Ministério da Educação
Secretaria de Educação Profissional e Tecnológica
Instituto Federal Catarinense
Câmpus Rio do Sul

FELIPE DE LIMA PERESSIM

PREVISÃO DE INUNDAÇÕES DO RIO ITAJAÍ-AÇU UTILIZANDO REDES NEURAIS PROFUNDAS

Rio do Sul
2019

FELIPE DE LIMA PERESSIM

PREVISÃO DE INUNDAÇÕES DO RIO ITAJAÍ-AÇU UTILIZANDO REDES NEURAIS PROFUNDAS

Trabalho de Conclusão de Curso apresentado ao Curso de graduação em Ciência da Computação do Instituto Federal Catarinense – Câmpus Rio do Sul para obtenção do título de bacharel em Ciência da Computação.
Orientador: Daniel Gomes Soares, Msc.

Rio do Sul
2019

FELIPE DE LIMA PERESSIM

PREVISÃO DE INUNDAÇÕES DO RIO ITAJAÍ-AÇU UTILIZANDO REDES NEURAIS PROFUNDAS

Este Trabalho de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo curso de graduação em Ciência da Computação do Instituto Federal Catarinense – Câmpus Rio do Sul.

Rio do Sul (SC), 29 de Novembro de 2019

Daniel Gomes Soares, Msc.
Instituto Federal Catarinense – Câmpus Rio do Sul

BANCA EXAMINADORA

Juliano Tonizetti Brignoli, Dr.
Instituto Federal Catarinense – Câmpus Rio do Sul

André Alessandro Stein, Msc.
Instituto Federal Catarinense – Câmpus Rio do Sul

Dedico este trabalho primeiramente a Deus, por ter me concedido a oportunidade da vida, por ser essencial em minha vida, autor de meu destino, meu guia, socorro presente na hora da angústia, ao meu pai Ezequiel Peressim e minha mãe Sueli de Lima Peressim, pelo apoio incondicional em todos os momentos difíceis da minha trajetória acadêmica.

RESUMO

As inundações são desastres naturais registrados desde a antiguidade e que remontam à história da humanidade. Entre outros desastres, as inundações são os mais destrutivos, causando danos à vida humana, infraestrutura, agricultura e no sistema socioeconômico. Para mitigar os efeitos causados por este fenômeno, tanto medidas estruturais quanto não estruturais, podem ser adotadas. As estruturais são obras de engenharia custosas e que nem sempre podem resolver totalmente o problema, enquanto que medidas não-estruturais são de baixo custo e auxiliam na prevenção e diminuição dos prejuízos ocasionados pelas inundações. Nesta pesquisa foi desenvolvido um modelo de previsão de inundações em curto prazo, que é uma medida não-estrutural, para isso utilizando-se das Redes Neurais Profundas, pois constituem o estado da arte de modelos de aprendizado de máquina atualmente. O município de Rio do Sul - SC foi o local escolhido como área de estudo devido ao seu extenso histórico de inundações que o acomete. Para alcançar os objetivos deste trabalho, inicialmente realizou-se uma revisão bibliográfica da literatura para identificar as variáveis que poderiam ser utilizadas no modelo, as topologias de RNAs profundas que vem sendo utilizadas para realizar a previsão de inundações em curto prazo. Através da revisão, pode-se observar na literatura que o uso de modelos profundos tem sido infrequente no que tange problemas de hidrologia. Este fato deu forma a pergunta de pesquisa proposta neste trabalho, que consiste em verificar se é possível obter melhores resultados na previsão de inundações através do aprendizado profundo em relação à abordagem tradicional. Neste sentido, para atingir os objetivos da pesquisa, foram modeladas, treinadas e testadas, com diversas configurações, parâmetros iniciais e principalmente variando-se os níveis de profundidade das RNAs adotadas, dentre elas duas vertentes de redes neurais recorrentes - LSTM e GRU; e dois modelos de redes neurais Híbridas, constituídas por camadas de redes Convolucionais, Recorrentes e Multilayer Perceptron. O desempenho das redes modeladas foi analisado por meio de índices estatísticos e gráficos. Os resultados mostram que as redes com maior profundidade foram mais representativas e superiores as redes rasas, consequentemente, a abordagem do aprendizado profundo mostrou que é possível fazer previsão de inundações na área de estudo, com erro e antecedências aceitáveis para ser utilizado como uma medida não-estrutural de um sistema de alerta de cheias.

Palavras-chave: Inundações. Redes Neurais Profundas. Inteligência Artificial.

ABSTRACT

Floods are natural disasters recorded since ancient times that dates back to the history of mankind. Among other disasters, floods are the most destructive, causing damage to human life, infrastructure, agriculture and the socioeconomic system. To mitigate the effects caused by this phenomenon, both structural and non-structural measures can be adopted. Structural measures are expensive engineering works that cannot always fully solve the problem, by other hand, non-structural measures are low cost that can help to prevent and reduce damage due flood. In this research, a short-term flood prediction model was developed, which is a non-structural measure, using Deep Neural Networks which constitute the state of the art of machine learning models nowadays. The township of Rio do Sul - SC was the place chosen as a study area due to its extensive history of flooding that affects it. To accomplish the aims of this research, a literature review was initially performed to identify the variables that could be used in the model and the Deep Neural Networks that have been actually used to forecast floods. Through the review, it were observed in the literature that the use of deep models has been infrequent regarding hydrology problems. This fact raised the research question proposed in this undergradute paper, which aims to verify whether it is possible to obtain better results in flood forecasting through deep learning in relation to the traditional approach. In order to achieve the research objectives, the deep models modeled, were trained and tested with various configurations, initial parameters and mainly by varying the depth levels. Among them two branchs of Recurrent Neural Networks - LSTM and GRU; and two Hybrid neural network models, consisting of Convolutional, Recurrent and Multilayer Perceptron layers. The performance of the networks was analyzed by statistical metrics and graphs. The results show that the deepest models were more representative and superior to the shallow networks, consequently, the deep learning approach showed that it is possible to predict flooding in the study area, with acceptable error and delay, to be used as a non-structural measure of a flood warning system.

Key-words: Floods. Deep Neural Networks. Artificial Intelligence.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

NSE – Coeficiente de Eficiência de Nash e Sutcliffe

DC – Criterion and Deterministic Coefficient

ELM – Extreme Learning Machine

RNA – Rede Neural Artificial

LSTM – Long Short-Term Memory

GRU – Gated Recurrent Unit

MLP – Multilayer Perceptron

RBF – Radial Basis Function

RMSE – Root Mean Square Error

MAPE – Mean Absolute Square Error

SAE – Stack Autoencoders

BPNN – Back Propagation Neural Network

MSE – Mean Squared Errors

SVM – Support Vector Machines

ET_P – error of time to peak discharge

EQ_p – error of peak discharge

PFS – Python Software Foundation

BSD – Berkeley Software Distribution

GAN – Generative Adversarial Network

NTM – Neural Turing Machine

MIT – Massachusetts Institute of Technology

ADAM – Adaptive Moment Estimation

ADAGRAD – Adaptative Gradient Algorithm

SUMÁRIO

1 INTRODUÇÃO

As inundações são desastres naturais registrados desde os tempos antigos e que remontam à história da humanidade. Entre outros desastres, as inundações são os mais destrutivos, causando danos à vida humana, infraestrutura, agricultura e no sistema socioeconômico.

Fenômenos oriundos das inundações fazem parte do ciclo hidrológico natural, entretanto o problema ocorre quando há ocupação inadequada das margens dos rios pela população, seja para fins de habitação, agrícola, comercial, ou uso industrial. De acordo com Tucci (2012) essa ocupação é consequência do crescimento urbano das últimas décadas e por isso tem se refletido no agravamento dos danos provocados por inundações e alagamentos.

Haddad e Teixeira (2013) correlacionam esse crescimento aos sistemas de drenagem urbano que têm promovido mudanças no uso do solo. Por isso, não se consegue drenar o excesso de chuva por conta da impermeabilização e aceleração do escoamento por meio de condutos e canais que levam uma quantidade de água cada vez maior para o sistema de drenagem.

A mitigação dos efeitos das inundações pode ser alcançada tanto através de medidas estruturais quanto não estruturais. Medidas estruturais consistem de trabalhos de engenharia tais como diques, canalização ou reservatórios de inundação, cuja finalidade é atenuar o pico de fluxo da inundação (VAROONCHOTIKUL, 2003). Estas medidas exigem maior investimento e geralmente não são viáveis economicamente.

Em contrapartida, medidas não estruturais utilizam-se de mecanismos de prevenção por meio de alerta, capacitação da população, profissionais qualificados e previsão em curto prazo que têm como objetivo reduzir as perdas econômicas em uma situação de inundação através de medidas que visam prevenir ou conviver com as inundações (TUCCI, 2012).

Modelos de previsão de curto e longo prazo são de importância significativa para a avaliação de risco e gestão de eventos extremos. Previsões com baixo erro contribuem para estratégias de gerenciamento de recursos hídricos, sugestões de políticas e análise, além de modelagem de evacuação adicional (MOSAVI; OZTURK; CHAU, 2018).

Uma abordagem de apoio a tomada de decisão baseada em medidas não estruturais que se tem desenvolvido e ganhado bastante popularidade são os modelos de previsão baseados em Redes Neurais Artificiais (RNAs) (ELSAFI, 2014), que se mostram vantajosos em relação a outros modelos devido a sua considerável capacidade de modelar sistemas não lineares complexos sem o conhecimento de elementos físicos destes sistemas (LIU; XU; YANG,

2017). Evidências de estudos recentes exibem a obtenção de excelentes resultados na previsão de inundações e apontam as RNAs como uma alternativa promissora aos sistemas de previsão hidrológicos tradicionais (SOARES, 2014), (WANG et al., 2017), (LIU; XU; YANG, 2017), (MOSAVI; OZTURK; CHAU, 2018), (SIT; DEMIR, 2019). As RNAs são uma técnica de Inteligência Artificial que se inspira no Neurônio Biológico e o projeta através de modelos matemáticos.

Levando-se em consideração os recentes avanços das RNAs, este trabalho tem por objetivo desenvolver um modelo de previsão de inundações para a cidade de Rio do Sul por meio da técnica de Redes Neurais Artificiais, sobretudo as Redes Neurais Recorrentes, utilizando-se da abordagem de aprendizado profundo (*Deep Learning*). A pesquisa realiza ainda um comparativo entre duas configurações do modelo proposto, utilizando o aprendizado profundo e não utilizando. Essa comparação visa constatar ou não, a vantagem do aprendizado profundo sobre a abordagem tradicional.

1.1 PROBLEMATIZAÇÃO

Shen (2017) salienta que as redes neurais multicamadas de gerações anteriores [1] podem ser configuradas para operar com mais camadas. Ao se aumentar a profundidade deste tipo de rede, porém, aumentam também as dificuldades em seu treinamento. A este problema a literatura dá o nome de desaparecimento dos gradientes (GAMBOA, 2017), caracterizado pela situação em que os pesos utilizados na otimização se tornam muito pequenos conforme propagam pela rede, minando a efetividade do algoritmo de retro propagação em redes neurais profundas.

Tal complicação, de acordo com Längkvist, Karlsson e Loutfi (2014), está relacionada ao desaparecimento dos valores otimizados que, exponencialmente pequenos, ficam muito próximos de zero; por conta disso o sinal de erro não é propagado pela rede, ocasionando pouca melhora nos resultados e uma convergência demasiadamente lenta. Pode-se concluir, então, que acoplar mais do que uma ou duas camadas ocultas se revela inviável nos modelos tradicionais (LIU; XU; YANG, 2017).

Soares (2014) evidenciou o mesmo problema ao realizar a previsão de inundações através do uso de RNA's *Multilayer Perceptron* em sua dissertação. O autor observou que ao uti-

[1] Redes Neurais multicamadas de gerações anteriores são modelos de RNAs anteriores ao advento do conceito de aprendizado profundo e dos algoritmos que surgiram com este.

lizar mais de uma camada oculta na rede seus resultados não mostraram melhoras significativas: apenas o tempo de treinamento aumentou consideravelmente.

Avanços tecnológicos em hardware combinados a melhorias nos algoritmos permitiram contornar tais dificuldades, assim garantindo à abordagem de aprendizado profundo um grande ganho de popularidade na última década (GOODFELLOW; BENGIO; COURVILLE, 2016). Este progresso tem permitido ao aprendizado profundo aprender representações de dados com muitos níveis de abstração através de modelos computacionais que são compostos de múltiplas camadas de processamento (HU et al., 2018).

Porém, mesmo com tais significativos avanços nas redes neurais profundas, outras complicações surgem ao se adotar esta abordagem. De acordo com Shen (2017), a disposição de grandes quantidades de dados é necessária para aprendizado profundo, pois pequenas amostras subutilizam a capacidade de construção de modelos complexos para conjuntos maiores. Inversamente, as vantagens do aprendizado profundo aumentam à medida que o tamanho dos dados aumenta.

Hu et al. (2018) mencionam também que não existem muitos estudos utilizando aprendizado profundo em previsão de inundações, especialmente para grandes conjuntos de séries temporais. O que destaca a afirmação dos autores supracitados - e confere relevância à presente investigação - é que de fato não existem trabalhos acadêmicos no local de estudo selecionado utilizando a abordagem específica aqui proposta. Surge, portanto, a seguinte pergunta de pesquisa: É possível obter melhores resultados na previsão de inundações através do aprendizado profundo em relação à abordagem tradicional?

1.1.1 Solução proposta

A proposta deste trabalho é desenvolver um modelo de previsão de inundações baseado em Redes Neurais Artificiais no local de estudo selecionado, utilizando dados hidrológicos. Nesta investigação, serão comparados dois modelos de RNA - o que utiliza a abordagem de aprendizado profundo e o que não a utiliza. Além disso, serão explorados diversos fatores envolvidos na adequação de uma topologia de RNA a um problema de previsão de inundações, dentre os quais:

1. Obtenção dos dados hidrológicos;
2. Tratamento dos dados;
3. Definição das variáveis de entrada;

4. Acoplamento temporal (atraso de tempo) entre as variáveis de entrada e a variável de saída;
5. Algoritmo de treinamento;
6. Algoritmo de regularização;
7. Algoritmo de otimização;
8. Função de ativação;
9. Número de camadas intermediárias;
10. Número de neurônios nas camadas intermediárias da rede.

1.1.2 Delimitação do escopo

O estudo apresentará a modelagem de Redes Neurais Artificiais para a previsão de inundações do Rio Itajaí Açu com seis horas de antecedência. Serão consideradas duas vertentes das Redes Neurais Recorrentes - LSTM (*Long Short Term Memory*) e GRU (*Gated Recurrent Unit*). O modelo selecionado será aplicado utilizando e não utilizando abordagem de aprendizado profundo, além disso, uma rede híbrida será modelada e utilizada como comparativo para com as redes recorrentes, ficando sua aplicação restrita à área de estudo selecionada, que consiste no município de Rio do Sul - Santa Catarina.

1.1.3 Justificativa

Embora a literatura não dê uma definição clara para o conceito de aprendizado profundo, alguns autores como Shen (2017), por exemplo, o caracterizam pela profundidade obtida pelo empilhamento de múltiplas camadas de neurônios em sistemas combinatórios e cuidadosamente projetados nas redes neurais multicamadas, que atuam diretamente em grandes quantidades de dados brutos. Essa utilização privilegiada em bases de dados volumosas pode ser apontada como a principal característica que distingue as redes que utilizam esta abordagem das que não utilizam.

Längkvist, Karlsson e Loutfi (2014) explicam que tal característica tem permitido aumentar a capacidade das redes neurais profundas de modelar estruturas complexas nos dados. Shen (2017) comenta que esta capacidade se estende a funções com alta complexidade de dependências espaço-temporais e distribuições de grandes quantidades de dados, advindas das estruturas formadas pelas redes de grande profundidade.

A capacidade aumentada tem proporcionado às redes que utilizam esta abordagem aprender caracteristícas dos dados através de representações que são expressas através de outras mais simples, ou seja, o aprendizado é realizado através de várias etapas (GOODFELLOW; BENGIO; COURVILLE, 2016), levando a uma maior compactação da rede neural, que assim passa a generalizar melhor (BENGIO, 2009).

Embora nos últimos anos esta se tenha tornado uma técnica bem sucedida e amplamente utilizada na resolução de problemas de inteligência articial considerados difíceis e complexos (BENGIO, 2009), o foco das pesquisas tem sido no desenvolvimento de modelos para dados estáticos e não tanto em dados de séries temporais (LÄNGKVIST; KARLSSON; LOUTFI, 2014).

O fato da pouca utilização da técnica no contexto das séries temporais é comprovado pelas revisões da literatura - Hu et al. (2018) mencionam efetivamente que não existem muitos estudos utilizando esta abordagem em previsão de inundações. Shen (2017) aponta, igualmente, que a aplicação da abordagem do aprendizado profundo tem sido infrequente na hidrologia, especialmente em configurações que abrangem grandes quantidades de dados.

Levando em consideração as afirmações dos autores citados, pretende-se através da presente investigação contribuir para a literatura por meio da modelagem de um modelo de RNA para previsão de inundações e de um comparativo de tal modelo em vários níveis de profundidade.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Desenvolver um modelo de previsão de inundações para a cidade de Rio do Sul por meio da técnica de Redes Neurais Artificiais e realizar um comparativo do modelo em vários níveis de profundidade.

1.2.2 Objetivos específicos

1. Investigar quais modelos de aprendizado profundo têm sido utilizados na previsão de inundações;
2. Definir as principais variáveis que possam auxiliar na previsão de inundações;
3. Obter conjuntos de dados hidrológicos para o treinamento, teste e verificação dos modelos;

4. Modelar e treinar as Redes Neurais em diversos níveis de profundidade para previsão de inundações no local de estudo selecionado;

5. Comparar as previsões feitas pelos modelos com dados de verificação.

1.3 METODOLOGIA

1. Revisão bibliográfica: Esta etapa objetiva o aprofundamento teórico acerca de previsão de inundações, Redes Neurais Artificiais e o conceito de Aprendizado profundo. A pesquisa bibliográfica foi realizada em livros, teses, dissertações, monografias e artigos de periódicos;

2. Estudo da previsão de inundações: Com base na revisão bibliográfica foram identificadas as principais variáveis para previsão de inundações, bem como o horizonte de previsão mais adequado.

3. Tratamento dos dados: Com os dados obtidos foram realizados uma separação entre os dados relativos às variáveis identificadas, gerando assim um novo conjunto de dados e por conseguinte realizou-se a normalização dos dados através de técnicas de estatística;

4. Modelagem das Redes Neurais Artificiais: Nesta etapa, são modeladas e treinadas diferentes configurações de RNAs para realizar a previsão de inundações na área de interesse.

5. Comparação dos modelos: Avaliar e comparar, através de métricas de erro, o desempenho dos modelos desenvolvidos a fim constatar o modelo de RNAs com menor erro e portanto verificar se o aprendizado profundo oferece vantagens significativas sobre a abordagem tradicional e também para selecionar um modelo para realizar as previsões do local de estudo selecionado. Elaborar gráficos para avaliar e comparar o desempenho dos modelos desenvolvidos, a fim de constatar qual dos modelos de RNAs melhor se aproxima dos dados reais na previsão.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 INUNDAÇÕES

As inundações são desastres naturais que sobrevêm quando há o extravasamento da calha dos rios devido à incapacidade de transporte das águas, que acabam por ocupar áreas povoadas próximas às margens dos rios (TUCCI; BERTONI, 2003).

Em função da frequência do fenômeno, Goerl e Kobiyama (2005) comentam que a terminologia utilizada para conceituá-lo é bastante variada, com termos como cheia, enchente, enxurrada, inundação gradual, inundação brusca, alagamentos, inundações ribeirinhas, inundações urbanas e enchentes repentinas sendo empregados de acordo com os locais das ocorrências.

Sobre a diversidade dos termos, Lohmann (2011) explica que geralmente são empregados incorretamente como sinônimos, enquanto Goerl e Kobiyama (2005) apontam que o uso inconsistente é feito em virtude de traduções equivocadas e adaptações malfeitas de termos provenientes de línguas estrangeiras, principalmente do inglês e do espanhol.

De acordo com , Goerl e Kobiyama (2005) as palavras cheia e enchente se originam do verbo encher, do latim *implere*, que significa "ocupar o vão, a capacidade ou a superfície de" ou "tornar-se cheio ou repleto". Portanto, quando as águas do rio atingem sua capacidade máxima e alcançam as margens, contudo sem transbordar nas áreas adjacentes, diz-se que ocorreu uma enchente. Doravante as águas transbordam e então ocorre a inundação, conforme ilustra a Figura 1.

Após um transbordamento decorrem diversos tipos de inundações (GOERL; KOBIYAMA, 2005). Tucci e Bertoni (2003), por exemplo, utilizam o termo inundação ribeirinha para caracterizar a inundação causada por um excesso de precipitação que não consegue ser drenada pelo solo e acaba ocupando a várzea.

