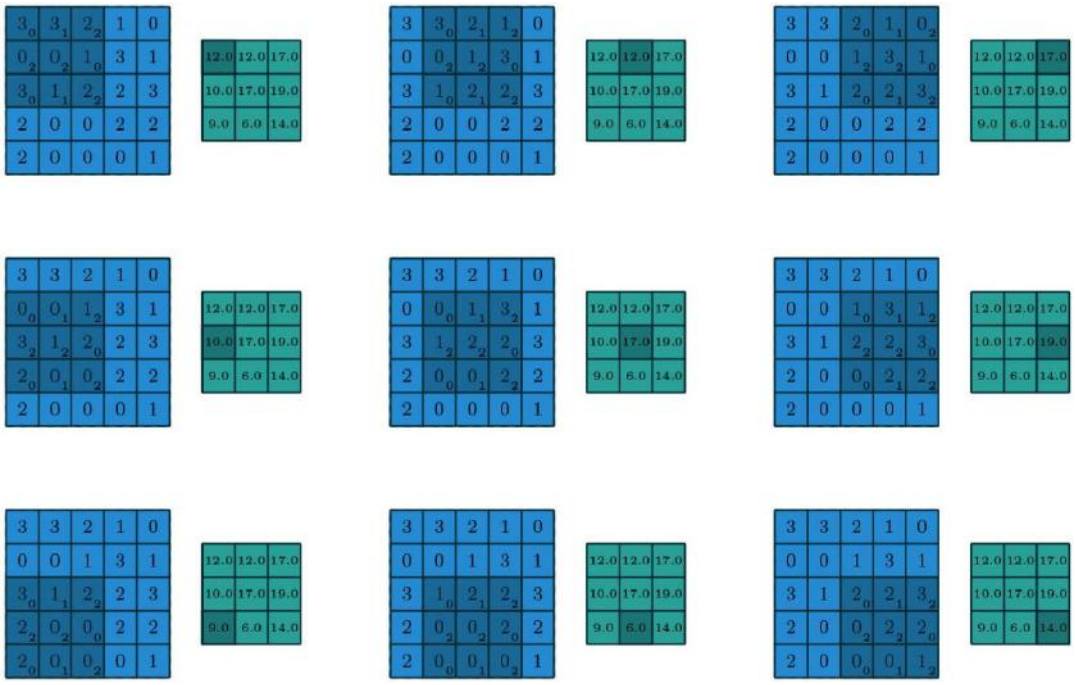
Se eventualmente, mais de um mapa de entrada estiver disponível: por exemplo, uma imagem RGB, em que cada canal é um mapa de entrada. O *kernel* deverá ser tridimensional, ou, cada mapa será convoluido com um *kernel* distinto e os mapas de saída resultantes serão somados, elemento por elemento, para produzir um mapa de saída ao final da convolução (Figura A-4) (DUMOULIN; VISIN, 2016).

Figura A-1: Matriz responsável pela operação de convolução o *Kernel*.



Fonte: Dumoulin e Visin (2016)

Figura A-2: Cálculo dos valores de saída de uma convolução discreta. Os valores calculados irão formar o mapa de característica de saída, em verde. A área sombreada na figura, representa a área encoberta pelo *kernel*. Pode-se observar que ela desliza da esquerda para a direita e de cima para baixo.



Fonte: Dumoulin e Visin (2016)

Figura A-3: Detalhes como o elemento, $S_{1,1}$, é gerado. Esse elemento faz parte do mapa de saída apresentado na Figura A-1. A matriz azul representa a área do mapa de entrada encoberta pelo *kernel* na Figura A-1 no topo a esquerda. Cada seta, partindo da matriz 3x3 azul, representa uma multiplicação.