No Manual de Desastres Naturais publicado em 2003 pelo Ministério da Integração Nacional, Castro (2003) define as inundações como um transbordamento de água proveniente de rios, lagos e açudes e as classifica em função de sua magnitude e evolução. Em relação à magnitude as inundações são classificadas, através de dados comparativos de longo prazo, como:

- inundações excepcionais;
- inundações de grande magnitude;

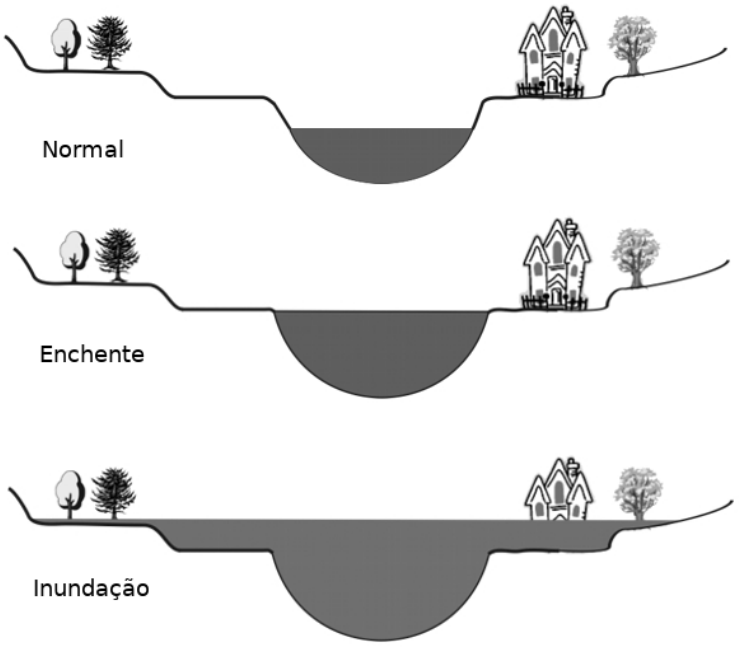
- inundações normais ou regulares;
- inundações de pequena magnitude.

Tomando-se sua evolução como critério, serão classificadas como:

- enchentes ou inundações graduais;
- enxurradas ou inundações bruscas;
- alagamentos;
- inundações litorâneas provocadas pela brusca invasão do mar

Dentre as supracitadas Kron (2002) cimenta que as mais comuns são as inundações graduais e bruscas como sendo as mais comuns.

Figura 1 – Elevação do nível de um rio provocada pelas chuvas, do nível normal até a ocorrência de uma inundação.



Fonte – Goerl e Kobiyama (2005)

2.1.1 Inundações graduais

De acordo com Goerl e Kobiyama (2005) as inundações graduais são caracterizadas pela elevação do nível das águas e consequentemente o transbordamento, que ocorrem paulatinamente e conforme complementa Castro (2003), com previsibilidade mantendo-se em situação de cheia durante algum tempo, em seguida escoando-se gradualmente

Castro (2003) explica que as inundações graduais são cíclicas e exibem aspectos de sazonalidade. Exemplo típico de periodicidade ocorre nas inundações anuais da bacia do rio Amazonas. No decorrer de quase uma centena de anos de observação e registro, caracterizou-se que, na cidade de Manaus, na maioria dos anos, o pico das cheias ocorre no mês de Junho.

Lohmann (2011) salienta que este fenômeno é exacerbado por variáveis climatológicas de médio e longo prazos e pouco influenciáveis por variações diárias do tempo. Kron (2002) complementa que essa exacerbação ocorre devido a períodos de chuvas prolongados e persistentes, e portanto Lohmann (2011) conclui que a inundação gradual caracteriza-se por sua abrangência e grande extensão.

Em relação a definição dos termos associados Castro (2003) utiliza o termo "Inundações graduais ou Enchentes"; Tucci e Bertoni (2003) "Inundações Ribeirinhas"; em contrapartida Kron (2002) emprega o termo "Inundação" - termo mais comumente utilizado na literatura.

2.1.2 Inundações bruscas

As inundações bruscas, também denominadas "enxurradas" são, segundo análise de Mendiondo (2005), eventos de curta duração caracterizados por precipitações intensas e concentradas que advêm repentinamente e, por geralmente ocorrerem em regiões de relevo acidentado, conforme explica Castro (2003), acabam por produzir violentas elevações dos caudais que escoam de forma rápida e intensa, causando um desequilíbrio entre o leito do rio e o volume caudal e assim produzindo um transbordamento.

Goerl e Kobiyama (2005) destacam que devido a seu desenvolvimento repentino tal fenômeno ocasiona uma escassez de tempo para realizar alertas nos locais de ocorrência. Por outro lado, a imprevisibilidade e violência características deste tipo de desastre, conforme Castro (2003), é o que acaba dificultando a tomada de medidas de proteção pelas populações afetadas.

Castro (2003) ainda complementa que áreas com relevo acidentado favorecem o escoamento das águas, que acabam ganhando força e velocidade na descida e por isso contribuem para intensificar a torrente e causar danos. Esse fenômeno costuma surpreender por sua violência e menor previsibilidade, exigindo uma monitoração complexa.

Kron (2002), em contrapartida, explica que as inundações bruscas não estão associadas apenas ao rápido fluxo de água em regiões de relevo acidentado, mas também às inun-

dações de locais planos. Por causa da intensa urbanização ocorrida principalmente nas últimas décadas, cidades de médio e grande porte, independente da inclinação de seus territórios, têm apresentado locais de ocorrências de inundações com maior velocidade.

Para definir o que aqui definimos como inundação brusca, Castro (2003) utiliza os termos "Inundação brusca ou Enxurrada"; Mendiondo (2005) e Kron (2002) ficam apenas com "Enxurrada"; Goerl e Kobiyama (2005) preferem "Inundação brusca".

As causas associadas a tal fenômeno expressam as principias variáveis que configuram o problema em diversos aspectos. Desta forma sua compreensão faz-se indispensável para este trabalho e portanto tais causas são elucidadas na subseção seguir.

2.1.3 Causas das inundações

Fenômenos oriundos das inundações fazem parte do ciclo hidrológico natural, podendo acontecer repentinamente ou lentamente, porém em ambos os casos a configuração das ocorrências depende de fatores naturais e humanos. Fatores naturais são: distribuição de precipitação, morfometria [1], fisiografia da bacia hidrográfica (PINZÓN; RESTREPO; CALDERÓN, 2015), degelo (KRON, 2002), escoamento da água em regiões com relevo acidentado (CASTRO, 2003), entre outros.

Já as variáveis humanas envolvidas são, como explica Kron (2002), um conjunto de desenvolvimentos que impactam os riscos de inundação e aumentam sua frequência e magnitude, sendo que a atuação humana reveste-se de acentuada importância mesmo quando complementa alterações climáticas.

Geralmente o problema ocorre quando há ocupação inadequada das margens dos rios pela população, seja para fins habitacionais, agrícolas, comerciais ou industriais. Tucci (2012) corrobora a tese de que a maior ocupação de áreas inundáveis é consequência do crescimento urbano das últimas décadas e tem se refletido no agravamento dos danos provocados por inundações e alagamentos.

Para Bonafé (1989), a urbanização relaciona-se com os fenômenos aqui abordados de maneiras diversas, entre elas:

- Escoamento superficial ocasionado pela crescente impermeabilização da bacia hidrográfica, resultante da substituição de áreas verdes por asfalto.

[1] Medida das formas físicas e dos fenômenos terrestres - como latitude, altitude, etc (DICIO - DICIONÁRIO ONLINE DE PORTUGUÊS, 2019).

- Erosão do solo, modificando as condições naturais do escoamento superficial e gerando assoreamento nos cursos dos rios.
- Construções de obras hidráulicas nos cursos d'água ocasionam alterações no regime do rio tanto para jusante - devido à retificações e canalizações - como para montante - por conta das barragens.

Haddad e Teixeira (2013) correlacionam essas implicações aos sistemas de drenagem urbano que têm promovido mudanças no uso do solo. Não se consegue drenar o excesso de chuva por conta da impermeabilização e aceleração do escoamento por meio de condutos e canais que levam uma quantidade de água cada vez maior para o sistema de drenagem.

A capacidade limitada de infiltração do solo é o motivo apontado por Tucci e Bertoni (2003) para um maior acumulo de água que, somado à precipitação intensa e persistente, esgota a capacidade de escoamento do sistema de drenagem. O resultado é que o excesso de volume de água que não pode ser drenada acaba ocupando a várzea e ocasionando inundação.

Kron (2002) argumenta, ainda, que a inundação por toda a extensão de um grande rio não é causada apenas por áreas urbanas impermeáveis, mas sim em conjunto com áreas que apresentam uma impermeabilidade natural do solo. A "impermeabilidade natural" é causada por chuvas persistentes que ocorrem com pequenos intervalos de diferença e consomem a capacidade de armazenamento do solo.

Pode-se concluir, por fim, que a inundação é um fenômeno natural que ocorre sempre que a vazão a ser escoada é superior a capacidade de descarga da calha do curso de água de um rio. O problema ocorre quando há interferência nos regimes funcionais da natureza - devido a fatores humanos o fenômeno deixa de ser natural, passando a ser caracterizado como desastre capaz de causar danos e impactos associados a anormalidades no ciclo hidrológico.

2.1.4 Impactos das inundações

A inundação é um fenômeno global cujo impacto causa grande devastação, prejuízos socioeconômicos, alterações na saúde e no deslocamento populacional, além de perdas e interrupção do fornecimento de alimentos e impactos ambientais.

Sobre os impactos ambientais, Gautam e Hoek (2003) comentam que estes podem ser vastos, desde a dispersão de resíduos domésticos de baixo nível do sistema fluvial até a contaminação do abastecimento de água e vida silvestre da comunidade, contaminando habitats com substâncias extremamente tóxicas, além das alterações na fauna e na flora ao longo do

curso do rio que muitas vezes são ocasionadas por processos de erosão e deposição decorrentes dos eventos catastróficos de inundação.

Os autores supracitados enfatizam que após ocorrer uma inundação muitos outros impactos ambientais podem surgir, como aumento no volume de detritos a ser coletado, danos aos serviços públicos de abastecimento de água e operações de esgoto e a entrada de poluentes agrícolas nos sistemas fluviais.

Em função dos impactos na saúde e de acordo com Few et al. (2004), surtos de doenças infecciosas e efeitos adversos na saúde mental são mais prováveis de ocorrer a médio e longo prazo, dentre os quais os autores destacam os seguintes: doença diarreica fecal-oral; infecção por helmintos transmitidos pelo solo; doenças transmitidas por roedores - leptospirose por exemplo; e mortes resultantes de tais doenças e de outros fatores ocasionados pelas inundações.

Devido a tais impactos, mortes e ferimentos são inevitáveis e geralmente apresentam maior probabilidade de ocorrência durante o período de início das cheias, por afogamento ou por ferimentos fatais sofridos quando atingidos por destroços em águas com grandes correntezas. Lesões também podem resultar das operações de evacuação e limpeza. Eventos de inundação de longa duração aumentam o risco de mortes por afogamento, porém podem ser atenuadas através de medidas de prevenção (FEW et al., 2004).

2.1.5 Medidas de Prevenção

A mitigação dos efeitos das inundações pode ser alcançada tanto através de medidas estruturais quanto não estruturais. Medidas estruturais consistem de trabalhos de engenharia tais como diques, canalização ou reservatórios de inundação, cuja finalidade é atenuar o pico de fluxo da inundação (VAROONCHOTIKUL, 2003).

Tucci e Bertoni (2003) caracterizam como medidas estruturais aquelas que modificam o sistema fluvial para evitar os prejuízos causados pelas cheias. Conforme expõe Bonafé (1989), trata-se de grandes obras hidráulicas cujo objetivo é reter, confinar ou escoar com maior rapidez e menores cotas o volume das cheias.

Tucci e Bertoni (2003) classificam estas medidas em extensivas ou intensivas. As extensivas exercem influência direta na bacia, modificando as relações entre precipitação e vazão, por exemplo, na alteração da cobertura vegetal do solo que diminui e atrasa os picos de cheia e controla a erosão da bacia. As medidas intensivas são aquelas que exercem influência

direta no rio e podem ser de três tipos:

1. Medidas que aceleram o escoamento: construção de diques e polders, aumento da capacidade de descarga dos rios e corte de meandros;
2. medidas que retardam o escoamento: reservatórios e bacias de amortecimento;
3. medidas de desvio do escoamento: construção de canais de desvios.

Lohmann (2011) explica que estas medidas são fundamentais e indispensáveis para resolver grande parte dos problemas advindos das inundações, mas que geralmente abrangem uma grande área de influência e consequentemente exigem maiores investimentos. Portanto, muitas vezes não são viáveis economicamente e não representam por si só soluções eficazes e sustentáveis dos problemas mais complexos de drenagem urbana.

Tucci e Bertoni (2003) complementam que as medidas estruturais podem não oferecer proteção completa e ainda assim criar uma falsa sensação de segurança, permitindo a ampliação da ocupação das áreas inundáveis, o que futuramente pode resultar em danos significativos.

Em contrapartida, medidas não estruturais se utilizam de mecanismos de prevenção por meio de alerta, capacitação da população, profissionais qualificados e previsão em curto prazo, tendo como objetivo reduzir as perdas econômicas e de vidas em uma situação de inundação através de providências que visem prevenir ou ajudar a conviver com as inundações (TUCCI, 2012). Mas, apesar de sua característica potencialmente preventiva, essas soluções também necessitam da alocação de partes significativas dos recursos exigidos pela execução daquelas obras chamadas estruturais (BONAFÉ, 1989).

Modelos de previsão de curto e longo prazo, ainda assim, são de importância crucial para a avaliação de risco e gestão de eventos extremos. Previsões com baixo erro contribuem para estratégias de gerenciamento de recursos hídricos, sugestões de políticas e análise, além de modelagem de evacuação adicional (MOSAVI; OZTURK; CHAU, 2018).

Uma abordagem de apoio a tomada de decisão baseada em medidas não estruturais que se tem desenvolvido e ganhado bastante popularidade são os modelos de previsão baseados em Redes Neurais Artificiais (ELSAFI, 2014), que se mostram vantajosos em relação a outros modelos devido a sua considerável capacidade de modelar sistemas não lineares complexos sem o conhecimento de elementos físicos destes sistemas (LIU; XU; YANG, 2017).

Neste sentido o presente trabalho objetiva se utilizar de medidas não estruturais tais como modelos de previsão de inundações, especificamente por meio da técnica de Redes Neurais Artificiais, para realizar a previsão de inundações do Rio Itajaí Açu para a cidade de Rio do Sul.

2.2 REDES NEURAIS ARTIFICIAIS

A Rede Neural Artificial é um mecanismo que é projetado para modelar a maneira como o cérebro realiza uma determinada tarefa ou função de interesse; geralmente é implementada utilizando componentes eletrônicos ou é simulada por programação em um computador digital (HAYKIN, 2009).

Braga, Carvalho e Ludermir (2011) conceituam Redes Neurais Artificiais como sistemas de processamento paralelo compostos por unidades de processamento simples que simulam neurônios artificiais e calculam determinadas funções matemáticas geralmente não lineares. Haykin (2009) complementa que tais unidades são naturalmente propensas a armazenar conhecimento adquirido através da experiência e torná-lo disponível para uso; consequentemente assemelhando-se ao cérebro humano em dois aspectos:

1. O conhecimento é adquirido pela RNA a partir de seu ambiente através de um processo de aprendizado.
2. As forças de conexão interna, conhecidas como pesos sinápticos, são usadas para armazenar o conhecimento adquirido.

A capacidade de aquisição de conhecimento permite as RNAs através de conjuntos de dados, mesmo que as vezes reduzidos, generalizar a informação e dar respostas coerentes para dados não conhecidos e extrair informações não apresentadas de forma explícita, podendo ir muito além do que simplesmente mapear relações de entrada e saída, fato este que caracteriza a importância das Redes Neurais Artificiais e as qualifica como mapeadores universais de funções multivariáveis (BRAGA; CARVALHO; LUDERMIR, 2011).

Essa capacidade é proporcionada pela mimetização do comportamento e da estrutura do cérebro humano, os quais conferem as RNAs certas características exclusivas dos sistemas biológicos (FINOCCHIO, 2014) e exibem algumas propriedades fundamentais que as diferenciam em relação a outros sistemas de aprendizado de máquina, dentre as destacadas por Haykin (2009) são:

a) Não-Linearidade: Um neurônio pode ser linear ou não-linear. Uma rede neural construída a partir de uma conexão entre neurônios não lineares é, igualmente não-linear. A não-linearidade é uma propriedade relevante, particularmente se o quando o sistema físico responsável for inerentemente não linear.

b) Mapeamento de entrada e saída: A rede é representada através de valores de ponderação, também denominados pesos sinápticos, que modificados minimizam a diferença entre a resposta desejada e a resposta real produzida por um sinal de entrada da rede de acordo com um critério estatístico apropriado. Esse processo acontece diversas vezes dentro de um ciclo até que a rede chegue a um estado estacionário onde as mudanças nos pesos sinápticos tornam-se insignificantes.

c) Adaptabilidade: As redes neurais têm uma capacidade integrada de adaptar seus pesos sinápticos às mudanças no ambiente circundante. Neste sentido tornam-se ideias para problemas de inundações, pois conforme as condições climáticas vão mudando com o tempo, a rede pode ser realimentada com dados e os pesos ajustados para os novos dados e portanto mantendo-se a confiabilidade da rede em suas previsões.

d) Resposta evidencial: No contexto da classificação de padrões, uma rede neural pode ser projetada para fornecer informações não apenas sobre qual padrão específico selecionar, mas também sobre a confiança na decisão tomada.

e) Informação contextual: O conhecimento é representado pela própria estrutura e estado de ativação de uma rede neural.

f) Analogia Neurobiológica: O estudo de uma rede neural é motivado pela analogia com o cérebro, que é a prova viva de que o processamento paralelo tolerante a falhas não é apenas fisicamente possível, mas também rápido e poderoso.

Essas propriedades conferem as Redes Neurais Artificiais a habilidade de aprendizado sem a necessidade de uma programação explícita. Desta forma assemelhando-se aos sistemas biológicos que consistem de células nervosas muito simples, porém numerosas, que trabalham maciçamente em paralelo e possuem a capacidade de reter conhecimento através do processo de aprendizado representado por sua estrutura (KRIESEL, 2007).

Neste sentido torna-se necessário que as propriedades biológicas das Redes Neurais Biológicas sejam consideradas a partir do ponto de vista do processamento de informações e que

alguns conhecimentos elementares destas sejam fundamentados para uma melhor compreensão das Redes Neurais Artificias.

2.2.1 Neurônio biológico

O córtex cerebral humano é a maior e melhor região desenvolvida do cérebro humano. Contendo aproximadamente 100 bilhões de neurônios, ligados por cordões nervosos (axônios) que se ramificam e terminam em sinapses. Essas sinapses são as conexões com outros neurônios. As sinapses conectam-se a dendritos, extensões ramificadas do corpo da célula neural projetadas para receber a entrada de outros neurônios na forma de sinais elétricos (MEHLIG, 2019). Essa complexa conexão entre os neurônios formam uma grande rede, chamada Rede Neural Biológica (FINOCCHIO, 2014).

Um neurônio pode ser descrito de forma simplificada através de quatro componentes principais que o compõe, ilustrados na Figura 2 e que conforme Finocchio (2014) são:

1. Dendritos - são prolongamentos geralmente ramificados. Eles são incumbidos pelo recebimento dos sinais de estímulos transmitidos pelos outros neurônios;

2. Corpo celular - é a região onde situa-se o núcleo e a maioria das estruturas citoplasmáticas. Esta região é responsável por coletar e combinar informações vindas de outros neurônios;

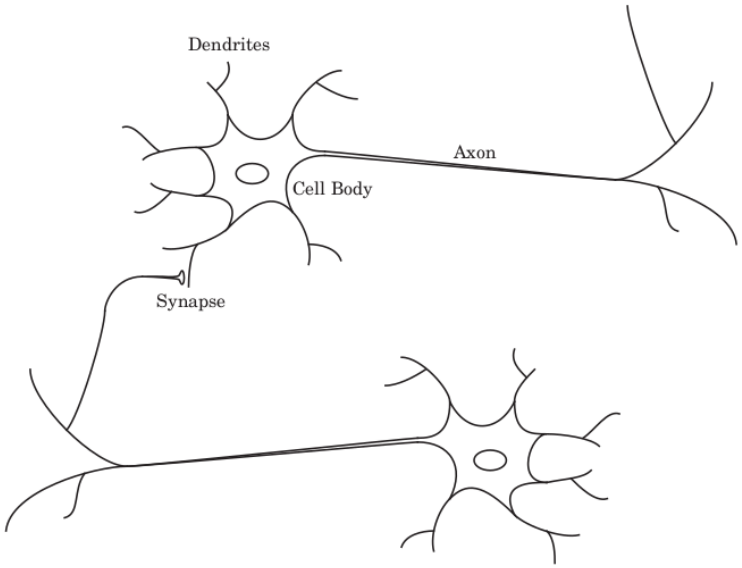
3. Axônio - responsável por transmitir os estímulos para outras células;

4. Sinapses: são responsáveis por transmitir as informações de uma célula para outra célula.

Os dendritos são redes receptoras, semelhantes a árvores, cujas ramificações são as fibras nervosas que transportam sinais elétricos para o corpo celular. O corpo da célula efetivamente soma e limita os sinais de entrada. O axônio é uma única fibra longa que transporta o sinal do corpo celular para outros neurônios. O ponto de contato entre um axônio de uma célula e um dendrito de outra célula é chamada de sinapse (HAGAN et al., 2014).

A sinapse é localizada entre a terminação axônica de um neurônio e o dendrito de outro. Através da sinapse os neurônios se unem funcionalmente e são determinadas por um processo químico complexo, que estabelece a função da rede neural (BRAGA; CARVALHO; LUDERMIR, 2011).

Figura 2 – Ilustração sistemática de Neurônios Biológicos.



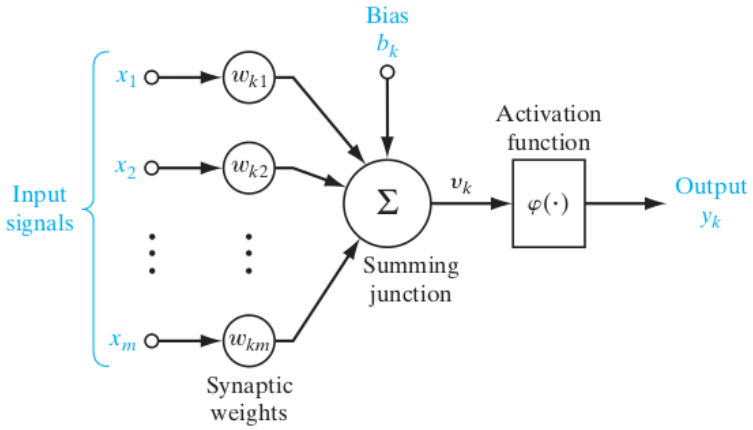
Fonte – Hagan et al. (2014).

2.2.2 Neurônio artificial

A analogia neurobiológica é a fonte de inspiração das redes neurais artificiais, que modeladas a partir da combinação de muitas unidades de processamento individuais (neurônios artificiais), objetivam assemelhar-se com os neurônios em sistemas biológicos vivos, porém possuem menos conexões e são relativamente menores (BROOKSHEAR, 2008). Neste sentido Barreto (2002) expõe que, faz-se necessário compreender que as semelhanças são mínimas, e conforme destaca Haykin (2009) são muito primitivas quando comparadas as redes neurais biológicas.

De acordo com Raschka e Mirjalili (2017), Warren McCullock e Walter Pitts publicaram o primeiro conceito de uma célula cerebral simplificada, o chamado neurônio McCullock-Pitts (MCP), em 1943. Braga, Carvalho e Ludermir (2011) complementam que esse modelo é uma simplificação do que se sabia na época sobre o neurônio biológico. Posteriormente na década de 1950, de acordo com Nielsen (2019), inspirado por esse modelo o cientista Frank Rosenblatt desenvolveu o modelo de neurônio artificial conhecido como Percepton (Figura 3).

Figura 3 – Representação do Neurônio Artificial.



Fonte – Haykin (2009).

De acordo Nielsen (2019), o Percepton de Rosenblatt pode tomar muitos valores binários como entradas, $x_1, x_2, \ldots$ e produzir um único valor binário de saída. Para calcular o valor de saída Rosenblatt propôs que pesos $w_{k1}, w_{k2}, \ldots$ fossem introduzidos para ponderar os valores de entrada de modo a expressar a importância entre os respectivos valores de entrada para a saída. Desta forma o valor de saída, 0 ou 1, é determinado se a soma ponderada $\sum_j w_j X_j$ é menor ou maior que um valor limiar.

A partir deste modelo de neurônio proposto por Rosenblatt, ilustrado na Figura 3, Haykin (2009) identifica três elementos básicos:

1. Um conjunto de sinapses cada um dos quais é caracterizado por um peso. Especificamente, um sinal x_j na entrada da sinapse j conectado ao neurônio k é multiplicado pelo peso sináptico w_{kj};

2. Um somador para somar os sinais de entrada, ponderados pelas respectivas forças sinápticas do neurônio, através de uma combinação linear;

3. Uma função de ativação para limitar a amplitude do valor de saída de um neurônio. A função de ativação limita a faixa de amplitude permitida pelo sinal de saída para algum valor finito dentro de um intervalo específico.

Além destes elementos básicos, o neurônio artificial da Figura 3 também inclui um limiar de ativação (bias) aplicado externamente, denotado por b_k . Haykin (2009) explica que este limiar tem o efeito de aumentar ou diminuir os valores entrada de entrada da função

de ativação, dependendo se é positivo ou negativo, respectivamente, assim caracterizando uma transformação afim.

Matematicamente o neurônio da Figura 3 de acordo com Haykin (2009) pode ser representado através das seguintes equações:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

$$y_k = \varphi\left(u_k + b_k\right) \tag{2.2}$$

Onde $x_1, x_2, \ldots, x_m$ são os sinais de entrada; $w_{k1}, w_{k2}, \ldots, w_{km}$ os respectivos pesos sinápticos do neurônio k ; u_k é o resultado da combinação linear dos sinais de entrada com os pesos; b_k é o limiar de ativação; $\varphi(.)$ a função de ativação; e y_k o sinal de saída do neurônio.

2.2.3 Funções de ativação

A escolha das funções de ativação é de acordo com Bishop (2006) determinada pela natureza dos dados e pela distribuição presumida das variáveis-alvo e segue a mesma consideração que os modelos lineares. O autor ainda explica que para problemas genéricos de regressão, a função identidade $y_k = a_K$ é utilizada como ativação.

2.2.3.1 Função degrau

Kovács (2006) explica que na função identidade a monotonicidade é preservada, no entanto o fenômeno da saturação é desprezado. Já na função utilizada no percepton de Rosenblatt - a função degrau - a saturação é preservada.

De acordo com Haykin (2009) a função de ativação degrau é definida pela por:

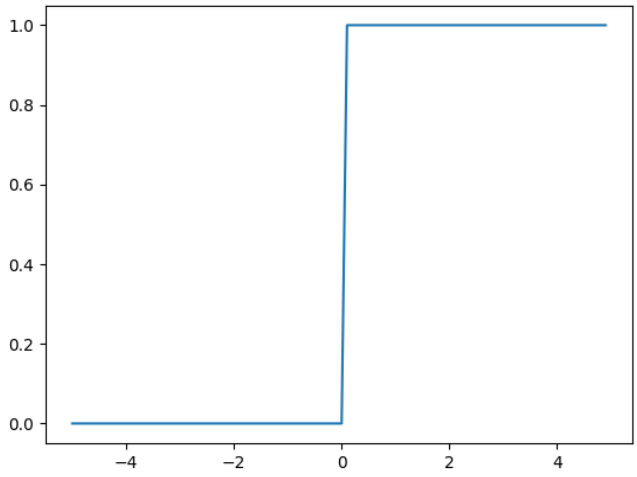
$$\varphi(v) = \begin{cases} 1 \text{ se } v \geq 0 \\ 0 \text{ se } v < 0 \end{cases} \tag{2.3}$$

onde

$$v = \sum wx + b \tag{2.4}$$

Esta função é mais comumente utilizada em problemas que envolvem circuitos lógicos, onde apenas dois valores são possíveis, 0 ou 1, conforme ilustra seu gráfico, na Figura 4.

Figura 4 – Função degrau.



Fonte – Elaborado pelo autor.

2.2.3.2 Função Sigmoide

A função Sigmoide é uma das funções de ativação mais importante e uma das mais utilizadas. De acordo com Battiti e Brunato (2017), Sigmoide é uma função de transferência suave e saturante (a saída satura a zero para sinais negativos de grande magnitude e valor um para sinais positivos de grande magnitude). A está função é dado o nome de Sigmoide ou Sigmoidal, devido à forma de "S"que está forma em seu gráfico (Figura 5).

A função Sigmoide conforme Bishop (2006) é utilizada em problemas de classificação binária. Battiti e Brunato (2017) explicam que está característica permite que está função seja mais adequada para dar respostas binárias tais como "sim/não"e modelar probabilidades, já que está produz valores de saída em um intervalo de números reais entre 0 e 1 incluindo ambos, desta forma sendo caracterizada com um novo tipo de neurônio.

Segundo a análise de Haykin (2009), Sigmoide é uma função estritamente crescente que exibe um equilíbrio admirável entre o comportamento linear e o não-linear. E segundo Bishop (2006) é definida por:

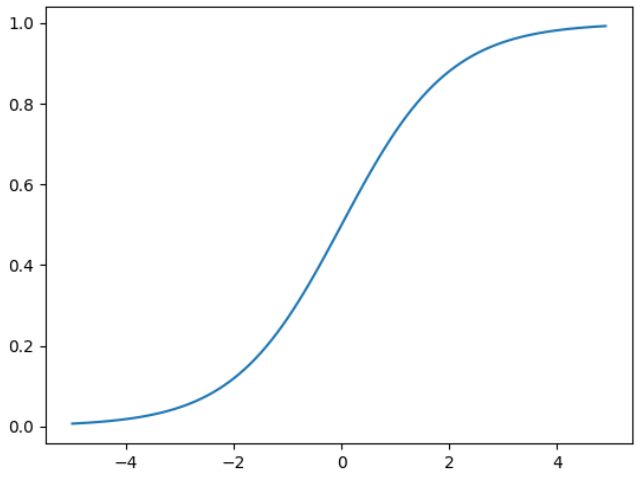
$$y_k = \sigma\left(a_k\right) \tag{2.5}$$

onde

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \tag{2.6}$$

e seu gráfico:

Figura 5 – Função Sigmoide.



Fonte – Elaborado pelo autor.

Haykin (2009) complementa ainda que, além de a função assumir um intervalo contínuo de valores de 0 a 1, está ainda é uma função diferenciável em todo seu domínio, enquanto a função degrau não é. A diferenciabilidade é uma característica fundamental da teoria das redes neurais, pois este é um dos principais fatores que contribuem para o aprendizado da rede.

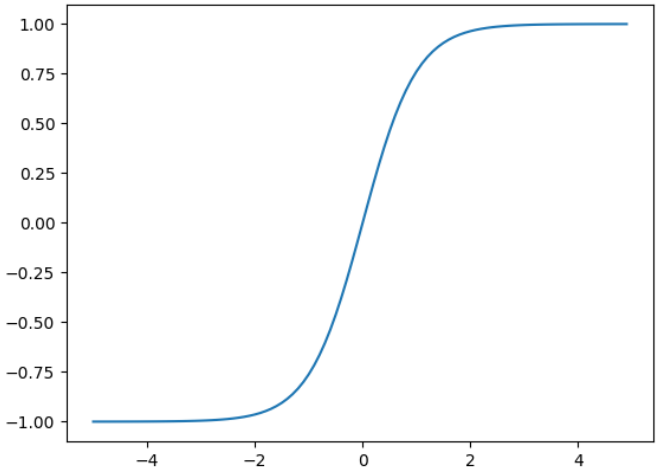
2.2.3.3 Função Tangente Hiperbólica

As funções de ativação apresentadas até o momento estão restritas a um intervalo de valores que varia de 0 a 1 apenas. Porém em algumas situações torna-se necessário mapear valores de entrada em um intervalo que mapeie a entrada para números negativos. Neste caso Nielsen (2019) explica que a função Tangente Hiperbólica (Tanh) é uma substituta da função Sigmoide. A saída da função Tanh com entrada x, peso w e limiar de ativação b, é definida por:

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.7}$$

A a equação 6 mostra que a função Tanh tem uma relação de proximidade com a Sigmoide e através de seu gráfico ilustrado na Figura 6 é possível observar Tanh é uma versão dimensionada da função Sigmoide.

Figura 6 – Função Tangente Hiperbólica.



Fonte – Elaborado pelo autor.

Nielsen (2019) ainda explica que a maior diferença entre Tanh e Sigmoide está no intervalo de saída da função, onde na primeira varia de -1 a 1, enquanto que na posterior apenas de 0 a 1.

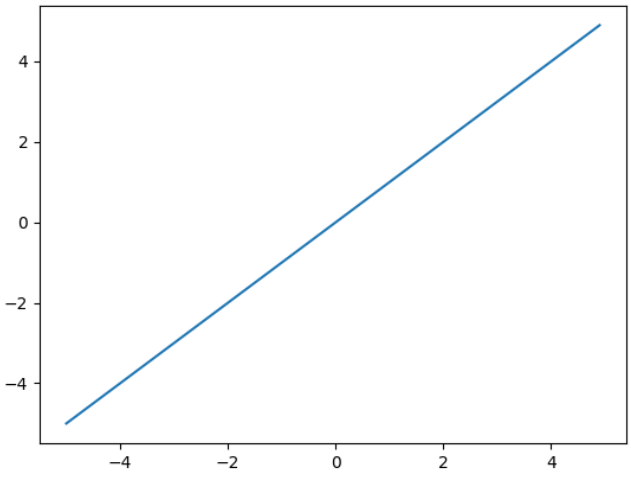
2.2.3.4 Função Linear

De acordo com Silva, Spatti e Flauzino (2010), a função de ativação linear produz resultados de saída idênticos aos valores de entrada da função e são definidos pela seguinte equação:

$$y_k = a_K \tag{2.8}$$

Em problemas de regressão, como o abordado nesta pesquisa, geralmente essa função é utilizada na camada de saída. A 7 a seguir, ilustra seu gráfico.

Figura 7 – Função Linear.



Fonte – Elaborado pelo autor.

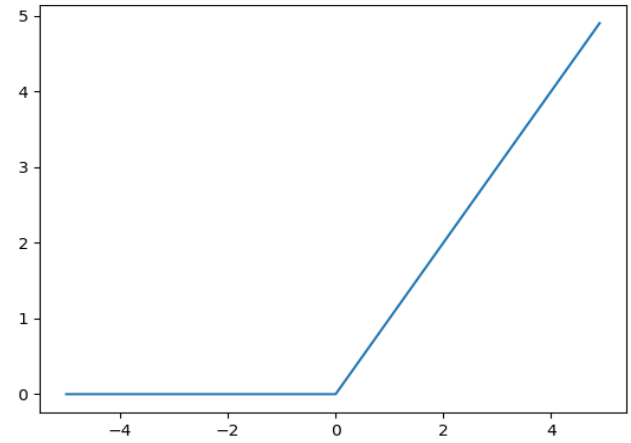
2.2.3.5 Função Rectified Linear Unit - ReLU

De acordo com Nielsen (2019) a função ReLU é uma outra variação da função Sigmoide. Sendo sua saída com respeito a entrada x, vetor de pesos w e limiar de ativação b definida por:

$$\max(0, wx + b) \tag{2.9}$$

Graficamente a função tem tem a seguinte aparência (Figura 8)

Figura 8 – Função ReLU.



Fonte – Elaborado pelo autor.

Nielsen (2019) ainda complementa que a Relu pode ser utilizada para computar qualquer tipo de função e pode ser treinada junto com algoritmos tais como backpropagarion e o gradiente descendente estocástico. E ainda consegue resolver alguns problemas de saturação que as funções Sigmoide e Tangente Hiperbólica apresentam.

2.2.3.6 Função Leaky Rectified Linear Unit - LeakyReLU

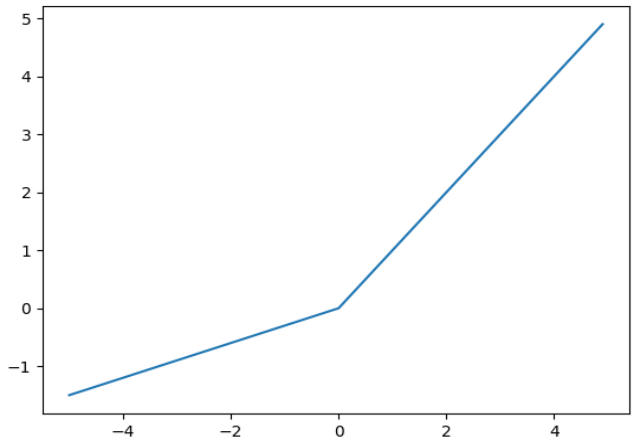
A ativação ReLU sofre do problema conhecido como dying ReLUs: durante o processo de treinamento da RNA, alguns neurônios efetivamente acabam "morrendo", isso é, eles não produzem outro valor além de 0 (zero). Em algumas situações, a metade da rede acaba morrendo, principalmente quando a soma ponderada dos neurônios na camada de entrada da rede produzem um resultado negativo (GÉRON, 2017).

Para resolver esse problema, de acordo com Géron (2017), a função LeakyReLU que é uma variante da função ReLU pode ser empregada.Essa função é similar a sua predecessora, porém nesta função a inclinação da reta para valores menores que zero é modificada por um parâmetro α, desta forma sendo definida como:

$$\max(\alpha \cdot w \cdot x, w \cdot x + b) \tag{2.10}$$

Graficamente através da Figura 9, pode-se observar a mudança da inclinação da reta usando-se 0.3 como um valor de α.

Figura 9 – Função LeakyReLU com α 0.3.



Fonte – Elaborado pelo autor.

Ainda de acordo com Géron (2017), a literatura tem comprovado através de uma série de experimentos que a função LeakyReLU tem superado a sua predecessora ReLU.

2.2.4 Processo de treinamento de uma Rede Neural Artificial

Uma das características mais importantes das RNAs é a sua capacidade de aprender por meio de exemplos. Essa capacidade atribuí as Redes Neurais Artificiais a habilidade de aprender por intermédio de treinamento e, após treinamento suficiente, serem capazes de resolver problemas desconhecidos da mesma classe (KRIESEL, 2007), lhes conferindo uma melhora de desempenho e as adaptando ao seu ambiente (FINOCCHIO, 2014).

Essa capacidade pode ser adquirida através de um processo iterativo de ajustes de parâmetros da rede, os pesos sinápticos, que guardam ao final do processo, o conhecimento que a rede adquiriu do ambiente externo, de modo que o processo de aprendizado encerra-se quando a RNA consegue generalizar soluções para uma classe de problemas (BRAGA; CARVALHO; LUDERMIR, 2011), (FINOCCHIO, 2014).

De acordo com Braga, Carvalho e Ludermir (2011), os algoritmos de treinamento diferem, primordialmente, na forma como os pesos sinápticos são calculados. Porém segundo a análise de Kriesel (2007), a Rede Neural muda conforme seus componentes mudam e teoricamente pode aprender por viés de outros fatores tais como:

1. desenvolvimento de novas conexões entre neurônios;
2. remoção de conexões existentes;
3. alteração dos pesos das sinapses;
4. mudança no limiar dos neurônios;
5. variação das funções de ativação entre os neurônios;
6. desenvolvendo novos neurônios, ou
7. excluindo neurônios existentes bem como suas conexões associadas.

Neste sentido é possível observar que existem diversas estratégias de treinamento para os algoritmos das RNAs que de acordo com Rojas (1996) podem ser agrupados em duas classes de algoritmos de aprendizado: o aprendizado supervisionado e o não supervisionado.

2.2.4.1 Aprendizado supervisionado

De acordo com Silva, Spatti e Flauzino (2010) este paradigma de aprendizado consiste em se ter disponível, considerando cada amostra dos sinais de entrada, as respectivas saídas desejadas, ou seja, cada amostra de treinamento é composta por dados de entrada e suas respectivas saídas.

Portanto conforme Kriesel (2007), para cada conjunto de treinamento que é alimentado na rede, a saída, é comparada diretamente com a solução correta e os pesos da rede são alterados de acordo com os dados reais e os gerados pela rede. Neste sentido existe torna-se necessário fazer uma separação do conjunto de dados.

O conjunto total das amostras segundo Finocchio (2014), deve ser dividido aleatoriamente em cerca de 60% a 90% para o treinamento da RNA, dados estes que serão utilizados durante o processo de treinamento, a fim de que a rede possa aprender as regras associadas ao processo. Enquanto que os dados restantes são utilizados na fase de testes de para verificar o grau de generalização da rede, observando se RNA produz saídas adequadas para os dados. Silva, Spatti e Flauzino (2010) salientam que se deve levar em consideração a caracterização estatística dos dados no dimensionamento dos conjuntos.

De acordo com Braga, Carvalho e Ludermir (2011) o aprendizado supervisionado é aplicável a problemas em que se deseja obter uma relação entre padrões de entrada e saída. Neste sentido Soares (2014) explica que este tipo de treinamento é o mais utilizado em problemas de previsão hidrológica. No contexto de previsão de inundações o nível do rio, e/ou vazão e ou/precipitação são utilizados com a saída, normalmente o nível do rio fornecido por postos de coleta que fornecem os sinais de entrada. Desta forma, ao serem apresentados dados não considerados na rede, poder- se-á obter uma relação entre os sinais de entrada e saída.

A aprendizagem supervisionada é de acordo com Rojas (1996) dividida em métodos que usam treinamento com reforço ou correção de erros. Braga, Carvalho e Ludermir (2011) explicam que o aprendizado por reforço se caracteriza por um processo de tentativa e erro que objetiva maximizar o índice de desempenho escalar chamado de sinal de reforço através da atualização e correção dos pesos.

De acordo com Rojas (1996) no aprendizado com correção de erros, a magnitude do erro, junto com o vetor de entrada, determina a magnitude das correções do peso, e em muitos os casos se procura eliminar o erro em uma única iteração de treino.

2.2.4.2 Aprendizado não-supervisionado

De acordo com Kriesel (2007) neste paradigma apenas o conjunto de padrões de entrada é considerado. Braga, Carvalho e Ludermir (2011) explicam que durante o processo de treinamento os dados são apresentados continuamente à rede, e a existência de padrões nestes dados fazem com que o aprendizado seja possível. Soares (2014) salienta que este tipo de treinamento é aplicado a problemas que objetivam à detecção de características estatisticamente relevantes nos dados de entrada, como por exemplo, a descoberta de agrupamentos, ou classes.