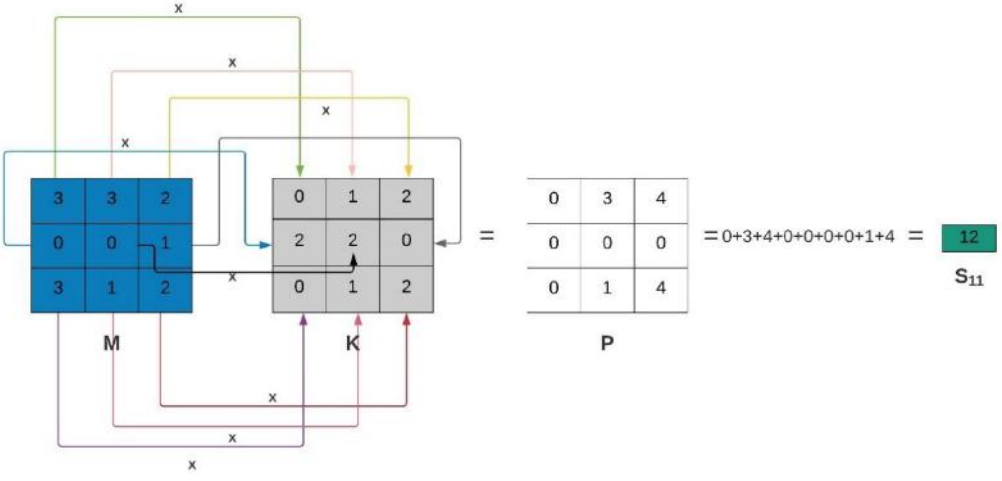
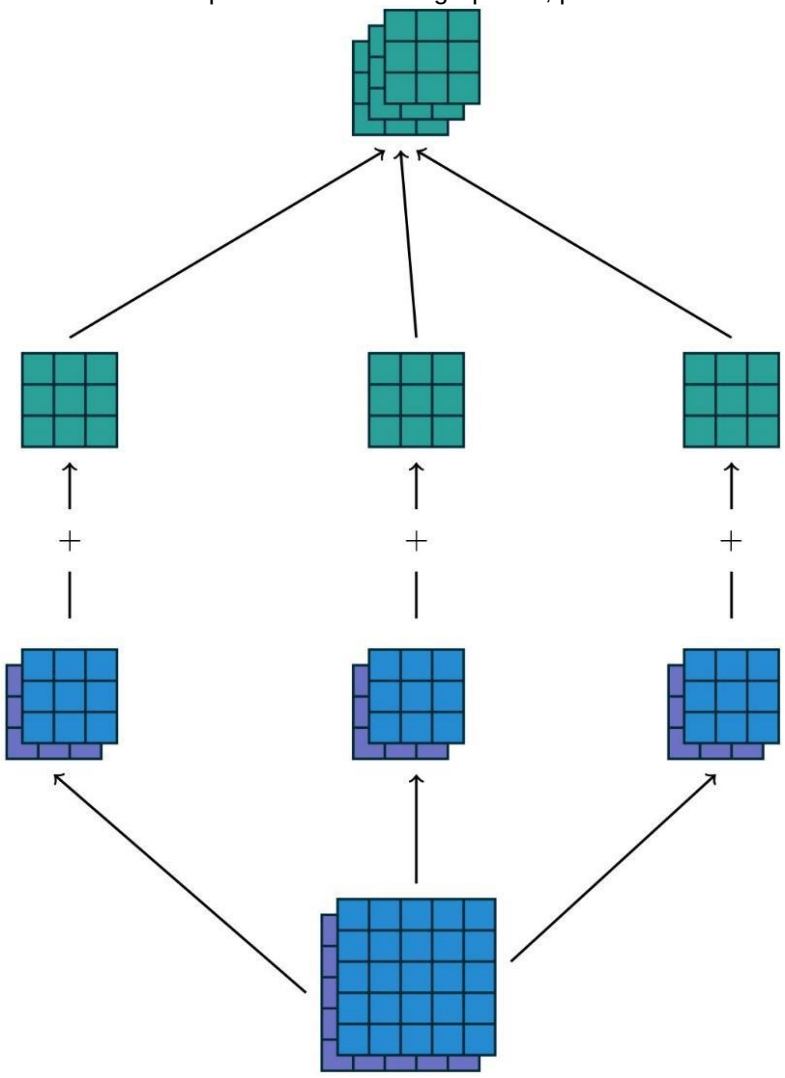


Figura A-4: Mapas de característica de saída resultante de uma operação de convolução. Dois mapas de característica foram utilizados como mapas de entrada (base da imagem), os mapas foram convoluidos utilizando um conjunto de kernels de 3 x 2 x 3 x 3 *w.* No caminho da esquerda, o mapa de entrada 1(roxo, base da imagem) é convoluido com o *kernel* $w_{1,1}$ (matriz 3x3 roxa após a seta), o mapa de entrada 2 (azul, base da imagem) é então convoluido com o *kernel*$w_{1,2}$(matriz 3x3 azul, após a seta) e os resultados são somados, elemento à elemento, para formar o primeiro mapa de característica de saída (matriz 3x3 verde). O mesmo processo é repetido nos outros dois caminhos (centro e direita) e todos os três mapas de saída são agrupados, para formar a saída (topo).