Silva, Spatti e Flauzino (2010) complementam que a própria rede deve se auto-organizar em relação a características existentes entre os elementos que fazem parte do conjunto total de dados, de modo a identificar subconjuntos que apresentem padrões. Desta forma o algoritmo de aprendizado ajusta os pesos sinápticos e limiares dos neurônios da rede de modo a refletir está representação internamente dentro da própria rede.

De acordo com Soares (2014) o aprendizado por competição é uma variação do aprendizado não supervisionado e segundo Braga, Carvalho e Ludermir (2011), neste paradigma quando um dado de entrada alimenta a rede as unidades de saída disputam entre sí para decidir qual delas será a vencedora, e consequentemente, sua saída será ativada e seus pesos atualizados. Ao final deste processo apenas um neurônio restará ativo junto com os seus pesos associados ajustados.

2.2.4.3 Algoritmo de Treinamento

O processo de treinamento das redes neurais artificiais normalmente ocorre através do algoritmo de retro propagação (Backpropagation), que se utiliza do método de treinamento supervisionado para calcular o erro produzido pela rede por intermédio de uma função de erro, com o objetivo de minimizá-lo através de um mecanismo de correção utilizado para ajustar os pesos e limiares da rede (BRAGA; CARVALHO; LUDERMIR, 2011), (GÉRON, 2017).

De acordo com Silva, Spatti e Flauzino (2010), o treinamento ocorre por meio de aplicações sucessivas de duas fases, geralmente denominadas de propagação adiante (forward propagation) e retro propagação (backward propagation).

Na primeira fase o algoritmo alimenta a rede com cada amostra do conjunto de dados e calcula a saída para cada neurônio em cada camada consecutiva, ou seja, essa fase é equivalente a fazer predições com a rede neural; o erro acometido pela rede no neurônio de saída é mensurado por uma função de erro através das respostas produzidas pela rede e as respostas

desejadas (GÉRON, 2017).

A segunda fase do treinamento utiliza-se do erro calculado na camada de saída durante a primeira fase para mensurar o quanto os neurônios da camada anterior contribuíram para o erro da camada posterior. Desta forma os erros são propagados das camadas posteriores para as anteriores até a camada de entrada. Assim o algoritmo consegue mensurar de forma eficiente o gradiente do erro através de todas as conexões entre os pesos da rede por meio da retro propagação do erro - procedimento este que deu origem ao nome do algoritmo (GÉRON, 2017).

De acordo com Rojas (1996) a minimização do erro é realizada por intermédio do método de descida do gradiente (Gradient descent), que busca pelo ponto mínimo da função de erro. De acordo com Brilliant (2019) para treinar uma RNA através deste método, é necessário realizar o calculo do gradiente de uma dada função erro $E(X, \theta)$ com respeito a matriz de pesos wij^k e a matriz de limiares b_i^k. Desta forma, dada uma taxa de aprendizado α que controla a direção do gradiente, os pesos e limiares (coletivamente denotados por θ) são atualizados a cada iteração do algoritmo de acordo com a equação 2.11:

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial E\left(X, \theta^t\right)}{\partial \theta} \tag{2.11}$$

Onde θ^t denota os parâmetros da RNA na iteração t da descida do gradiente.

Desta forma, para que todos os pesos e limiares de todas as camadas da RNA sejam atualizados, a regra da cadeia é aplicada, permitindo assim que o erro seja retro propagado da camada de saída até a camada de entrada enquanto que os parâmetros da rede são atualizados. A equação 2.12 exemplifica a regra da cadeia aplicada sobre a derivada parcial da função de erro:

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial a_j^k} \frac{\partial a_j^k}{\partial w_{ij}^k} \tag{2.12}$$

onde a_j^k é a função de ativação do neurônio j na camada k aplicada sobre a combinação linear das entradas com os pesos somados aos limiares – procedimento análogo ao descrito nas equações 2.1 e 2.2. Assim a equação 2.12 basicamente expressa que, a mudança na função de erro com respeito aos pesos w_{ij}^k é um produto da mudança da função de erro E com respeito a ativação a_j^k , pela mudança na ativação a_j^k com respeito aos pesos w_{ij}^k.

Normalmente a função de erro E usada pelo algoritmo de retro propagação é a função do erro médio quadrático (*Mean Square Error* - MSE) (Equação 2.13):

$$E(X,\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{2.13}$$

Onde y_i é o valor real passado para a rede através do par de entrada e saída (x_i, y_i) e $\hat{y}_i$ é a saída produzida pela rede de acordo com entrada x_i .

- **Variantes do método de descida do gradiente**

De acordo com Garcia (2019), existem três variações do algoritmo de descida do gradiente, que são baseados na quantidade de amostras utilizadas para se calcular o gradiente:

- *Batch Gradient Descent*
- *Stochastic Gradient Descent*
- *Mini-batch Gradient Descent*

O *Batch Gradient Descent* calcula o erro para cada observação no conjunto de dados, porém a atualização dos pesos ocorre apenas após o erro de todas as observações serem avaliados. Esta estratégia requer um grande consumo de recursos computacionais pois exige que todo o conjunto de amostras seja mantido na memória.

O método estocástico (*Stochastic Gradient Descent*) realiza a atualização dos parâmetros para cada observação do conjunto de amostras, porém utiliza-se de apenas uma observação para realizar a atualização dos parâmetros, desta forma acaba sendo mais rápido que o método *Batch*, no entanto, as atualizações frequentes levam as taxas de erro a apresentarem uma alta variança.

A variante *Mini Batch Gradient Descent* é uma combinação das duas variantes supracitadas. Este método realiza as atualizações dos pesos para um *batch* (conjunto) de observações, que normalmente varia no tamanho, geralmente de 32 à 256 observações por *batch*.

Essa estratégia a torna uma boa escolha ao se dispor de conjuntos de dados volumosos. Além disso também é o método recomendado pela literatura e portanto também utilizado pelas RNAs propostas nesta pesquisa.

2.2.5 Arquiteturas de Redes Neurais Artificiais

De acordo com Haykin (2009) a forma que os neurônios de uma rede neural são estruturados está intimamente ligada ao algoritmo de aprendizado usado para treiná-la. Desta

forma pode-se entender, que os algoritmos de aprendizado (regras) usados no projeto de redes neurais como sendo estruturados.

Neste contexto, segundo Silva, Spatti e Flauzino (2010), a arquitetura de uma RNA define a forma como os seus diversos neurônios estão arranjados uns em relação aos outros. Enquanto que a topologia de uma RNA, considerando determinada arquitetura, pode ser determinado como sendo as diferentes formas de composições estruturais que esta poderá assumir.

As principais arquiteturas de RNAs segundo identifica Haykin (2009) bem como Silva, Spatti e Flauzino (2010) são:

1. Arquitetura *feedforward* de camada simples: Nesta arquitetura tem-se uma única camada de nós de neurônios que se projetam diretamente para uma camada de saída.

2. Arquitetura *feedforward* de camadas múltiplas: Essa arquitetura de distingue pela presença de uma ou mais camadas ocultas, cujos nós são identificados como neurônios ocultos ou unidades ocultas. O termo "oculto"diz respeito ao fato de que está parte da RNA não év vista diretamente da entrada ou saída da rede.

3. Arquitetura recorrente ou realimentada: São redes em que as saídas dos neurônios são realimentadas como sinais de entrada para outros neurônios, ou seja, possuí um laço de *feedback*. A característica da retroalimentação permite que tais redes possam ser utilizadas em sistemas variantes com o tempo, como previsão de séries temporais, como é o caso da natureza de dados hidrológicos.

2.2.6 Redes Neurais Profundas

Segundo Deng e Yu (2014), o aprendizado profundo é constituído por um conjunto de técnicas de aprendizado de máquina, onde múltiplas camadas de processamento de informações em arquiteturas hierárquicas são exploradas e o conhecimento é caracterizado através de múltiplos níveis de abstração, que partem de conceitos mais simples nos níveis mais baixos até conceitos mais avançados, ligeiramente mais abstratos, nos níveis mais altos.

No contexto de Redes Neurais, de acordo com Shen (2017), o aprendizado profundo é caracterizado pelo empilhamento de múltiplas camadas de neurônios em sistemas combinatórios e cuidadosamente projetados nas redes neurais multicamadas, que atuam diretamente em grandes quantidades de dados brutos.

Essa utilização privilegiada em bases de dados volumosas pode ser apontada como a principal característica das RNAs profundas, consequentemente devido a natureza estrutural que

apresentam, se diferenciam das RNAs rasas através de alguns aspectos dentre os quais Patterson e Gibson (2017) destacam: quantidade mais acentuada de neurônios; complexas conexões entre as camadas; aumento do poder de processamento e extração de recursos automática.

Tais características de acordo com Längkvist, Karlsson e Loutfi (2014) tem permitido aumentar a capacidade das redes neurais profundas de modelar estruturas complexas nos dados. Shen (2017) comenta que esta capacidade se estende a funções com alta complexidade de dependências espaçotemporais e distribuições de grandes quantidades de dados, advindas das estruturas formadas pelas redes de grande profundidade.

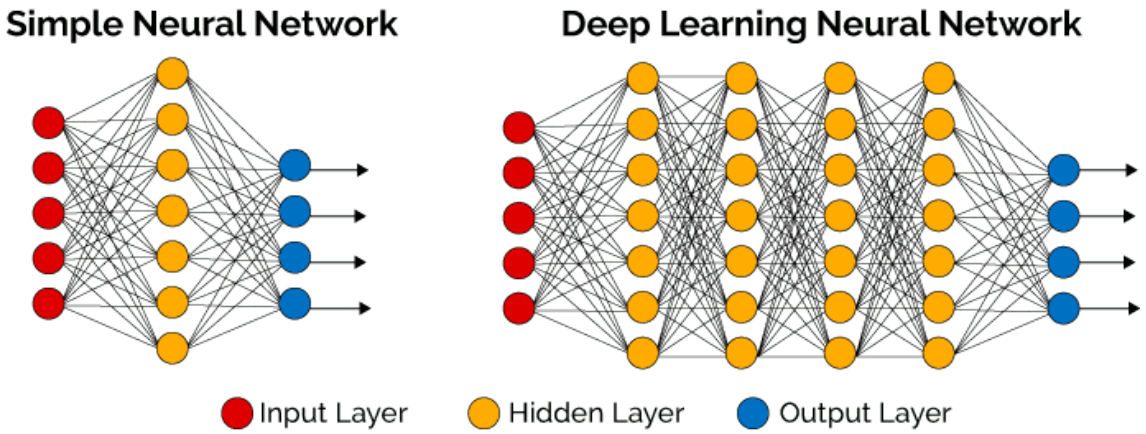
A capacidade aumentada tem proporcionado às redes que utilizam esta abordagem aprender características dos dados através de representações que são expressas através de outras mais simples, ou seja, o aprendizado é realizado através de várias etapas (GOODFELLOW; BENGIO; COURVILLE, 2016).

Estas etapas permitem que os dados analisados sejam divididos em várias partes, formando uma hierarquia de conhecimento, onde camadas iniciais são responsáveis pela detecção de características mais gerais e as camadas seguintes vão especificando cada vez mais características analisadas.

Neste sentido segundo a análise de LeCun, Bengio e Hinton (2015), as RNAs profundas exploram a propriedade de que muitos sinais naturais são hierarquias de composição, nas quais recursos de nível superior são obtidos compondo-os de nível inferior. Tomando-se a classificação de imagens de rostos de pessoas como exemplo, as camadas iniciais seriam responsáveis por analisar e detectar bordas nos cantos do rosto, identificando a sua forma, enquanto das camadas intermediárias para as finais identificariam a presença de outros aspectos tais como a presença, cor dos olhos, dentes, boca, etc.

Através da Figura 10 essa classificação fica evidente ao se observar as diferenças entre as características estruturais das RNAs. Enquanto a RNA que possuí apenas uma camada escondida (rasa) precisa construir todo o conhecimento de uma única vez nesta sua única camada. A outra que possui mais camadas (profunda) adquire o conhecimento parcialmente através da divisão dos dados que são processados paralelamente entre as múltiplas combinações de seus neurônios, que se encontram nas camadas finais e se juntam para caracterizar o conhecimento adquirido.

Figura 10 – Comparativo entre uma RNA rasa com uma RNA profunda.



Fonte – Data Science Academy (2019).

O paralelismo inerente da estrutura de alguns modelos de RNA, é o principal fator que dá popularidade ao aprendizado profundo, porém essa popularidade só pode ser alcançada recentemente devido a avanços tecnológicos em hardware (GOODFELLOW; BENGIO; COURVILLE, 2016), com o surgimento de arquiteturas de processamento paralelo e posteriormente ao adotar-se o uso de GPUs como principal componente de processamento no treinamento das RNAs profundas (ZHANG et al., 2019).

Zhang et al. (2019) ainda salienta que muitos desses avanços de processamento se deram pela necessidade de maior poder computacional exigido pelo aumento da disseminação de grande quantidade de dados através World Wide Web. Além disso, o sucesso das redes neurais profundas não são apenas caracterizadas pelos avanços de hardware, mas também por avanços algorítmicos, que permitiram contornar dificuldades encontradas nas redes neurais tradicionais e deram surgimento à novas arquiteturas de RNAs profundas, dentre as quais Deng e Yu (2014) as classificam em três classes principais:

1. **Redes profundas para aprendizado não supervisionado ou generativo:** Os algoritmos desta classe têm como objetivo capturar correlações de alta ordem dos dados observados ou visíveis para fins de análise ou síntese de padrões, geralmente na indisponibilidade de informação de rótulos de classes - (valores de saída).

2. **Redes profundas para aprendizado supervisionado:** Fornecem poder discriminativo para fins de classificação de padrões, diversas vezes caracterizando as distribuições posteriores de classes condicionadas aos dados visíveis. Neste sentido rótulos de classe

estão sempre disponíveis no conjunto de dados no aprendizado supervisionado. Neste contexto as RNAs são chamadas de redes profundas discriminativas.

3. **Redes profundas híbridas:** As arquiteturas híbridas objetivam utilizar a arquitetura discriminativa através de resultados provindos da arquitetura generativa ou não supervisionada. Esse objetivo pode ser alcançado tanto através de otimizações e regularizações das arquiteturas supervisionadas, quanto também ao através de critérios discriminativos do aprendizado supervisionado usados para estimar os parâmetros em qualquer uma das redes profundas, generativas ou não supervisionadas.

As arquiteturas profundas podem pertencer a uma ou mais das classes acima citadas, dentre as destacadas pela literatura, algumas das quais pode-se citar são:

• ***Deep Belief Network (DBN)*:** Representam modelos probabilísticos generativos compostos de múltiplas camadas escondidas de variáveis estocásticas (DENG; YU, 2014). Esses modelos resolvem problemas encontrados em algoritmos tradicionais de redes neurais profundas tais como: grandes conjuntos de dados rotulados para treinamento; tempo de treinamento lento e técnicas de seleção de parâmetros para encontrar mínimos locais ótimos (AREL; ROSE; KARNOWSKI, 2010).

• ***Boltzmann Machine (BM)*:** Representa uma rede simetricamente conectada, através de unidades neuronais que tomam decisões binárias estocasticamente (DENG; YU, 2014).

• ***Deep Neural Network (DNN)*:** É caracterizado por uma rede neural Multilayer Percepeton (MLP) com muitas camadas ocultas, cujos pesos são completamente conectados e frequentemente inicializados através de técnicas tanto de aprendizado supervisionado quanto não supervisionado (DENG; YU, 2014).

• ***Deep Autoencoder*:** É um modelo de rede neural profunda cujos valores de saída são os próprios valores de entrada, ou seja, não se utiliza de rótulos de classes no aprendizado e portanto é categorizado como modelo de aprendizado não supervisionado (DENG; YU, 2014).

• ***Recurrent Neural Network (RNN)*:** Geralmente utilizada em problemas de séries temporais, são caracterizadas ao se conectar os neurônios de saída nos de entrada em uma arquitetura feedforward (LÄNGKVIST; KARLSSON; LOUTFI, 2014). O objetivo principal é realizar predições através de um processo iterativo que utiliza não apenas os dados

de entrada, mas também resultados produzidos pela rede como dados de entrada para realizar as próximas predições (PETNEHÁZI, 2019).

• ***Convolutional Neural Network (CNN)*:** Representa uma família de redes neurais multicamadas particularmente modeladas para serem utilizadas em dados de bidimensionais, tais como dados de imagens e vídeos (AREL; ROSE; KARNOWSKI, 2010). Podendo também serem aplicadas a dados unidimensionais para sinais e sequências incluindo linguagem, bem como dados tridimensionais também para vídeos ou imagens volumétricas (LECUN; BENGIO; HINTON, 2015).

Essas arquiteturas melhoraram drasticamente o estado da arte no reconhecimento de fala, reconhecimento visual de objetos, detecção de objetos e muitos outros domínios tais como, descoberta de drogas, genômica, previsão da bolsa de valores, previsões hidrológicas, etc. As redes CNN trouxeram avanços no processamento de imagens, vídeo, fala e áudio, enquanto as redes recorrentes se mostraram eficazes em dados sequenciais como texto e fala (LECUN; BENGIO; HINTON, 2015).

Essa gama de aplicações permitiu que nos últimos anos a esta abordagem se tornar uma técnica bem-sucedida e amplamente utilizada na resolução de problemas de inteligência artificial considerados difíceis e complexos (BENGIO, 2009). No entanto a literatura aponta que seu uso na hidrologia tem sido infrequente, especialmente em configurações que abrangem grandes quantidades de dados (HU et al., 2018), (SHEN, 2017). Neste sentido a modelagem de uma RNA profunda para para a previsão de inundações torna-se um desafio do qual este trabalho objetiva superar a fim de responder a pergunta de pesquisa inicialmente proposta.

2.3 RNAS EMPREGADAS NA PESQUISA

O objetivo desse capítulo é fundamentar as Redes Neurais Profundas utilizadas nesta pesquisa. Inicialmente as RNAs recorrentes e suas vertentes LSTM e GRU são descritas, na sequência as redes convolucionais e por último a rede MLP.

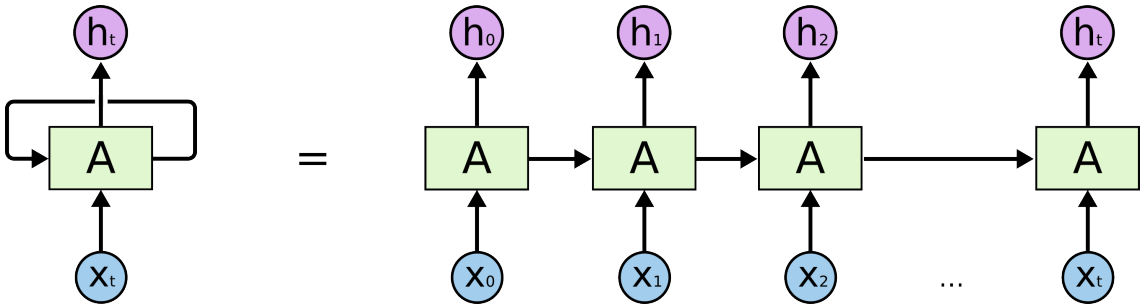
2.3.1 Redes Neurais Recorrentes

De acordo com Goodfellow, Bengio e Courville (2016), as Redes Neurais Recorrentes fazem parte de uma família de RNAs que processam dados sequenciais e de comprimento variável. Em contraste às redes neurais tradicionais de topologia *feedforward*, as RNNs possuem ciclos que permitem ter conexões com células de camadas anteriores ou da mesma camada.

A estrutura de conexão formada pelos neurônios das RNNs, normalmente denotados como células, lhes confere a habilidade manter um estado interno com informações de sequências processadas anteriormente (GOODFELLOW; BENGIO; COURVILLE, 2016). Tal estado interno funciona como uma espécie de memória de curto prazo, utilizada para encontrar correlações temporais em dados sequenciais que possuam algum tipo de relacionamento temporal.

A Figura 11 ilustra uma RNA recorrente, onde A é uma célula da rede, x_t o sinal de entrada no instante t e h_t o respectivo sinal de saída, também utilizado para alimentar as entradas das próximas camadas ocultas. A representação da rede no tempo é apresentada ao se desenrolar o laço em alguns instantes, desta forma a saída computada no instante $t = 0$ influenciará a saída no próximo instante no momento em que a rede receber uma nova entrada, e assim de maneira sucessiva para os próximos instantes.

Figura 11 – Uma RNN desenrolada.



Fonte – Olah (2015)

Durante o processo de treinamento da rede, problemas de dependências de longo prazo surgem devido ao uso do algoritmo Backpropagation. Goodfellow, Bengio e Courville (2016) explicam que devido aos gradientes serem propagados por muitos estágios da rede, na maioria das vezes eles tendem a desaparecer, levando a rede a diminuir a velocidade do aprendizado até o ponto em que para de aprender. Uma outra situação mais atípica que também pode ocorrer é explosão dos gradientes: situação onde se tornam muito grandes e a rede começa a oscilar, nesse caso inibindo a função de erro de convergir para um mínimo local.