Fonte: Dumoulin e Visin (2016)

O conjunto de kernels que define uma convolução discreta tem o formato correspondente a uma permutação de (*n, m, k*), onde:

- *n* é o número de mapas de características de saída.
- *m* é o número de mapas de características de entrada.
- k_j é o tamanho do kernel ao longo do eixo *j.*

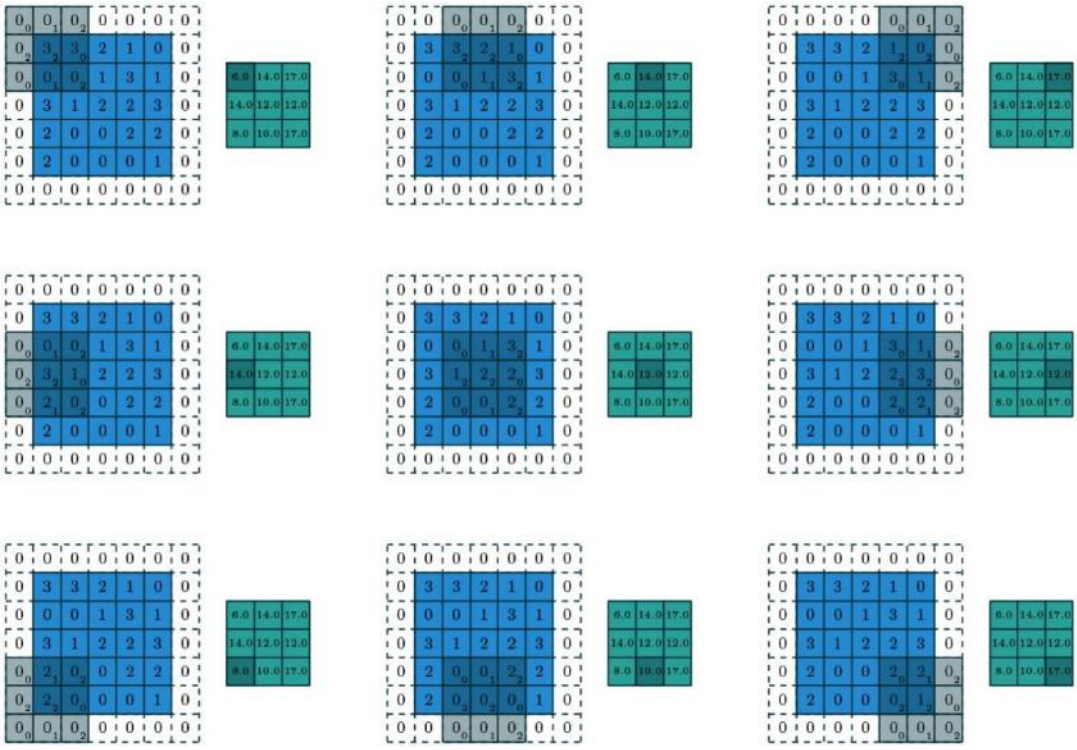
As seguintes propriedades afetam as dimensões da saída de uma camada convolutiva ao longo do eixo *j*:

- i_j dimensões do dado de entrada ao longo do eixo *j*.

- k_j dimensões do *kernel* ao longo do eixo j.
- s_j passo (*stride*) (distância entre duas posições consecutivas do *kernel*) ao longo do eixo *j*.
- p_j preenchimento(*padding*) (número de zeros concatenados no início e no final de um eixo) ao longo do eixo *j*.

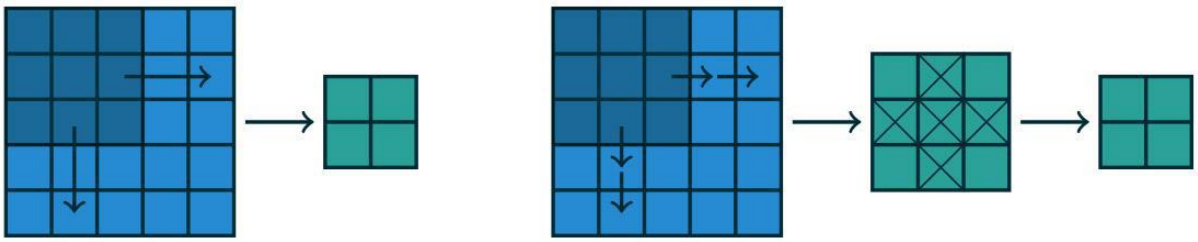
A Figura A-5 mostra um *kernel* 3x3 aplicado à um mapa de entrada 5x5, com o preenchimento com uma borda de zeros 1x1 usando um passo 2x2. Nota-se que o passo utilizado constitui uma forma de subamostragem, além de uma medida do quanto o *kernel* foi transladado. O passo também pode ser interpretado como o quanto do mapa de entrada é mantido. É importante ressaltar que mover o *kernel* com um passo 2 é o mesmo que move-lo com passo igual a 1 e ignorar as saídas dos elementos pares, mantendo só os ímpares (Figura A-6) (DUMOULIN; VISIN, 2016).

Figura A-5: Cálculo dos valores de saída de uma convolução discreta para $n = 2$, $i_1 = i_2 = 5$, $k_1 = k_2 = 3$, $s_1 = s_2 = 2$ e $p_1 = p_2 = 1$.



Fonte: Dumoulin e Visin (2016)

Figura A-6: Forma alternativa de entender os passos. Ao invés de transladar o *kernel* 3x3 com o incremento *s* = 2, o kernel é transladado com um incremento *s* = 1 e apenas 1 elemento, a cada dois incrementos é mantido.



Fonte: Dumoulin e Visin (2016)

A.1 Agrupamento (*Pooling*)

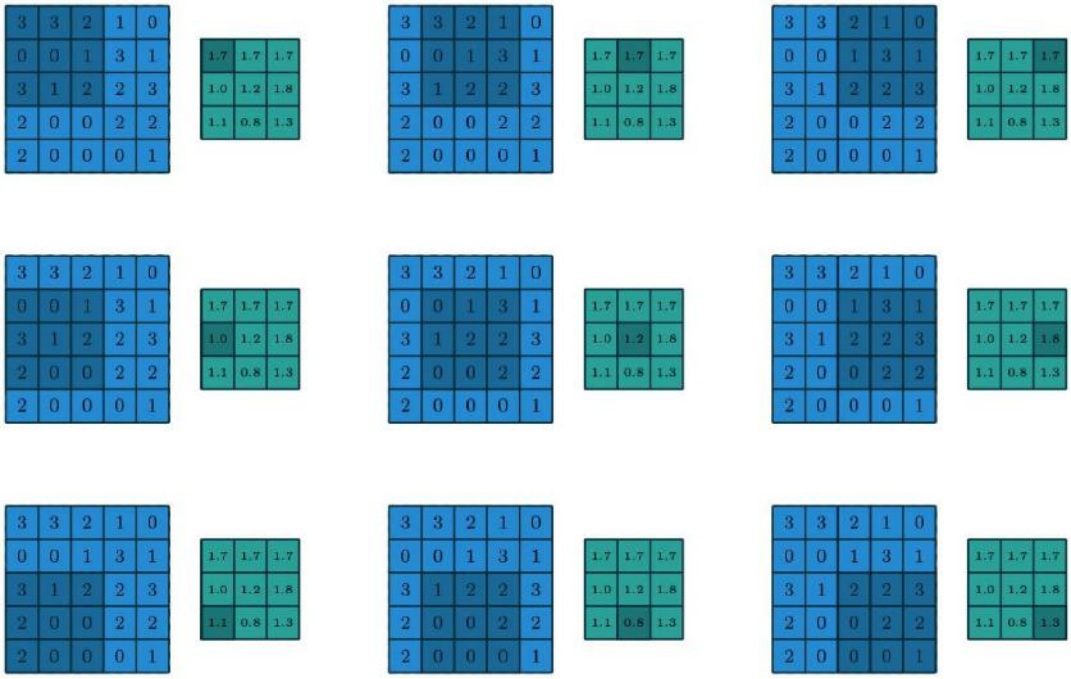
Dumolin e Visin (2016), também atestam que além das operações de convolução as operações de agrupamento, ou *pooling*, são bastante utilizadas na construção de RNPs. O agrupamento reduz o tamanho dos mapas de características ao utilizar uma função, por exemplo a média ou o valor máximo, para resumir as características de determinada área do mapa de característica.