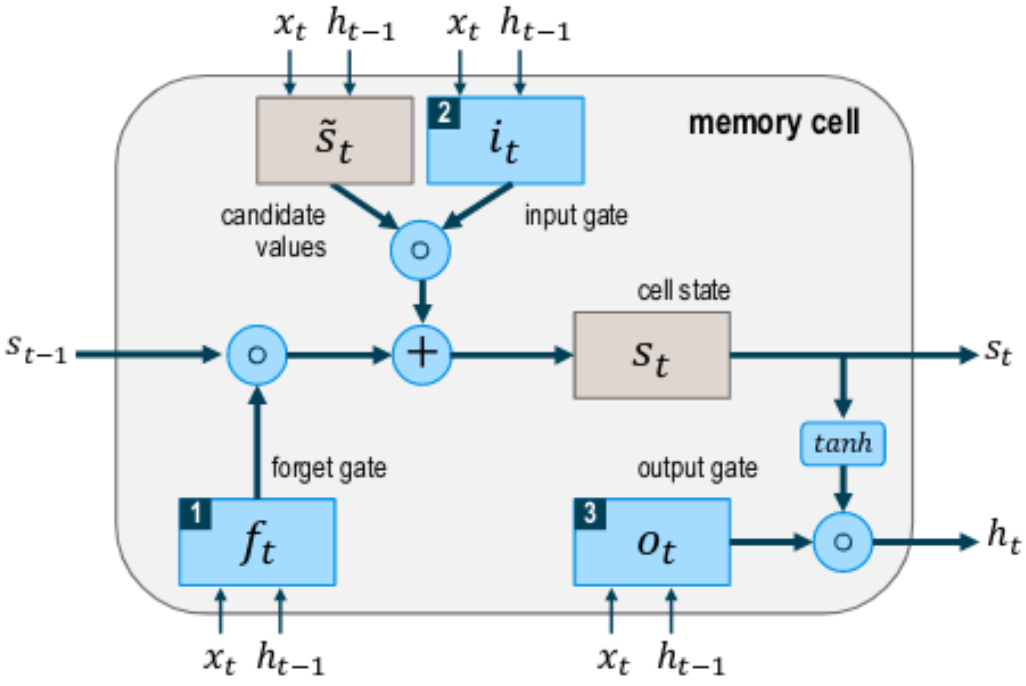
Para superar essas limitações Hochreiter e Schmidhuber (1997) propuseram um novo tipo de arquitetura baseada em rede neural recorrente que aplica fluxo constante de erros através do estado interno de novas unidades especiais, chamadas unidades de memória de longo prazo (*Long short-term memory*). O fluxo constante de erros impede que o gradiente exploda ou desapareça e ainda permite que a rede neural lide com ruído e valores contínuos.

2.3.1.1 Long Short-Term Memory - LSTM

A rede LSTM é uma variante das redes neurais recorrentes, sendo introduzida por Hochreiter e Schmidhuber (1997), tem sido especificamente designada para tarefas de aprendizado que lidam com grandes sequências de dados que possuem dependencias de longo prazo, por serem capazes de superar os problemas inerentes das RNNs tradicionais, tais como o desaparecimento dos gradientes.

De acordo com Olah (2015), o modelo LSTM é organizado através de uma estrutura em cadeia, no entanto o módulo de repetição possuí uma estrutura diferente. Em constraste a genérica Rede Neural Recorrente, a Rede Long Short-Term Memory possuí quatro camadas de interação com um único método de comunicação. A estrutura de uma célula de memória LSTM é ilustrada pela Figura 12 a seguir:

Figura 12 – Arquitetura da célula de memória LSTM.



Fonte – Adaptado de Fischer e Krauss (2017)

As redes LSTM são compostas de uma camada de entrada, uma ou mais células de memória e uma camada de saída. De acordo com Hu et al. (2018), a característica principal das redes LSTM está contida na camada oculta que consiste nas chamadas células de memória (Figura 12). Cada uma das células de memória possui três portões que mantêm e ajustam seu estado celular s_t: um portão de esquecimento f_t, um de entrada i_t e um de saída o_t.

Fischer e Krauss (2017) explicam que a cada etapa temporal t, cada um dos três portões são apresentados com a entrada de um elemento da sequência temporal x_t, bem como a saída da célula de memória da etapa temporal anterior h_{t-1}. Desse modo, os portões atuam como filtros, cumprindo as seguintes finalidades:

- O portão do esquecimento (*forget gate*) define quais informações serão removidas da célula de estado (*cell state*).
- O portão de entrada (*input gate*) especifica quais informações devem ser adicionadas a célula de estado.
- O portão de saída (*output gate*) determina quais informações da célula de estado são usadas como saída.

De acordo com Olah (2015), durante o processo de treinamento, o primeiro passo que ocorre em uma célula de memória é decidir quais informações serão removidas da célula de estado s_t. Tal decisão é tomada pela função de ativação *Sigmoid* na camada de esquecimento onde situa-se o forget gate f_t.

Conforme Fischer e Krauss (2017), os valores de ativação f_t na etapa temporal t são computados com base na entrada atual x_t, na saída h_{t-1} da célula de memória da etapa temporal anterior $t-1$ e no limiar do forget gate b_f, de acordo com a equação 2.14:

$$f_t = \sigma\left(W_{f,x}x_t + W_{f,h}h_{t-1} + b_f\right) \tag{2.14}$$

O papel da função de ativação *Sigmoid* na equação 2.14 é escalonar os valores de ativação para o intervalo $[0, 1]$, onde 0 indica que a informação deve ser descartada (esquecida) e 1 indica que deve ser mantida (relembrada).

No estágio seguinte, de acordo com Olah (2015), determina-se quais informações serão adicionadas à célula de estado. Conforme Fischer e Krauss (2017), para que isso ocorra, um procedimento é realizado através de duas operações: Inicialmente, valores potencialmente candidatos s̃ a serem adicionados a célula de estado são computados (equação 2.15); Na sequência, os valores de ativação i_t do *output gate* são calculados (equação 2.16).

$$\tilde{s} = \tanh\left(W_{\tilde{s},x}x_t + W_{\tilde{s},h}h_{t-1} + b_{\tilde{s}}\right) \tag{2.15}$$

$$i_t = \sigma\left(W_{i,x}x_t + W_{i,h}h_{t-1} + b_i\right) \tag{2.16}$$

No terceiro estágio, de acordo com Fischer e Krauss (2017) a nova célula de estado s_t é calculada usando-se os resultados dos estágios anteriores de acordo com a equação 2.17:

$$S_t = f_t \circ S_{t-1} + i_t \circ \tilde{S}_t \tag{2.17}$$

Onde $\circ$ denota o produto de Hadamard.

No estágio final, a saída h_t da célula de memória é computada através das equações 2.18 e 2.19:

$$o_{\mathrm{t}} = \sigma \left(W_{o,x} x_t + W_{o,h} h_{t-1} + b_o \right) \tag{2.18}$$

$$h_t = o_t \circ \tanh \left(s_t \right) \tag{2.19}$$

Fischer e Krauss (2017) explicam que a rede LSTM processa uma entrada sequencial, computando sequência por sequência temporal de atributos. Desta forma, a entrada é computada pelas equações supracitadas, uma vez que o último elemento da sequência temporal é processado, a saída final baseada em toda a sequência é retornada.

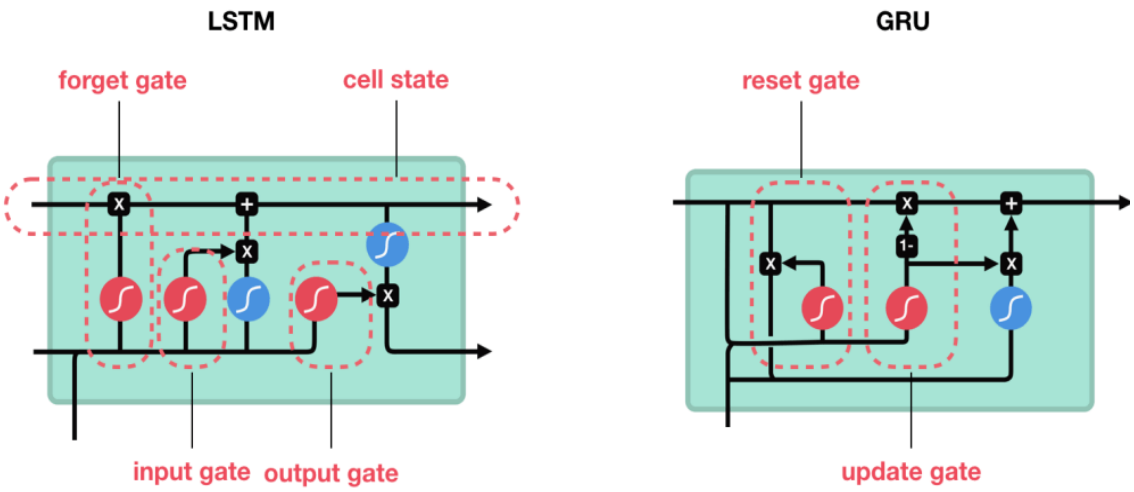
2.3.1.2 Gated Recurrent Unit – GRU

A RNN GRU é uma variação da rede LSTM introduzida por Cho et al. (2014). De acordo com Olah (2015), nesta variação os portões *forget gate* e o *input gate* são combinados em um único portão denominado de *update gate* (portão de atualização) e a célula de estado s_t e o estado oculto h_t são mesclados. Desta forma, uma célula GRU possuí apenas 2 portões, um portão de redefinição (*reset gate*) e um portão de atualização (*update gate*).

Similarmente as redes LSTM, as GRUs utilizam os portões como um mecanismo para filtrar informações, de modo a decidir quais devem ser descartadas e quais devem ser mantidas. Desta forma lhes conferindo a habilidade de manter dependências de longo prazo, e a contornar o inerente problema do desaparecimento dos gradientes, presente nas RNNs tradicionais.

A estrutura da célula GRU é uma versão simplificada da LSTM que geralmente oferece desempenho comparável e é significativamente mais rápida de calcular (ZHANG et al., 2019). A Figura 13 exibe um comparativo entre uma célula LSTM e uma célula GRU.

Figura 13 – Comparativo entre as células LSTM e GRU.



Fonte – Adaptado de Data Science Academy (2019)

De acordo com Kostadinov (2017) o *reset gate* é utilizado para decidir o quanto das informações passadas devem ser descartadas. Basicamente, esse *gate* serve como uma memória de curto prazo. O *update gate* ajuda o modelo a determinar o quanto de informação de etapas anteriores serão passadas adiante – desta forma o modelo consegue inibir o problema do desaparecimento dos gradientes, pois serve como uma memória de longo prazo.

Segundo (ZHANG et al., 2019), as equações para o reset gate e update gate são respectivamente:

$$r_t = \sigma\left(x_t W_{x,r} + h_{t-1} W_{h,r} + b_r\right) \tag{2.20}$$

$$z_t = \sigma\left(x_t W_{x,z} + h_{t-1} W_{h,z} + b_z\right) \tag{2.21}$$

Conforme Zhang et al. (2019), a equação 2.22 tem o propósito de ser usada em conjunto com o *reset gate* para armazenar informações relevantes dos etágios passados, servindo como uma espécie de memória da etapa atual.

$$\tilde{h}_t = \tanh\left(x_t W_{x,\tilde{h}} + \left(r_t \circ h_{t-1}\right) W_{\tilde{h},\tilde{h}} + b_h\right) \tag{2.22}$$

Por último, a rede precisa calcular a informação para a etapa da célula atual de modo a passá-la para frente na rede. Zhang et al. (2019) explica que para isso ocorrer, o efeito do *update gate* deve ser incorporado de modo a determinar o novo estado h_t, que se baseia

na quanto de informação do estado $tildeh_t$ é usada. Desta forma a equação final é calculada conforme 2.23:

$$h_{\mathrm{t}} = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t \tag{2.23}$$

Embora as duas topologias sejam semelhantes, avaliações empíricas realizadas por Chung et al. (2014) e Jozefowicz, Zaremba e Sutskever (2015), mostram que em muitas tarefas ambas arquiteturas possuem um desempenho comparável e que a seleção de parâmetros para ajustar a rede é mais importante do que a escolha entre um modelo e outro. No entanto, devido a célula GRU necessitar de menos parâmetros, o processo de treinamento da rede pode ser mais rápido, ou precisar de menos dados para generalizar. Em contrapartida, ao se ter grande volume de dados, o poder expressivo da LSTM pode levar a resultados melhores.

2.3.2 Convolutional Neural Network – CNN

A Redes Neurais Convolucionais foram introduzidas por LeCun et al. (1998) na década de 90, inicialmente para tarefas de reconhecimento de dígitos manuscritos, no entanto tem sido satisfatoriamente empregadas em outras tarefas de visão computacional, dentre as tais: Detecção e localização de objetos; detecção da face; análise de documentos; classificação e geração de imagens; etc.

A topologia CNN é designada para processar imagens e por isso tem revolucionado a área de visão computacional, permitindo que muitas tarefas antes consideradas difíceis, pudessem ser realizadas facilmente através de redes neurais.

No entanto, as CNNs não se restringem apenas a tarefas de imagens, de acordo com Goodfellow, Bengio e Courville (2016), as CNNs são especializadas no processamento de dados que possuem uma topologia vetorial multidimensional (*grid*). Exemplos incluem dados de séries temporais, representados em um vetor de 1 dimensão, colhendo amostras em intervalos regulares, e dados de imagens, que podem ser considerados como uma matriz de pixeis de 2 dimensões (escala de cinza) ou 3 dimensões (colorida – RGB).

A arquitetura básica de uma CNN se baseia em duas camadas: Camada de convolução e camada de Pooling.

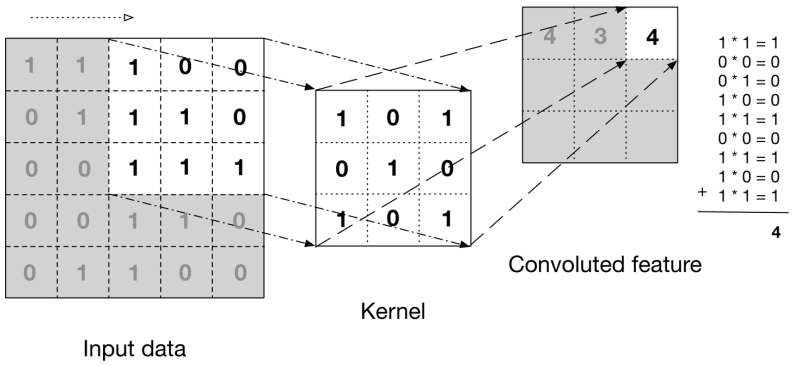
2.3.2.1 Camada de Convolução

De acordo com Karn (2016) a camada convolucional é utilizada para extrair características únicas dos dados de uma imagem. A operação de convolução serve para preservar o relacionamento espacial entre os pixeis através do aprendizado das tais características por viés de filtros que são usados nesta operação.

De acordo com Patterson e Gibson (2017) a operação de convolução é uma operação matemática que aplica filtros (*kernel*) de convolução nos dados de entrada da rede para extrair atributos. Como resultado da operação as informações extraídas são representadas como mapas de características. Esses mapas possuem um formato de representação semelhante aos dados de entrada, geralmente numa forma reduzida, onde apenas as informações mais relevantes são preservadas.

A Figura 14 ilustra a aplicação de um filtro em uma imagem durante uma operação de convolução. Basicamente o filtro representado por uma matriz 3x3 desliza pela imagem representada pela matriz 5x5. Em cada etapa, o filtro é multiplicado ponto a ponto com os valores da imagem dentro de seus limites, no final os valores são somados, produzindo assim o mapa de características - uma matriz também 3x3.

Figura 14 – Operação de convolução.



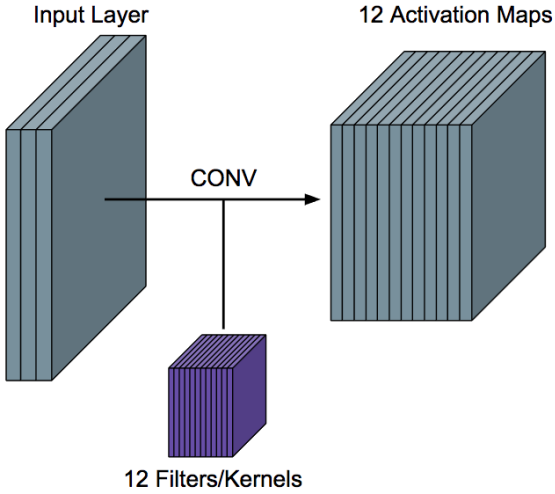
Fonte – Patterson e Gibson (2017)

Normalmente um número maior de filtros são utilizados durante essa operação, pois conforme explica Karn (2016), quanto maior a quantidade de filtros utilizados, mais características da imagem podem ser extraídas, permitindo assim que a rede generalize melhor. Neste sentido o tamanho resultante do mapa de características é controlado por três parâmetros:

- **Profundidade**: A profundidade corresponde a quantidade de filtros usados pela operação de convolução. A Figura 15 por exemplo, ilustra o resultado da operação de convolu-

ção aplicada em uma imagem colorida (RGB) com 12 filtros, produzindo um volume com 12 mapas de características – que podem ser visualizados como matrizes de 2 dimensões empilhadas.

Figura 15 – Operação de convolução com 12 filtros



Fonte – Deshpande (2017)

- **Stride**: O *stride* representa o número de pixeis no qual o filtro desliza sobre a imagem. Quando o *stride* é 1, o filtro é deslizado apenas um pixel por vez. Quando o *stride* é 2, o filtron pula dois pixeis a cada deslize. Quanto maior o *stride* menores são os mapas de características.
- **Zero-padding**: É uma técnica aplicada para evitar que a matriz de entrada seja reduzida e também para controlar o tamanho dos mapas de características. Para isso a matriz de entrada é preenchida com zeros ao seu redor.

Após a operação de convolução, o mapa de características é submetido a uma função de ativação para que o modelo possa representar uma função não linear. Normalmente a função de ativação utilizada é a função *ReLU*, pois não possuí regiões de saturação como as funções *Sigmoid* e *Tanh* (DESHPANDE, 2017).

Matematicamente a operação que ocorre na camada de convolução é semelhante a operação que ocorre em uma rede neural padrão do tipo *feedforward*. De acordo com Valdenegro-Toro (2019) a equação que representa a camada convolucional é dada por:

$$y = f(x * F + b) \tag{2.24}$$

Onde $*$ é a operação de convolução, x é a imagem de entrada, F é o filtro de convolução, b o limiar e f é a função de ativação – aplicada elemento a elemento nas matrizes resultantes do processo de convolução.

Semelhantemente às RNAs tradicionais que aprendem com os dados através do algoritmo Backpropagation, as CNNs também passam pelo mesmo processo, porém neste caso os filtros é que são ajustados pela descida do gradiente, desta forma atuando como se fossem os pesos que ponderam a rede neural tradicional.

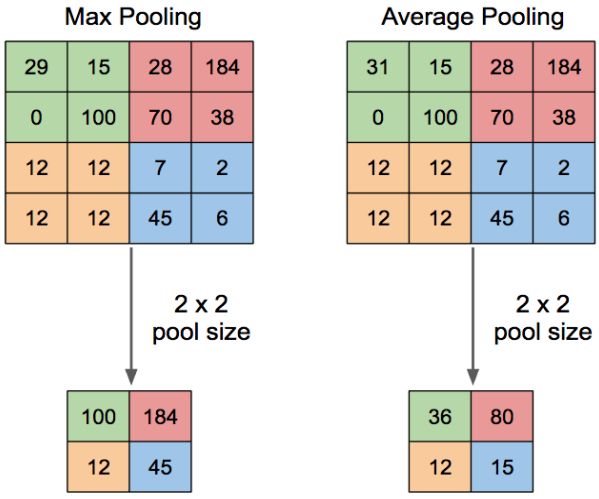
2.3.2.2 Camada de Pooling

A camada de *pooling* é aplicada com o objetivo de realizar uma subamostragem não linear no mapa de características. Essa camada reduz a dimensionalidade de cada mapa de características visando eliminar informações desnecessárias para a identificação de atributos da imagem.

Segundo Deshpande (2017), essa camada é usada principalmente para ajudar a reduzir a complexidade computacional e extrair atributos importantes. O resultado é um volume dos mapas de características menor. A profundidade da entrada ainda é mantida, por exemplo, se houver 12 mapas (como no exemplo da Figura 14), que entram em uma camada *pooling*, a saída também terá 12 mapas de ativação.

A proporção em que o volume será diminuído é definido através do *pool size*. Esse parâmetro diz respeito ao quanto da largura e da altura do volume será redimensionado. Na Figura 16 por exemplo, o pool size de tamanho 2 diminuí a imagem de entrada pela metade.

Figura 16 – Operações de pooling.



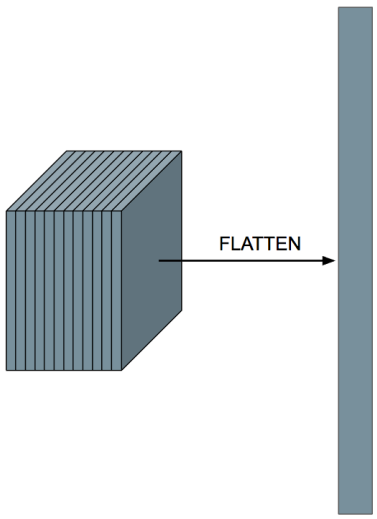
Fonte – Deshpande (2017)

Conforme se pode observar na Figura 16, duas operações distintas de *pooling* são realizadas. A operação *Max Pooling* seleciona os maiores valores dentro de cada janela; a operação *Average Pooling* calcula a média dos valores dentro de cada janela. De acordo com Deshpande (2017), a operação mais utilizada é a *Max Pooling* e o *pooling size* mais comum é 2x2.