O *pooling* funciona de maneira similar a convolução, desliza-se uma matriz "*janela"* através do dado de entrada, transmitindo seu conteúdo para uma função de agrupamento. A diferença do *pooling* para a convolução discreta é que no *pooling*, a combinação linear realizada pelo *kernel* é substituída por outra função. A Figura A-7 mostra um exemplo de agrupamento por média, a Figura A-8 mostra o agrupamento por valor máximo.

As seguintes propriedades afetam as dimensões da matriz de saída do *pooling*:

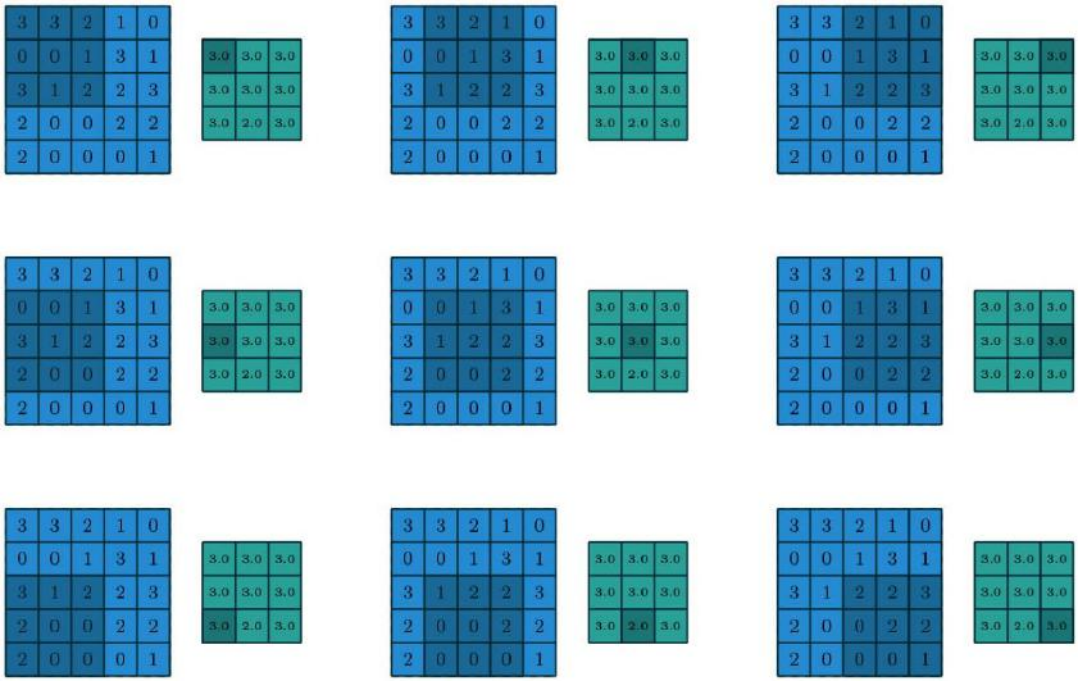
- i_j dimensões do dado de entrada ao longo do eixo *j*.
- k_j dimensões da matriz de agrupamento ao longo do eixo j.
- s_j passo (*stride*) (distância entre duas posições consecutivas da matriz de agrupamento) ao longo do eixo *j*.

Figura A-7: Cálculo dos valores de saída de um agrupamento por média (*average pooling*) em um mapa de entrada 5x5 utilizando um passo(*stride*) 1x1.



Fonte: Dumoulin e Visin (2016)

Figura A-8: Cálculo dos valores de saída de um agrupamento por valor máximo (*max pooling*) em um mapa de entrada 5x5 utilizando um passo(*stride*) 1x1.