2.3.2.3 Fully-Connected Layer

A etapa final de treinamento de uma rede CNN, consiste em "achatar" o volume de dados (Figura 17) resultante das operações de convolução e *pooling*, para que eles possam ser submetidos a uma camada completamente-conectada (*Fully-Conected Layer*), ou seja, uma, ou mais, camadas de rede neural tradicional *feedforward*.

Figura 17 – Volume achatado depois de passar pelas camadas de Convolução e Pooling.



Fonte – Deshpande (2017)

Karn (2016) explica que a saída das camadas convolucionais e de *pooling* representa atributos de alto nível da imagem de entrada. O objetivo da camada *Fully-Connected* é usar esses atributos para classificar a imagem de entrada, com base no conjunto de dados de treinamento. Por exemplo, em uma tarefa de classificação de objetos, a rede aprenderia através dos tais atributos a distinguir entre um objeto pertencente a uma classe A e outro pertencente a uma classe B.

2.3.3 Multilayer Perceptron - MLP

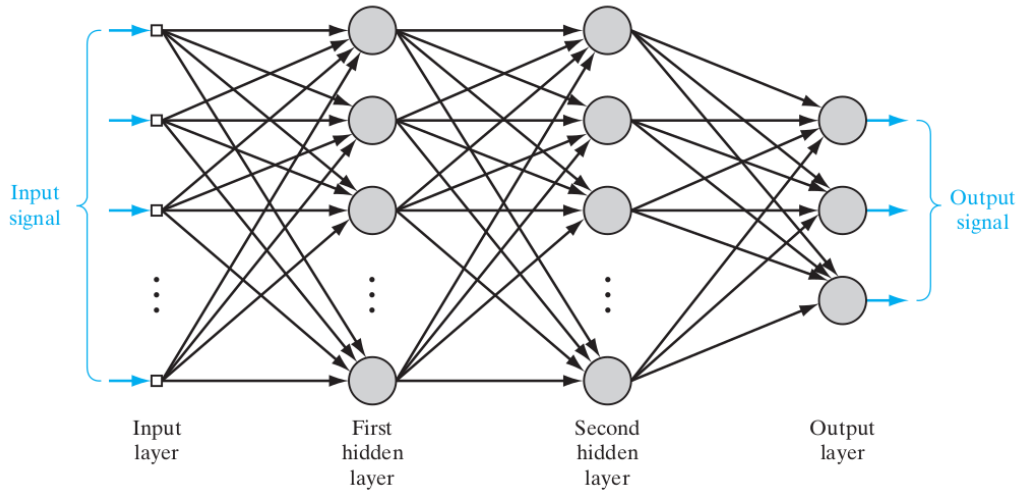
Segundo Silva, Spatti e Flauzino (2010), as Redes Neurais *Multilayer Perceptron* pertencem à topologia *feedforward* de múltiplas camadas, pois são caracterizadas por, pelo

menos, uma camada oculta (intermediária) situada entre as camadas de saída e de entrada. Desta forma as redes MLP possuem pelo menos duas camadas de neurônios, distribuídas entre as camadas intermediárias e a camada de saída.

De acordo com Braga, Carvalho e Ludermir (2011), teoricamente as redes com duas ou mais camadas ocultas podem aproximar qualquer função contínua, neste sentido, grande parte dos problemas práticos dificilmente necessita de mais de uma camada oculta, havendo apenas necessidade no caso de funções descontínuas.

A Figura 18 ilustra uma topologia MLP com duas camadas ocultas e n neurônios, onde os neurônios de cada camada são completamente conectados a todos os outros neurônios de camadas adjacentes. O fluxo de sinal através da rede progride em uma direção para a frente, da esquerda para a direita e camada por camada.

Figura 18 – Multilayer Perceptron com duas camadas ocultas.



Fonte – Haykin (2009)

De acordo com Haykin (2009) dois tipos de sinais são identificados na topologia apresentada pela Figura 18.

1. Sinal de Função: Um sinal de função é um sinal de entrada (estímulo) que chega a extremidade de entrada da rede, propaga-se para a frente (neurônio por neurônio) através da rede e emerge na extremidade de saída da rede como um sinal de saída.

2. Sinal de erro: Um sinal de erro se origina em um neurônio de saída da rede e propaga para trás (camada por camada) através da rede.

Em função de cada camada, Patterson e Gibson (2017) explicam que na camada de entrada (*input signal*), o número de neurônios é o mesmo o a quantidade de amostras de dados disponíveis. Todos os dados são completamente conectados aos neurônios da camada adjacente. Os pesos associados entre as conexões dentre as camadas, codificam a informação extraída dos dados brutos. Desta forma as camadas ocultas são parte essencial da rede MLP que permite as RNAs a modelarem funções não lineares – uma das limitações das redes perceptron de uma camada. Por último a camada de saída produz a resposta gerada pela rede.

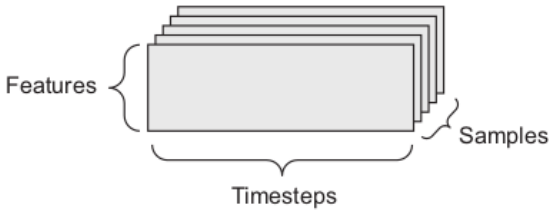
Em uma rede MLP as relações entre as camadas são as conexões de saída de todos os neurônios da camada anterior para todos os neurônios da camada seguinte. Os pesos e limiares associados as tais conexões são ajustados progressivamente a medida em que a rede é treinada através do algoritmo Backpropagation (SILVA; SPATTI; FLAUZINO, 2010) (BRAGA; CARVALHO; LUDERMIR, 2011).

2.3.4 Consideraçoẽs finais

As RNAs recorrentes pregam um papel central nesta pesquisa, pois as vertentes LSTM e GRU fundamentadas são utilizadas tanto de forma isolada, como em conjunto com as CNNs e MLPs nas redes neurais híbridas. Assim as redes CNNs e MLPs prestam um papel "coadjuvante", pois não são utilizadas de forma isolada como as RNNs, porém são utilizadas em conjunto com estas.

Um fato importante ainda a se destacar, diz respeito em como as RNAs são alimentas ao se trabalhar com séries temporais. De acordo com CHOLLET (2018), ao se trabalhar com conjuntos de dados de séries temporais, os dados devem ser armazenados em um vetor de 3D dimensões, com um eixo explicitamente designado para tempo (Figura 19).

Figura 19 – Um vetor de 3 dimensões de sequências temporais.



Fonte – CHOLLET (2018)

Na Figura 19, o primeiro eixo do vetor diz respeito a quantidade de amostras (*samples*) disponíveis no conjunto de dados - geralmente não representam o conjunto total de da-

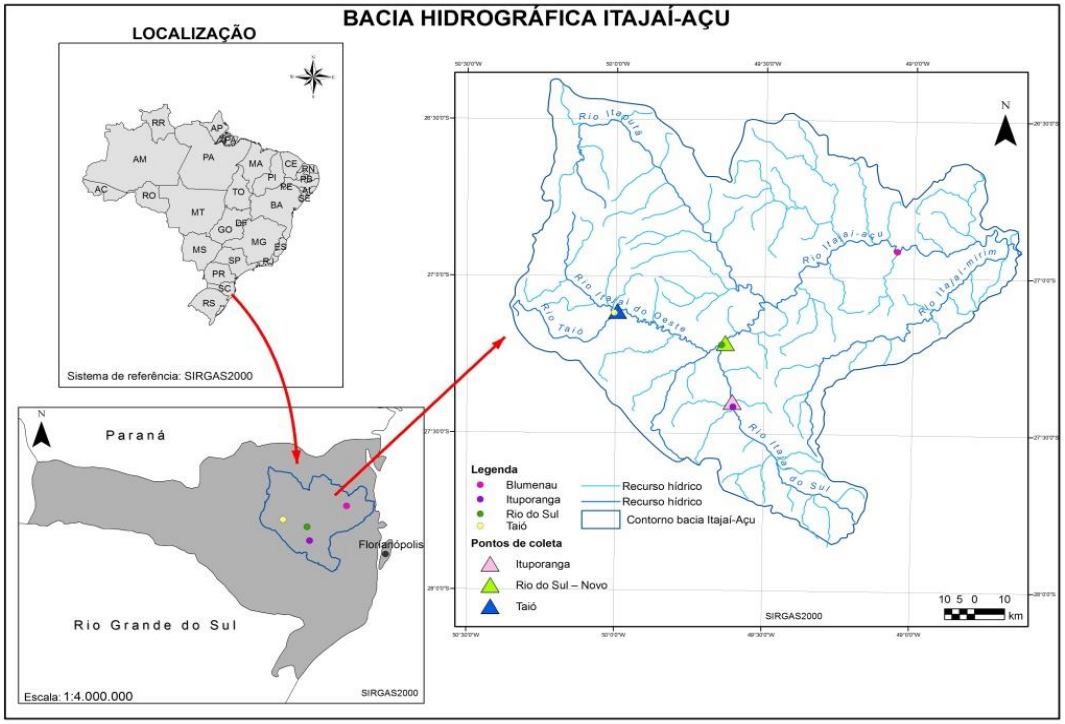
dos, porém quantidades menores (*Batchs*) devido ao uso do *Mini-batch Backpropagation*. Cada amostra do primeiro eixo representa uma sequência de observações temporais no segundo eixo (*Timesteps* - sequências temporais), onde cada sequência temporal possuí n atributos presentes no terceiro eixo do vetor (*features*).

Desta forma, o vetor de 3 dimensões possuí o formato (*features*, *timesteps*, *samples*). Sendo mandatório o uso dessa representação para os dados antes de serem submetidos as RNNs, exceto para as redes CNNs e MLP. Por último deve-se salientar que as sequências temporais (*timesteps*) tiveram um papel fundamental no treinamento das RNAs aqui propostas, pois também foi um parâmetro ajustado durante o treinamento dos modelos.

2.4 ÁREA DE ESTUDO

A bacia hidrográfica do rio Itajaí está situada na vertente atlântica do Estado de Santa Catarina, entre as coordenadas 26°22’ e 27°53’ de latitude sul e, 48°30’ e 50°22’ de longitude oeste (SOARES; TEIVE, 2015). O Vale do Rio Itajaí Açu encontra-se no Nordeste do Estado de Santa Catarina (Figura 20) e a bacia hidrográfica que o constitui é a maior do Estado, ocupando aproximadamente 15.000 km^2 (ESPÍNDOLA; NODARI, 2013).

Figura 20 – Vale do Itajaí - SC.



Fonte – Soares e Teive (2015)

Segundo Espíndola e Nodari (2013) os rios Itajaí do Sul, Itajaí do Oeste e Itajaí do Norte, formadores da bacia, descem a partir de suas nascentes, no Planalto Catarinense traçando cursos tortuosos e de alta declividade, que afluem, em direção aos Rios Itajaí Açu e Itajaí mirim.

O maior curso de água da bacia é o rio Itajaí Açu, formado pela encontro dos rios Itajaí do Oeste e Itajaí do Sul, no município de Rio do Sul (SOARES; TEIVE, 2015). Localizado na mesorregião do Vale do Itajaí entre a Serra do Mar e a Serra Geral, Rio do Sul é composto por uma população de pouco mais de 70 mil habitantes e ocupa uma área de aproximadamente 260 km^2 (IBGE, 2019).

A cidade é conhecida pelo longo histórico de inundações que a acomete e, de acordo com Cordero, Momo e Severo (2011), a primeira inundação documentada data do ano de 1911, quando foi registrado um pico de 12.60m. Além desta, diversas outras inundações atingiram o município, sendo as mais catastróficas durante a década de 1980 - em 1983, atingindo um pico de 13,58m e em 1984, alcançando uma cota de 12,80m (CORDERO; MOMO; SEVERO, 2011).

O ano de 2011 também foi considerado como um dos mais catastróficos para a cidade de Rio do Sul (Figura 21). De acordo com Espíndola e Nodari (2013), naquele mesmo ano foram registrados duas inundações seguidas: em Agosto o nível do rio atingiu um pico de 8,76m e em Setembro um pico de 12,98m.

Figura 21 – Inundação de 2011 do rio itajaí Açu em Rio do Sul.

Fonte – Dancuco (2011)

As inundações de 2011 causaram diversos prejuízos de ordem econômica e social. Segundo Silva e Souza (2016) só em prejuízos econômicos públicos estima-se que foram gastos R$ 3.468.796,97; em relação aos prejuízos econômicos privados foram estimados R$ 101.423.824,00 em prejuízos totais; além disso foram alocados 800 profissionais para restaurar a ordem e atenuar os prejuízos das inundações que atingiram mais de 50 mil pessoas da região. De acordo com Laurentino (2016) estima-se que no total foram gastos mais de 280 milhões.

Ao todo, desde 1911 até 2018 foram registradas 61 inundações na cidade de Rio do Sul (DEFESA CIVIL, 2019). Deu-se, portanto, a escolha deste local como área de estudo devido a seu histórico de ocorrências de inundações.

3 TRABALHOS CORRELATOS

No decorrer deste trabalho realizou-se uma revisão bibliográfica da literatura com o objetivo de identificar o estado da arte de modelos e métodos utilizados para a previsão de inundações. Através de tal foi possível identificar as principais características utilizadas nas previsões, tais como: variáveis utilizadas nos modelos; arquiteturas de Redes Neurais utilizadas; tempo de antecedência e métricas de erro para análise de qualidade da previsão.

A posteriori, quatro trabalhos foram selecionados como componentes estruturais da pesquisa aqui proposta e serão aclarados nas seções a seguir.

3.0.1 TRABALHOS SELECIONADOS NA REVISÃO BIBLIOGRÁFICA

Na Tabela 1 são expostos comparativamente as quatro pesquisas de previsão de inundações selecionadas durante a revisão bibliográfica. As informações dispostas descrevem as técnicas utilizadas, as variáveis aplicadas aos modelos, o horizonte de previsão e as métricas utilizadas para avaliar a qualidade da previsão de inundações.

Tabela 1 – Trabalhos selecionados na revisão bibliográfica.

Referência	Técnica utilizada	Variáveis	Tempo de antecedência	Métrica de Erro
Soares (2014)	MLP e RBF	Nível do Rio e Precipitação	6 Horas	NSE; RMSE e MAPE
Liu, Xu e Yang (2017)	SVN; BPNN; RBFNN; ELM; SAE-BP e SAE-BP+Kmeans	Vazão e precipitação	6 Horas	MSE e DC
Hu et al. (2018)	LSTM e MLP	Pico de descarga e precipitação	1-6 horas	R^2 ; RMSE; NSE; MAE; ET e EQ_p
Sit e Demir (2019)	GRU e Fully Connected Network	Precipitação e nível do rio.	Tempo real	MSE

Fonte: Elaborado pelo autor.

3.0.2 Previsão De Cheias Do Rio Itajaí Açu Utilizando Redes Neurais Artificiais

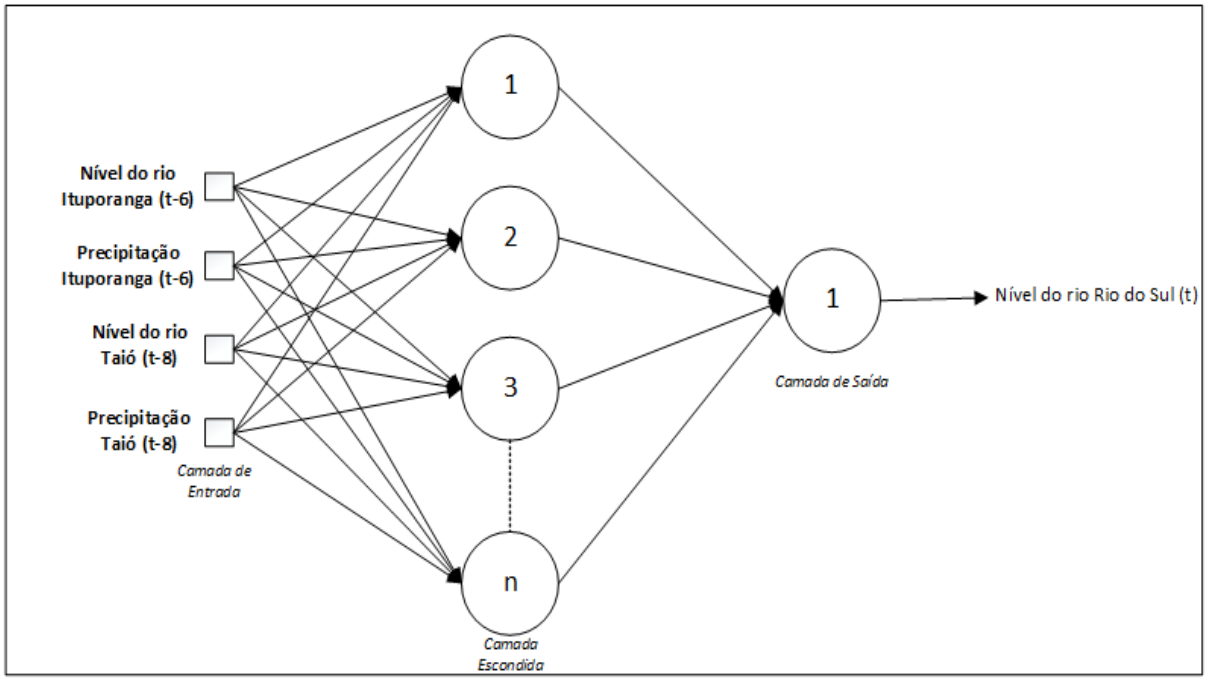
Na pesquisa de Soares (2014) foram elaborados dois modelos de RNAs - *Mullti-layer Percepton (MLP)* e *Radial Basis Function (RBF)* - para a previsão de cheias do Rio Itajaí Açu em um ponto que se encontra no município de Rio do Sul. As previsões foram realizadas com um horizonte de seis horas de antecedência. O trabalho ainda realiza uma comparação do

desempenho das RNAs para identificar qual o modelo e tipo de RNA é mais aderente a este tipo de previsão. Tal comparação é realizada através de métricas de desempenho, às quais os resultados das redes foram submetidos, sendo elas: Coeficiente de Eficiência de Nash e Sutcliffe (NSE), *Root Mean Square Error (RMSE)* e *Mean Absolute Square Error (MAPE)*.

As RNAs elaboradas por Soares (2014) possuem quatro entradas: uma camada oculta com neurônios que variam de 3 até 10 unidades e um neurônio na camada de saída. Essa configuração com apenas uma camada oculta e até 10 neurônios foi selecionada após a realização de alguns ensaios que mostraram que o uso de mais de uma camada oculta e mais de 10 neurônios não produziram melhoras significativas nos resultados, mas apenas aumentava o tempo de treinamento demasiadamente, tal como observado pelo autor.

O modelo arquitetural da RNA construída por Soares (2014) é ilustrado na Figura 22. Conforme se pode observar, na camada de entrada o autor utiliza como variáveis dados relativos ao nível do rio e à precipitação dos municípios de Ituporanga e Taió, ambos próximos a Rio do Sul, e na camada de saída o nível do rio de Rio do Sul.

Figura 22 – Modelo de RNA proposto por Soares.



Fonte – Soares (2014)

Os dados relativos aos municípios destacados abrangem um período que vai de Agosto de 2005 até Fevereiro de 2014 e foram selecionados segundo a análise de Soares (2014),

em função de uma característica importante da bacia hidrográfica, que é o tempo que a água leva para ir dos postos de medição a montante do ponto de interesse (ou seja, Ituporanga e Taió) até a área selecionada para realizar as previsões (Rio do Sul). Esse tempo é de 6 horas a partir do posto localizado no município de Ituporanga até Rio do Sul e de 8 horas a partir do posto no município de Taió até Rio do Sul.

Os experimentos realizados por Soares contaram com um total de 430 configurações de RNAs MLP e RBF. Dentre as redes MLP a que obteve melhor desempenho foi a RNA configurada com 9 neurônios na camada oculta, função de ativação logística (Sigmoide) e algoritmo de treinamento Gradiente descendente com *momentum*, obtendo-se assim um NSE de 0.9779; RMSE de 0.0201; e MAPE 5.625. Já a melhor configuração de RNAs RBF foi a rede com spread de 0.2, função de ativação radbas e 69 neurônios na camada oculta, que logrou obter um NSE de 0.9633; RMSE de 0.383 e MAPE de 4.7089.

Ao comparar as duas RNAs Soares (2014) constatou que a MLP foi a que obteve melhor desempenho, pois exibiu resultados superiores à RBF em duas métricas, NSE e RMSE, mostrando apenas um desempenho inferior na métrica MAPE. Desta forma, o autor concluiu que a rede neural do tipo MLP foi a que mais se adequou ao problema de previsão de inundação para o local de estudo selecionado, provendo um horizonte de previsão de seis horas de antecedência que, segundo a visão do autor, são úteis para tomadas de decisão pela defesa civil.

3.0.3 A Flood Forecasting Model based on Deep Learning Algorithm via Integrating Stacked Autoencoders with BP Neural Network

A pesquisa de Liu, Xu e Yang (2017) propõe um modelo de rede neural profunda através da integração dos modelos *Stack Autoencoders (SAE)* e *Back Propagation Neural Network (BPNN)*, para realizar a previsão de vazão por intermédio de dados a montante do ponto de interesse e consequentemente prever inundações. A combinação dos modelos melhora a capacidade de simulação de não linearidade ao classificar todos os dados em várias categorias pelo módulo de agrupamento K-means.