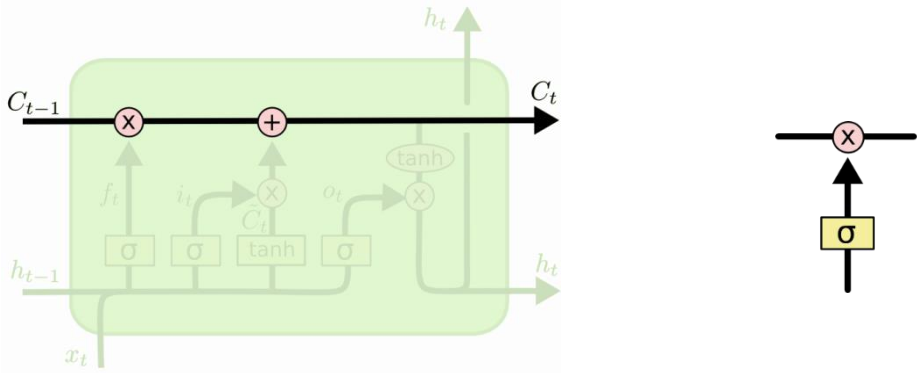


Fonte: Dumoulin e Visin (2016)

Apêndice B –Passo a passo de uma célula LSTM

A célula LSTM possui a habilidade de adicionar ou remover, através de portas, a informação do estado da célula (c*ell state*). O *cell state* (Figura B-1, esquerda) representa o fluxo de informações que passa por dentro de cada unidade oculta, funcionando como uma “esteira” de informações, que liga todas as células da RNN. Uma célula LSTM possui 3 portas, responsáveis por controlar as informações armazenadas no *cell state* (OLAH, 2015; HOCHREITER; SCHMIDHUBER, 1997).

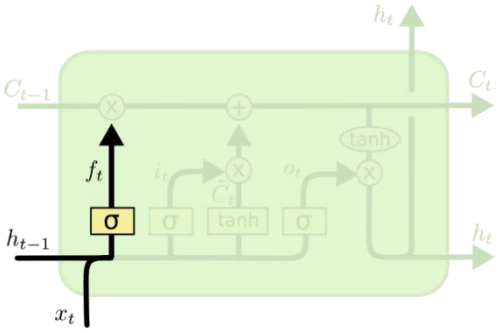
Figura B-1: O diagrama na esquerda mostra o *cell state* C_t, em determinado passo de *tempo t*. O diagrama à direita mostra a estrutura de uma porta em uma célula LSTM. É composta por uma função sigmoide σ e uma operação conhecida como *pointwise product*, definida como $((f.g)(x) = f(x).g(x)$) (WIKIPEDIA CONTRIBUTORS, 2017).



Fonte: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Tomando como exemplo um modelo linguístico que tenta prever a próxima palavra na sentença “Eu moro na Alemanha”, com base em todas as palavras anteriores. Ao fazer uma nova previsão com esse modelo, é necessário decidir que informações, da previsão anterior, vão ser retiradas do *cell state* C_{t-1}. Essa decisão é tomada pela porta conhecida como *forget gate.* Ao analisar as informações contidas em h_{t-1} e X_t, o *forget gate* multiplica a informação em seu interior por 0 ou 1.O número 0 significa deletar, “esquecer”, a informação e o 1, manter, a informação no *cell state* (Figura B-2). No caso do modelo no exemplo, C_{t-1}pode conter características da palavra anterior, “Eu”, como gênero, número, grau e etc. Para que seja possível fazer a previsão da próxima palavra, de forma que os pronomes corretos sejam utilizados, o gênero deve ser esquecido (BEGINNERSGUIDE/OVERVIEW - PYTHON WIKI, 2017; OLAH, 2015).

Figura B-2: Ativação (não ativação) do *forget gate* em um primeiro momento. Onde h_{t-1} é o estado oculto em *t-1*, X_t é a entrada no instante *t* e f_t é a função resultante da aplicação da função sigmóide σ após a multiplicação do peso W_f por h_{t-1} e X_t. f_t só é multiplicada por C_t se possuir valor próximo de 1.O valor contido anteriormente no *forget gate* é denotado pela letra b_f.

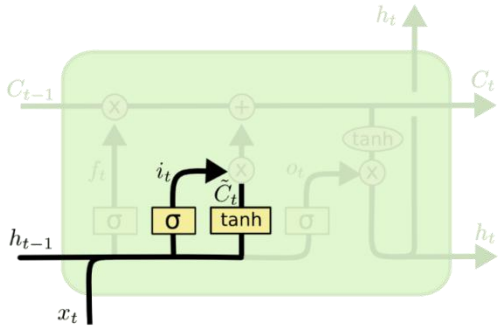


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Fonte: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

O passo seguinte consiste em decidir quais novas informações serão adicionadas ao *cell state*, esse processo é composto de duas etapas. Na primeira, uma porta conhecida como portão de entrada (*input gate*), composto por uma função σ, que decide quais valores serão atualizados, produzindo uma função i_t. Depois uma função *tanh* (tangente hiperbólica) cria um vetor de novos candidatos, $\tilde{C}_t$, que podem (ou não) ser adicionados ao *cell state*. No passo seguinte, os dois são combinados através de um *pointwise product*, para atualizar C_{t-1} (Figura B-3). Voltando ao exemplo, nós queremos adicionar o gênero da nova palavra ao *cell state*, para substituir o que esquecemos no primeiro passo (OLAH, 2015).

Figura B-3: Diagrama a esquerda mostra o funcionamento do *input gate*. A direita, podemos ver a função que faz com que o *input gate* funcione, i_t, e o a função que cria o vetor de valores candidatos, $\tilde{C}_t$. Os valores anteriores de i_t e $\tilde{C}_t$ são denotados, respectivamente pelas letras b_i e b_c.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

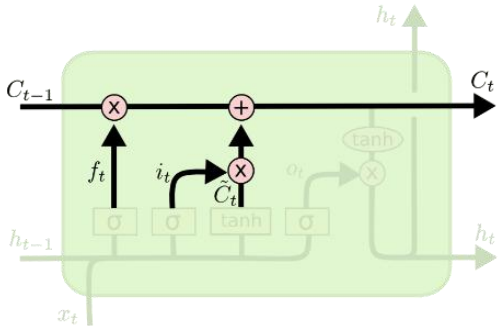
Fonte:https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Como os passos anteriores já decidiram o que fazer é nesse passo em que tudo é posto em prática. Multiplica-se C_{t-1}por f_t, esquecendo o que for necessário esquecer (gênero da palavra anterior, “Eu”), depois multiplicamos i_t *por* $\tilde{C}_t$, que são os valores (no nosso caso, palavras) candidatos escalonados pelo o quanto se quer mudar o valor anterior. No caso do modelo de linguagem é aqui em que a informação sobre o

gênero da palavra anterior é descartada e a nova informação é adicionada, como decidido nos passos anteriores (Figura B-4) (OLAH, 2015).

Finalmente, é preciso decidir qual será a saída. Essa saída é baseada no *cell state*C_{t-1}, porém será uma versão filtrada. Primeiro, ela passa por outra porta, chamada de *output gate*, que aplica à saída uma função sigmoide, ou logística, para decidir que partes de C_{t-1} serão preservadas. Posteriormente o *cell state* filtrado é submetido novamente a uma tanh, que obriga que seus valores estejam entre -1 e 1. Por último, este novo *cell state* é multiplicado por o_t, saída do *output gate*, de forma que a saída h_t será formada apenas pelas informações que as portas deixaram fluir (Figura B-5) (OLAH, 2015).

Figura B-4:Diagrama a esquerda mostra o processo de modificação de C_{t-1} de acordo com a atuação do *forget gate e input gate.* A direita está a regra de atualização do *cell state*.

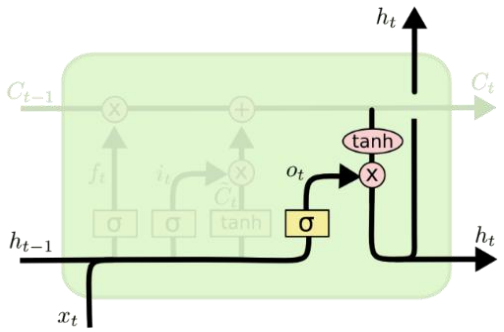


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Fonte: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Fazendo novamente um paralelo com o modelo linguístico, como “Eu” é um pronome, a saída da célula LSTM poderá ser um verbo, no nosso caso “moro”. As informações contidas do *cell state* filtrado pelas portas possuem as informações necessárias para ajudar a escolher o verbo(h_t) com a correta conjugação, e se o mesmo está no plural ou singular, etc (OLAH, 2015).

Figura B-5: Processo final de modificação de C_{t-1} e a disponibilização da saída h_t A direita, estão as funções que descrevem como funciona o *output gate*, o_t e a saída h_t



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Fonte: https://colah.github.io/posts/2015-08-Understanding-LSTMs/