A abordagem proposta é comparada respectivamente com o modelo *Support Vector Machines (SVM)*, o modelo *Back Propagation Neural Network (BPNN)*, o modelo *Neural Network Radial Basis Function (RBF)* e o modelo *Extreme Learning Machine (ELM)*. Os resultados experimentais mostram que o algoritmo integrado do SAE-BP tem um desempenho superior sobre os outros modelos.

O modelo então proposto é utilizado para realizar a previsão da vazão com 6 horas

de antecedência através de dados coletados concernentes a anos anteriores ao da pesquisa. Para tanto os autores coletaram amostras de dados de vazão e precipitação dos períodos de cheia que abrangem o período que vai do ano de 1998 ao de 2010, com uma taxa de amostragem dos dados de uma hora. A quantidade de amostras é de 6482 dentro do intervalo que transcorre entre 1989 e 2008 e as 1676 amostras restantes dos dois anos seguintes.

A configuração do modelo proposto possui duas camadas escondidas com 20 e 15 neurônios respectivamente e um valor de 4 para o modelo com Kmeans. Como fatores de previsão os autores utilizaram amostras de dados concernentes à vazão e à precipitação provenientes de seis estações a montante do ponto de interesse, com um período de antecedência de 4 horas para as vazões e de 4 a 7 horas para as precipitações, intervalos estes que serviram como dados de entrada nos modelos utilizados na pesquisa.

Os resultados submetidos nos tais modelos foram avaliados através das métricas de desempenho *Mean Squared Errors (MSE)* e *Citerion and Deterministic Coefficient (DC)* e então comparados para verificar aquele que obteve o melhor desempenho. A Tabela 2 exibe o resultado das comparações dos modelos.

Tabela 2 – Comparativo do resultado dos modelos de predição.

Modelo	MSE	DC
SVM	4930	0.816
BP neural network	6999	0.707
RBF neural network	7295	0.695
Extreme Learning Machine	6807	0.715
SAE-BP	3644	0.848
SAP-BP+Kmeans	2877	0.88

Fonte: Liu, Xu e Yang (2017).

Conforme se pode observar pela Tabela 4 o modelo SAE-BP é o que exibe o melhor resultado. Em relação aos outros modelos os autores observaram que o SVM e o RBF demonstram um melhor desempenho em seções com baixa vazão, enquanto que a rede neural BP atinge melhor desempenho em seções com alta vazão. Já o modelo ELM não é estável para seções com desvios. Os autores então concluem através desses modelos tradicionais que SAE-BP faz previsões mais precisas, especialmente para seções de alta vazão.

3.0.4 Deep Learning with a Long Short-Term Memory Networks Approach for Rainfall-Runoff Simulation

O objetivo do estudo de Hu et al. (2018) é a construção de modelos de tempo real orientados a dados que permitam simular e prever o escoamento das chuvas a partir dos dados disponíveis. Esta modelagem baseada em dados analisa as relações entre as séries temporais de precipitação e escoamento.

Para atingir tal objetivo os autores empregam o modelo de rede neural tradicional *Multilayer Percepton (MLP)* e o modelo de rede neural profunda *Long Short Term Memory (LSTM)* para simular o processo chuva-vazão com base nos eventos de inundação de 1971 a 2013 na bacia do Rio Fen - localizado na província de Shanxi, no norte da China - monitorados através de 14 estações pluviométricas e uma estação hidrológica na bacia hidrográfica de Jingle.

Os dados experimentais são de 98 eventos de chuva-vazão do período em questão, dentre os quais 86 eventos de precipitação pluvial são usados como conjunto de treinamento e o restante como conjunto de testes. Os resultados, segundo os autores, mostraram que ambas as redes são adequadas para modelar o escoamento da chuva e se mostram melhores que modelos conceituais e físicos. Os modelos LSTM superam os modelos de MLP com os valores de R^2 e NSE acima de 0,9, respectivamente.

Os autores explicam que embora o processo de inundação seja de difícil simulação, os modelos MLP e LSTM conseguiram emular o fenômeno satisfatoriamente. Ao comparar com a simulação de pico de descarga, o valor obtido através do modelo MLP foi sempre maior que os dados observados. No baixo valor da simulação de descarga, os valores modelados das RNAs mostraram flutuações anormais.

A Tabela 3 exibe a comparação entre os modelos MLP e LSTM quando submetidos a previsão de vazão com 1 hora de antecedência. Nesta comparação, seis métricas de desempenho são utilizadas para avaliar os modelos, dentre elas: R^2; RMSE; NSE; MAE; *Error of Time to Peak Discharge* ET_p; e *Error of Peak Discharge* EQ_p.

Tabela 3 – Comparação da performance dos modelos MLP E LSTM para previsão de vazão com 1 hora de antecedência. Na calibração têm-se 86 eventos de inundações e na validação 12 eventos de inundações.

Eventos	Modelos	R^2	RMSE (m^3s^{-1})	NSE	MAE (m^3s^{-1})	$ET_p(h)$	EQ_p
Calibração							
Séries de 86 eventos	MLP	0.81	124.21	0.83	47.23	5.4	12%
	LSTM	0.95	45.12	0.97	12.4	2.6	4%
Validação							
Séries de 12 eventos	MLP	0.83	35.6	0.83	23.6	3.7	14%
	LSTM	0.96	12.4	0.96	6.3	1.4	3%

Fonte: Hu et al. (2018).

A tabela 3 demonstra que todos os valores das métricas R^2 e NSE estão acima de 0.95 nos resultados da modelagem LSTM nos períodos concernentes aos dados de calibração e validação. Ao comparar com o modelo MLP, Hu et al. (2018) observam que no modelo LSTM os valores das métricas RMSE, MAE, ET_p e EQ_p são menores que os valores do modelo MLP, especialmente o valor de EQ_p que é 4 vezes menor no modelo LSTM, desta forma apresentando um desempenho melhor na simulação do escoamento de chuva. O valor R^2 superior na rede LSTM indica, ademais, que este modelo reflete de maneira satisfatória a relação entre a descarga observada e simulada. Tais resultados, ainda segundo análise dos autores, comprovam que o modelo LSTM simulou com precisão suficiente.

Os autores também realizaram experimentos variando o tempo de antecedência de 1 a 6 horas nos modelos MLP E LSTM. No geral o modelo LSTM apresentou melhores resultados de simulação do que o modelo MLP em diferentes tempos de espera. Na etapa de calibração e validação, os valores dos critérios de desempenho no modelo LSTM são todos melhores do que os obtidos pelos modelos de RNA.

Ao comparar com diferentes situações de tempo de antecedência, Hu et al. (2018) observaram que os valores de R^2 e NSE diminuíam conforme se aumentava o tempo de espera. Os valores de RMSE, MAE e ET_p não mostraram mudanças significativas. Os valores do EQ_p na LSTM foram os menores com tempo antecedência de1 hora, enquanto o modelo MLP apresentou desempenho ruim para o intervalo de 6 horas, exibindo valores de R^2 e NSE próximos a 0.7. Mesmo que a capacidade de predição de LSTM estivesse induzindo com grande lead time, os valores de R^2 e NSE ainda estavam acima de 0,8. Comparado com o modelo MLP, LSTM também possui baixo valor de ET_p e EQ_p.

Os autores então concluem, através de tais resultados obtidos, que o modelo LSTM demonstrou possuir o melhor desempenho na previsão de pico de descarga em cada evento de inundação e que, portanto, o modelo em questão é o mais adequado para a modelagem do escoamento de chuva e deve ser preferido na utilização em pesquisas de hidrologia.

3.0.5 Decentralized Flood Forecasting Using Deep Neural Networks

A pesquisa de Sit e Demir (2019) propõe um conjunto de dados de referência de previsão de inundação para futuras aplicações em aprendizado de máquina, bem como uma abordagem escalonável para prever o estágio do rio para pontos de pesquisa individuais em rios. A abordagem leva em consideração os dados do estágio histórico dos pontos de pesquisa nos locais selecionados a montante, bem como os dados de precipitação. Essa abordagem é descentralizada e não precisa de dados históricos de pontos de pesquisa não relacionados e pode ser usada em tempo real.

Uma ramificação de Redes Neurais Recorrentes (RNNs) em particular, as redes *Gated Recurrent Unit (GRU)*, são utilizadas ao longo da pesquisa, demonstrando resultados satisfatórios ao serem aplicadas ao estado de Iowa (Estados Unidos da América) como prova de conceito. Além disso, para fins de comparação, o modelo de rede neural profunda *Fully-connected network* também foi empregado.

Os dados coletados são provenientes dos sensores do Iowa *Flood Center (IFC)* e do US *Geological Survey (USGS)*, atuantes nos rios do estado de Iowa e compreendem 3 conjuntos diferentes. O primeiro conjunto de dados consiste na altura do medidor dos sensores USGS e UFC. O segundo conjunto de dados é o produto de precipitação do radar NOAAs Stage IV e o último conjunto de dados é a informação de metadados e sensores da IFC. A USGS possui 201 sensores de altura de medição no estado de Iowa, cada um fornecendo medições de altura manométrica com 15 minutos de resolução temporal. Os conjuntos pertencem ao período entre Janeiro de 2009 e Junho de 2018 - 80% do conjunto foi utilizado no treinamento, os 20% restantes para testes.

Para realização dos testes dois conjuntos de dados foram utilizados, um menor e um maior. Ambos foram submetidos à rede *Fully-connected network* e apenas o maior à rede GRU. Uma vez que os resultados foram por sua vez submetidos à métrica de desempenho MSE (Tabela 4), os autores observaram que ao aumentar-se os dados não se melhorava significativamente o desempenho do modelo *Fully-connected network*. No entanto, Sit e Demir (2019)

observaram que a escolha do modelo tem efeitos importantes no desempenho geral do teste do modelo. Considerando o MSE relatado e a semelhança entre as medições e as previsões feitas pelo modelo baseado em GRU, aqueles autores concluíram que o desempenho geral das redes neurais com a abordagem descentralizada proposta é aceitável e que os modelos baseados em RNN fazem melhores escolhas de arquitetura para tarefas de previsão de inundação.

Tabela 4 – Pontuação Mean Sequared Error para o conjunto de testes.

	Conjunto Maior	Conjunto menor
Fully-connected	189.43	189.87
GRU	154.65	

Fonte: Adaptado de Sit e Demir (2019)

Por último os autores afirmam que os modelos propostos em sua pesquisa podem ser usados para apresentar resultados de previsão mais aprimorados em sistemas de informações operacionais, juntamente com previsões de modelos hidrológicos avançados. Os resultados obtidos corroboram o argumento de que a abordagem de previsão de inundação descentralizada baseada em redes neurais artificiais para o estado de Iowa faz previsões muito próximas das medidas reais.

4 DESENVOLVIMENTO

Este capítulo tem por objetivo detalhar os dados disponíveis, como foram obtidos e o pré-processamento realizado ao qual os tais foram submetidos. Em seguida os conceitos referentes à validação cruzada empregada no conjunto de dados são elucidados.

Também elucidar-se-á acerca dos recursos utilizados nas simulações, tais como equipamentos e softwares. Por conseguinte os algoritmos de treinamento e os modelos de RNAs empregados, incluindo os parâmetros alterados durante o treinamento são aclarados. Por último são descortinadas as métricas de desempenho utilizadas para avaliar o desempenho de aprendizado e generalização dos modelos em questão, e também como foi realizada a definição da melhor configuração de RNA.

4.1 DADOS DISPONÍVEIS

Os dados observados na bacia do rio Itajaí Açu são coletados e registrados pela rede telemétrica da Agência Nacional de Águas (ANA), operada pela Epagri. Os dados utilizados nesta pesquisa foram obtidos a partir de três postos de coleta, localizados nas cidades de Ituporanga e Taió, cidades a montante do ponto de interesse; e Rio do Sul local escolhido para realizar as previsões.

A escolha dos pontos de coleta em questão para se realizar as previsões, foram realizadas com base no trabalho de Soares (2014). A seguir nas tabelas 5 6 e 7 são exibidos os dados referentes às estações de coleta localizadas nos municípios supracitados.

Tabela 5 – Dados da estação de Rio do Sul.

Código Fluviométrico	83300200
Código Pluviométrico	2749039
Nome	Rio do Sul – Novo
Bacia	Atlântico (8), trecho Sudeste
Sub-bacia	Rio Itajaí-Açu (83)
Rio	Rio Itajaí-Açu
Latitude	-27 12’ 42.84”
Longitude	-49 37’ 54.12”
Altitude	350

Fonte: ANA (2019).

Tabela 6 – Dados da estação de Taió.

Código Fluviométrico	83030000
Código Pluviométrico	2750014
Nome	Taió
Bacia	Atlântico (8), trecho Sudeste
Sub-bacia	Rio Itajaí Açu (83)
Rio	Rio Itajaí do Oeste
Latitude	-27 6’ 47.16”
Longitude	-49 59’ 39.84”
Altitude	360

Fonte: ANA (2019).

Tabela 7 – Dados da estação de Ituporanga.

Código Fluviométrico	83250000
Código Pluviométrico	2749002
Nome	Ituporanga
Bacia	Atlântico (8), trecho Sudeste
Sub-bacia	Rio Itajaí Açu (83)
Rio	Rio Itajaí do Sul
Latitude	-27 23’ 54.96”
Longitude	-49 36’ 20.88”
Altitude	370

Fonte: ANA (2019).

As amostras de dados a serem submetidas as redes neurais propostas são constituídas de quatro atributos de entrada e um de saída. Os atributos de entrada são dados de precipitação e nível do rio dos postos de coleta de Taió e Ituporanga, já o atributo de saída é o nível do rio do posto de coleta de Rio do Sul.

A base de dados usada no treinamento, validação e teste das RNAs consiste de dados coletados durante o período de 29/06/2005 à 22/10/2018. Estes dados estão disponíveis no portal de telemetria do Sistema Nacional de Informações Sobre Recursos Hídricos (SIRH) da ANA e podem ser obtidos nos formatos *txt* e *xls*. Neste trabalho optou-se por obter os arquivos com os dados no formato *xls*. A seguir a Figura 23 exibe a forma em que os dados estão dispostos nos arquivos obtidos.

Figura 23 – Disposição dos dados em todos os arquivos obtidos no Hidro Telemetria da ANA.

Código	Data e Hora	Chuva horária	QChuva	Nível adotado	QNível	Vazão	QVazão
83300200	22/10/2018 23:45:00	0	0	235	0	117,95	
83300200	22/10/2018 23:30:00	0	0	236	0	118,79	
83300200	22/10/2018 23:15:00	0	0	237	0	119,63	
83300200	22/10/2018 23:00:00	0	0	236	0	118,79	
83300200	22/10/2018 22:45:00	0	0	237	0	119,63	

Fonte – Acervo do autor.

Os dados brutos estavam organizados em 3 arquivos, cada um referente a um ponto de coleta das estações de interesse da pesquisa. Os dados coletados durante o período de 29/06/2005 a 11/11/2011 possuem uma resolução horária, ou seja, eram observados e registrados a cada 1 hora; a partir de 11/11/2011 as observações passaram a ser registradas a cada 15 minutos.

De acordo com operadores da ANA os atributos com o prefixo Q são gerados automaticamente ao se exportar os dados no formato *xls* e portanto devem ser ignorados. Além disso os arquivos possuem um atributo com informações coletadas sobre a vazão dos pontos de interesse; atributo este que não foi considerado como entrada das RNAs, pois também não foi utilizado na pesquisa de Soares (2014).

As unidades de medida dos atributos chuva, nível adotado e vazão são respectivamente: mm, cm e m^3/s Após a obtenção dos arquivos com os dados, uma série de tratamentos foram realizados de modo a torná-los aptos a serem submetidos para treinamento nas redes neurais, sendo os tais descritos a seguir.

4.2 TRATAMENTO DOS DADOS

A primeira ação tomada na preparação do conjunto de dados após sua obtenção foi dispô-los em uma única planilha no formato *csv*. Os atributos foram organizados em cinco colunas, onde cada linha é uma amostra de treinamento. Em seguida realizou-se um atraso temporal de 6h e 8h nos dados concernentes às estações de Ituporanga e Taió.

A decisão de realizar tal atraso temporal nos dados foi tomada com base no trabalho de Soares (2014), pois segundo o autor os pontos das estações em questão estão a montante do ponto de interesse – Rio do Sul – fato este que além de permitir realizar uma previsão mais precisa, devido onda de deslocamento da água nas estações supracitadas serem de 6h e 8h, também permite realizar uma previsão com 6 horas de antecedência.

O quadro 4 a seguir ilustra a disposição dos dados com o atraso temporal nas colunas em questão:

Tabela 8 – Disposição dos dados para o treinamento das RNAs.

Saída	Entradas			
Nível do Rio Rio do Sul	Precipitação Taió	Nível do Rio Taió	Precipitação Ituporanga	Nível do Rio Ituporanga
t	t-8	t-8	t-6	t-6

Fonte: Elaborado pelo autor.

Com a nova disposição dos dados em uma única tabela, na sequência estes passaram por uma etapa de pré-processamento com o intuito de obter melhorias no desempenho do treinamento.

4.3 PRÉ-PROCESSAMENTO DOS DADOS

A primeira ação de pré-processamento realizada após a disposição dos dados com o atraso temporal entre as variáveis de entrada em relação a variável de saída, foi a remoção de linhas que continham dados faltantes, pois em alguns períodos que podem variar de semanas ou até mesmo meses não existem coletas de dados.

A ação seguinte consiste da remoção de amostras duplicadas, visando evitar um aprendizado tendencioso. Após essa etapa a base de dados ficou com 61095 amostras, que em seguida passaram por um processo de normalização.

4.3.1 Normalização dos dados

De acordo com Géron (2017) a etapa do pré-processamento mais importante a se realizar sobre os dados é o escalonamento de atributos, pois algoritmos de aprendizado de máquina não conseguem ter um bom desempenho no aprendizado quando atributos passados como entrada se encontram em escalas diferentes.

O processo de escalonamento além de transformar os atributos para uma mesma escala de valores, também conforme complementam Silva, Spatti e Flauzino (2010), desloca os valores para o mesmo intervalo numérico das funções de ativação das camadas intermediárias, tipicamente representadas pelas funções *Sigmoid*, Tangente Hiperbólica e *ReLU*, assim permitindo evitar regiões de saturação das tais funções.

Conforme explica Géron (2017) existem duas formas comuns utilizadas para transformar os atributos para uma mesma escala de valores, sendo elas: escalonamento *Min-Max* – também conhecido como normalização – e a padronização – do inglês *standardization*.

De acordo com Raschka e Mirjalili (2017) normalmente a normalização escalona os valores para o intervalo numérico $[0, 1]$, sendo este um caso especial do escalonamento *Min-Max*. Desta forma para normalizar os dados, deve-se aplicar a equação 4.1 para cada atributo do conjunto de amostras de forma individual, em cada amostra em particular, obtendo-se assim o novo valor normalizado $x^{(i)}_{norm}$.

$$x^{(i)}_{\text{norm}} = \frac{x^{(i)} - x_{\min}}{x_{\max} - x_{\min}} \tag{4.1}$$

Onde $x^{(i)}$ é uma amostra em particular x_{min} é o menor valor do atributo sendo considerado e x_{max} o maior valor.

Ainda de acordo com os autores, a padronização dos dados consiste de uma transformação para centralizar os atributos de modo a se ter uma distribuição dos dados com média 0 e desvio padrão 1, ou seja, consiste de uma transformação na qual o conjunto de valores dos atributos assumam uma forma parecida com a distribuição normal.

O processo para se realizar tal transformação é expresso pela equação 4.2:

$$x^{(i)}_{std} = \frac{x^{(i)} - \mu_x}{\sigma_x} \tag{4.2}$$

Onde $x^{(i)}$ é uma amostra em particular μ_x é a média de um atributo em particular e σ_x é o desvio padrão correspondente.

Após o treinamento, ao se realizar uma predição com o modelo de rede neural, há também uma necessidade de realizar operações inversas de escalonamento de modo a permitir que os dados possam ser interpretados, ou seja, a saída da rede deve ser convertida para os valores que representam os domínios reais da aplicação.

O processo inverso de escalonamento para o método de normalização pode ser realizado através da equação 4.3:

$$x^{(i)} = x^{(i)}_{\text{norm}} \cdot (x_{\max} - x_{\min}) + x_{\min} \tag{4.3}$$

E o processo de inverso da padronização pode ser realizado através da equação 4.3:

$$x^{(i)} = x^{(i)}_{\text{std}} \cdot \sigma_x + \mu_x \tag{4.4}$$

Neste trabalho os dois métodos de escalonamento foram utilizados e para cada método os dados foram transformados para ambos os intervalos numéricos $[-1, 1]$ e $[0, 1]$. Após realizados diversos ensaios com as redes neurais, utilizando-se ambos os métodos, os desempenhos foram comparados e o método no qual as redes mostraram o melhor desempenho, ou seja, melhor se adaptaram, foi o método de normalização *Min-Max* com os dados deslocados para o intervalo $[0, 1]$.

Através do método escolhido no intervalo em questão as redes mostravam progresso no aprendizado, enquanto que em outras configurações os dados não eram representativos para as redes, isso é, as redes neurais testadas não conseguiam extrair recursos dos dados de modo que o aprendizado se tornasse possível, ou seja, as redes não conseguiam encontrar padrões nos dados de modo a aprender com os tais.

Para deslocar-se os dados para os intervalos numéricos destacados, as equações passaram por uma pequena modificação de modo a produzir resultados dentro dos tais intervalos. Devido a equação eleita a mais propícia para normalizar os dados já originalmente produzir resultados dentro do intervalo eleito, e por motivos de brevidade, as equações modificadas foram omitidas de serem apresentadas nesta seção.

Cabe ainda salientar que mesmo que o pré-processamento tenha sido feito corretamente, os dados ainda podem conter algumas imperfeições. Neste sentido Silva, Spatti e Flauzino (2010) explicam que a qualidade do conjunto de treinamento disponível pode ser afetada sensivelmente em função da quantidade de ruídos presentes nas amostras e da presença de amostras espúrias (*outliers*), dentre outros. Uma das técnicas comumente utilizada para atenuar o efeito deste tipo de amostra é a validação cruzada.

4.4 VALIDAÇÃO CRUZADA

De acordo com Silva, Spatti e Flauzino (2010), o objetivo da validação cruzada é avaliar a qualidade de generalização de um modelo de aprendizado de máquina ao aplicá-lo a um conjunto de dados diferente daquele utilizado durante o processo de treinamento. Desta forma é possível estimar o erro da previsão do modelo através da divisão do conjunto de dados em subconjuntos que são utilizados para validá-lo e testá-lo.

Existem diferentes métodos de validação cruzada, dentre as opções disponíveis dois métodos foram utilizados nesta pesquisa: O método de amostragem não aleatória - respeitando-se a ordem temporal das observações; e o método de amostragem aleatória. Em ambos os mé-

todos o conjunto total de amostras disponíveis foi dividido em três subconjuntos: treinamento, validação e teste.

O conjunto de treinamento é utilizado durante o processo de aprendizado, enquanto que o conjunto de validação é utilizado para fornecer uma avaliação imparcial do modelo ajustado no conjunto de treinamento ao se ajustar os parâmetros da rede neural. No entanto, a avaliação se torna mais tendenciosa à medida que a habilidade no conjunto de dados de validação é incorporada na configuração do modelo, desta forma o afetando de maneira indireta, e por isso emprega-se o conjunto de testes, que é então utilizado para fornecer uma avaliação imparcial do modelo final adequado ao conjunto de dados do treinamento (SHAH, 2017).

A amostragem não aleatória dos subconjuntos foi realizada respeitando-se a ordem temporal das observações conforme a literatura recomenda ao se trabalhar com séries temporais. Além disso conforme fundamentado na seção 2.3.1, sabe-se que as redes neurais recorrentes utilizam informações de observações passadas para prever observações futuras e portanto a ordem das observações nos conjuntos de dados deve ser respeitada ao se utilizar este tipo de topologia.

Dentro deste contexto, o método de múltipla divisão (*Multiple Train-Test Splits*) foi empregado para se realizar a separação do conjunto total de amostras nos subconjuntos de treino, validação e teste. Nesta estratégia o processo de separação das amostras é repetida por diversas vezes se separando os dados em diferentes frações e em cada separação um modelo diferente é treinado e avaliado. Desta forma é possível se obter uma estimativa mais robusta do desempenho esperado do método e configuração escolhidos com base nos dados de validação (BROWNLEE, 2016).

Os subconjuntos foram divididos em três grupos de frações diferentes 50%, 25% e 25%; 60%, 20% e 20%; 70% 15% 15%; e para cada grupo de frações um conjunto de testes foi realizado de modo a escolher o grupo de divisão dos dados que quando submetidos aos modelos de RNAs propostos neste trabalho apresentassem um desempenho com menor erro em relação aos outros grupos de frações.

Após a realização dos testes o conjunto de frações em que os modelos apresentaram um menor índice de erro foi o conjunto com 50% dos dados separados para o subconjunto de treinamento, 25% para o de validação e os 25% restantes para o subconjunto de testes. Entretanto embora tenha se encontrado a melhor partição possível os modelos ainda assim eram incapazes generalizar ao serem submetidos ao conjunto de validação e testes, ou seja, não havia

progresso na minimização do índice de erro e portanto os dados não se mostravam representativos para as redes neurais utilizadas.

Uma provável hipótese para o problema da generalização pode estar atrelada ao pequeno volume de dados disponível para as RNAs profundas, assim sugerindo que um modelo menos complexo e de uso mais geral fosse utilizado. Neste caso um conjunto de testes foram realizados utilizando-se o modelo MLP e os resultados foram ainda piores em relação aos resultados obtidos nos modelos recorrentes, desta vez as RNAs nem ao menos tinham habilidade de aprender com os dados separados para o treinamento, consequentemente também não poderiam generalizar nos outros subconjuntos.

Na sequência devido aos resultados insatisfatórios, o método de amostragem aleatória foi utilizado e a mesma estratégia de múltipla divisão foi realizada com as mesmas proporções já utilizadas na amostragem não aleatória, desta vez o grupo de proporções que se mostrou mais adequado foi o de 70%, 15% e 15%.

Devido ao método de amostragem aleatória embaralhar os dados aleatoriamente antes de se realizar a separação dos subconjuntos, consequentemente a ordem temporal das observações não é mais mantida e portanto as redes neurais recorrentes deixam de ser adequadas, devido as razões já destacadas a priori.

Levando-se essas observações em consideração, novamente um conjunto de configurações de redes MLP foram testadas e novamente os resultados foram similarmente insatisfatórios em relação as redes já testadas anteriormente. Na sequência mesmo em vista da inadequação de se utilizar as RNNs em subconjuntos em que a ordem temporal foi quebrada, novamente as redes recorrentes propostas foram testadas e os resultados obtidos foram satisfatórios. Além das redes apresentarem baixo índice de erro durante o aprendizado, também mostravam progresso na generalização quando submetidos aos conjuntos de validação e teste, desta forma as redes RNN se mostraram, ainda que com a ordem temporal quebrada, adequadas para serem utilizadas nesta pesquisa.

Por fim, cabe ainda salientar que a ordem temporal das observações não é quebrada em sua totalidade. Conforme é explicado na seção 2.3.4, para se alimentar as redes neurais recorrentes, os dados devem ser dispostos em um vetor de três dimensões no formato (amostras, sequências temporais, atributos). Sendo que o embaralhamento é realizado com respeito a dimensão das amostras, onde cada amostra corresponde a uma matriz de dimensões (sequências temporais, atributos), desta forma a ordem temporal em que as sequências são dispostas não é

alterada.

4.5 RECURSOS UTILIZADOS

A preparação e pré-processamento dos dados, criação de gráficos, bem como a modelagem, treinamento e testes das RNAs desenvolvidas para essa pesquisa, foram realizadas através da linguagem de programação *Python*, especificamente uma série de bibliotecas disponíveis para esta linguagem foram utilizadas, dentre as tais: *Pandas*, *Numpy*, *Scikit-Learn*, *Matplotlib* e *Keras*.

A obtenção e preparação do conjunto de dados ocorreram em um computador pessoal Intel *Core i5-8265U* com *8Gb* de memória Ram e sistema operacional *Ubuntu 18.04.3 LTS*. O ambiente de desenvolvimento e o interpretador para a linguagem *Python* utilizados foram respectivamente o editor de textos *Emacs 25.2.2*; e o interpretador *Python* versão *3.6.8*.

O pré-processamento, geração de gráficos, modelagem, treinamento e testes das RNAs foram realizadas através do ambiente colaborativo de aprendizado de máquina do Google (*Google Colaboratory*), disponível na nuvem. O ambiente de desenvolvimento utilizado foi o *Jupyter Notebook* [1] configurado com o interpretador para a linguagem *Python* na versão *3.6.8*. Além disso, sempre que disponíveis, os aceleradores de processamento disponíveis na plataforma (GPU e TPU) foram utilizados favoravelmente para acelerar o treinamento dos modelos.

Durante a modelagem das RNAs, diversas configurações foram experimentadas, variando número de camadas ocultas, quantidade de células e neurônios das camadas ocultas e funções de ativação. Dentre estas configurações, alguns parâmetros não foram alterados, como o algoritmo de treinamento e seus parâmetros internos por exemplo.

Nas subseções a seguir, todos os recursos utilizados, incluindo funções de ativação e algoritmos de treinamento são elucidados.

[1] O *Jupyter Notebook* é uma aplicação web de código aberto que permite criar e compartilhar documentos que contém código fonte ativo, equações, visualizações, e texto narrativo. Geralmente são utilizados para realizar limpeza e transformação de dados, simulação numérica, modelagem estatística, visualização de dados, aprendizado de máquina e etc (JUPYTER, 2019).

4.5.1 Bibliotecas

4.5.1.1 Pandas

Pandas é uma biblioteca criada para a linguagem de Programação Python para manipulação e análise de dados. Especificamente, oferece estruturas de dados e funcionalidades para manipulação de tabelas numéricas e séries temporais. A biblioteca é mantida por desenvolvedores voluntários e seu código fonte é aberto e disponibilizado sob a licença BSD (PANDAS, 2019).

Neste projeto a biblioteca foi especificamente utilizada para as seguintes tarefas:

- **Carregamento dos dados**: Inicialmente os dados foram carregados dos arquivos no formato *xls* para uma estrutura de dados fornecida pela biblioteca – estrutura essa similar a uma planilha eletrônica. Cada arquivo foi carregado de forma individual, embora representados pela mesma estrutura de dados, porém armazenados e referenciados por variáveis distintas.

- **Sumarização**: Através de métodos fornecidos pela estrutura de dados em questão, obteve-se uma sumarização dos dados com informações estatísticas objetivando verificar dados faltantes, *outliers* dentre outros. Além disso, também foram gerados gráficos de modo a realizar uma análise gráfica da sumarização.

- **Geração de datas e concatenação**: Como haviam datas faltantes nos três arquivos obtidos, as tais datas foram geradas como novas linhas de dados, porém sem dados nas colunas dos atributos. Essa ação objetivou deixar os três arquivos uniformemente com as mesmas datas para não haver discrepâncias ao se realizar o atraso temporal dos atributos. Após essa ação os atributos das três planilhas carregadas foram concatenadas em uma única estrutura de dados.

- **Atraso temporal e remoção de linhas com dados faltantes e outliers**: Com os atributos referentes aos três conjuntos de dados dispostos em uma única estrutura de representação, duas ações foram realizadas: Primeiro o atraso temporal dos atributos; e na sequência a remoção de dados faltantes, linhas que continham dados duplicados e *outliers*.

Após todas essas ações terem sido realizadas, ainda por viés da biblioteca em questão, a estrutura de dados foi persistida em disco no formato *csv* para ser carregada e utilizada a posteriori no *Google Colaboratory*, onde novamente os dados são carregados através do *Pandas*

e também convertidos para uma estrutura de dados do tipo vetor *Numpy*, antes de alimentarem as RNAs.

4.5.1.2 Numpy

A biblioteca Numpy é um pacote fundamental de computação científica da linguagem de programação *Python*. Dentre suas principais funcionalidades, *Numpy* oferece suporte para matrizes multidimensionais de larga escala com uma grande coleção de funções matemáticas de alto nível para operar nessas matrizes. Assim como *Pandas*, *Numpy* também tem o código aberto e é disponibilizado sob a licença BSD (NUMPY, 2019).

A estrutura de dados que representa um vetor multidimensional disponibilizada pela biblioteca *Numpy*, foi utilizada no desenvolvimento deste trabalho para se dispor os dados antes de submetê-los para o treinamento das RNAs. Além disso, algumas funcionalidades foram utilizadas para redimensionar o vetor bidimensional gerado pelo Pandas para um formato tridimensional, sendo esse um pré-requisito para treinar redes neurais recorrentes.

4.5.1.3 Scikit-Learn

O *Scikit-Learn* é uma biblioteca que integra uma ampla variedade de algoritmos de aprendizado de máquina de ponta para problemas de aprendizado supervisionado e não supervisionado de média escala, além de uma ampla variedade de funcionalidades para pré-processamento de dados. A biblioteca também possuí código aberto e é disponibilizada sob a licença BSD (SCIKIT-LEARN, 2019).

Neste trabalho a biblioteca foi utilizada favoravelmente para realizar o pré-processamento dos dados. Através das funcionalidades disponíveis, realizou-se as etapas de padronização e normalização dos dados e também a etapa de validação cruzada.

4.5.1.4 Matplotlib

O *Matplotlib* é uma biblioteca de plotagem 2D que produz gráficos de qualidade para publicação em diversos formatos de cópia impressa e para ambientes interativos entre plataformas. A biblioteca possuí o código aberto e é disponibilizada sob a licença *Python Software Foundation* (PFS) (MATPLOTLIB, 2019).

Os gráficos gerados por viés do *Matplotlib* no decorrer deste trabalho foram utilizados para analisar visualmente os dados brutos durante a etapa de preparação dos dados. E

também durante o processo de treinamento das redes neurais para acompanhar a evolução do erro e para gerar os gráficos das previsões realizadas pelos melhores modelos.

4.5.1.5 Keras

Segundo CHOLLET (2018), *Keras* é um *framework* de aprendizado profundo para *Python* que fornece uma maneira conveniente de definir e treinar quase qualquer tipo de modelo de aprendizado profundo. O *Keras* foi desenvolvido inicialmente para pesquisadores, com o objetivo de permitir experimentação rápida e possuí as seguintes características:

- Permite que o mesmo código seja executado perfeitamente na CPU ou GPU.
- Possuí uma API amigável que facilita a prototipagem rápida de modelos de aprendizado profundo.
- Possuí suporte interno para redes convolucionais, redes recorrentes e qualquer combinação de ambas.
- Suporta arquiteturas de rede arbitrárias: modelos de múltiplas entradas ou saídas, compartilhamento de camadas, compartilhamento de modelos e assim por diante. Isso significa que o Keras é apropriado para criar essencialmente qualquer modelo de aprendizado profundo, de uma rede neural GAN (*Generative Adversarial Network*) a uma Máquina de Turing Neural (*Neural Turing Machine* – NTM).

Esse *framework* é capaz de rodar sobre outras APIs de aprendizado profundo, tais como *Tensorflow*, *CNTK*. Ou *Theano*. Além disso, possuí código aberto é licenciado sob a licença MIT (KERAS, 2019).

Neste trabalho *Keras* foi utilizado para modelar, treinar e testar as RNAs propostas e por isso a nomenclatura adotada para os recursos utilizados é a mesma utilizada por este software. Através deste *framework* foi possível contar uma grande gama de algoritmos de treinamento, otimizadores, funções de ativação, diferentes modelos de RNAs, funções de erro/custo esquemas de parada antecipada e uma série de outras funcionalidades usadas que posteriormente serão relatadas.

4.5.2 Google Colaboratory

O *Google Colaboratory* é um ambiente na nuvem que disponibiliza *Jupyter Notebooks* como ambiente de desenvolvimento e que não requer nenhum tipo de configuração. Através

desta plataforma é possível executar códigos em *Python*, salvar e compartilhar análises além de acessar poderosos recursos de computação científica, gratuitamente direto do navegador (GOOGLE COLAB, 2019).

O Google ainda oferece acesso à GPUs Telsa K80 da Nvidia e TPUs para acelerar a execução dos modelos de aprendizado de máquina implantados neste ambiente. No entanto, devido à gratuidade de utilização do ambiente e seus recursos, o uso dos aceleradores é limitado, e nem sempre estão disponíveis, porém quando não disponíveis ainda assim é possível executar modelos sem os aceleradores.

O uso do ambiente em si também possuí um limite de 12 horas de utilização, sendo que após esse período a sessão estabelecida com o usuário é terminada abruptamente e os dados da execução de qualquer modelo não finalizada são perdidos. Porém não há limite de espera para que o usuário possa recomeçar uma nova sessão, após terminada uma sessão atual devido ao limite de uso, o usuário pode iniciar uma nova sessão sem nenhuma restrição.

Esse ambiente foi fundamental para o desenvolvimento deste trabalho, pois todos os modelos foram desenvolvidos, treinados e testados nesta plataforma. Por tratar-se de modelos dispendiosos os aceleradores foram extremamente uteis para treinar os modelos com maior velocidade, além disso a cada etapa quatro modelos eram treinados por vez em diferentes notebooks sem que houvesse perda de desempenho.

4.5.3 Funções de ativação

De acordo com Braga, Carvalho e Ludermir (2011), ao se definir a estrutura de uma RNA de múltiplas camadas, deve-se determinar as funções de ativação usadas pelos neurônios de cada camada. Nas camadas ocultas é necessário utilizar funções de ativação não lineares para que a composição das funções nas camadas seguintes tenha a habilidade de resolver problemas de maior ordem no espaço de entrada.

A utilização de funções lineares nas camadas intermediárias resultaria numa RNA linear, ou de única camada, já que funções lineares sucessivas podem ser descritas como uma única transformação linear. Neste caso, Bishop (2006), Braga, Carvalho e Ludermir (2011) e CHOLLET (2018) explicam que é comum utilizar a função linear na camada de saída em problemas de regressão.

Neste sentido, todas as RNAs modeladas no decorrer desta pesquisa utilizam a função linear na camada de saída, que possuí um único neurônio nesta camada, uma vez que o

objetivo é obter um valor real previsto pela rede, i.e o nível do rio de Rio do Sul.

As funções de ativação utilizadas nas camadas intermediárias dos modelos foram variadas empiricamente entre as funções *Tanh*, *ReLU* e *LeakyReLU* com variando entre os valores 0,1; 0,2 e 0,3. Sendo que a variação entre as funções foi realizada em relação aos modelos e não as camadas ocultas dos modelos, ou seja, ao se escolher uma função de ativação para se utilizar em um modelo, todas as camadas intermediárias utilizavam a mesma função de ativação.

Salienta-se ainda que as funções *Tanh* e *Sigmoid* sempre estão presentes nas camadas intermediárias dos modelos, pois estas funções fazem parte da estrutura interna dos modelos neurais recorrentes LSTM e GRU. Dessa forma a função de ativação variada nos modelos é utilizado na saída das células das RNNs, já as funções internas não são alteradas, pois mudariam a estrutura na qual os modelos utilizados foram originalmente projetados.

4.5.4 Algoritmos de treinamento - otimização

Conforme descrito na seção 2.2.4.3, o algoritmo *Backpropagation* ajusta os pesos em função do erro médio quadrático, porém sua utilização tende a convergir muito lentamente, exigindo assim um esforço computacional dispendioso. Para contornar esse problema, Silva, Spatti e Flauzino (2010), explicam que existem diversas técnicas de otimização que têm sido incorporadas ao algoritmo *Backpropagation* com o intuito de tornar o processo de convergência de ajuste dos pesos mais eficiente.

Dentre as variações disponíveis, o método adotado para este trabalho foi o *Adaptive Moment Estimation* – ADAM. Esse algoritmo é uma extensão da descida estocástica do gradiente que recentemente teve uma adoção mais ampla em aplicações de aprendizado profundo, principalmente em visão computacional e processamento de linguagem natural (BROWNLEE, 2017).

4.5.4.1 Adaptive Moment Estimation – ADAM

De acordo com Kingma e Ba (2014), ADAM é um método para otimização estocástica eficiente que requer apenas gradientes de primeira ordem com pouco requisito de memória. O método calcula as taxas de aprendizado adaptativo individual para diferentes parâmetros das estimativas do primeiro e do segundo momento dos gradientes.

Esse algoritmo é projetado para combinar as vantagens de outros dois algoritmos de otimização, também extensões do algoritmo de descida do gradiente, especificamente de acordo

com Brownlee (2017):

- ***Adaptative Gradient Algorithm*** **(AdaGrad)** – Mantém uma taxa de aprendizado por parâmetro para melhorar o desempenho em problemas em que as matrizes dos gradientes são esparsas.
- ***Root Mean Square Propagation*** **(RMSProp)** – Também mantém taxas de aprendizado por parâmetro que são adaptadas com base na média das magnitudes recentes dos gradientes para o peso (por exemplo, a velocidade em que a média muda). Neste sentido, esse algoritmo se sai bem em problemas em que os dados contém muito ruído.

Em vez de adaptar a taxa de aprendizado baseada apenas na média do primeiro momento, como acontece no RMSProp; Adam também usa a média do segundo momento dos gradientes. Além disso, o algoritmo calcula a média móvel exponencial do gradiente e do quadrado do gradiente, e introduz dois parâmetros denominados de *beta1* e *beta2*, que são usados para controlar o decaimento das médias móveis (BROWNLEE, 2017).

Segundo Goodfellow, Bengio e Courville (2016) , ADAM inclui um parâmetro de correção (bias) que é utilizado para corrigir a inicialização das médias móveis e dos termos *beta1* e *beta2* para eliminar tendências ao valor zero, devido a inicialização dos parâmetros serem centrados na origem.

De acordo com Brownlee (2017), os parâmetros de configuração do algoritmo ADAM são:

- ***alpha*** – Também referido como taxa de aprendizado (*learning rate*), diz respeito a proporção em que os pesos são atualizados, por exemplo 0.001. Valores maiores tais como 0.3, resultam em um aprendizado inicial mais rápido antes da atualização da taxa. Valores menores como $1.0E-5$, diminuem a velocidade do aprendizado durante o treinamento.
- ***beta1*** – Taxa de decaimento exponencial para estimava do primeiro momento
- ***beta2*** – A taxa de decaimento exponencial para estimativa do segundo momento. Esse valor deve ser definido próximo de 1 para problemas com gradientes esparsos.
- ***Epsilon*** – Um número pequeno usado para prevenir divisão por zero.

Os parâmetros do algoritmo utilizados no treinamento das RNAs propostas nesta pesquisa não foram alterados durante os experimentos. Os valores usados são os valores pa-

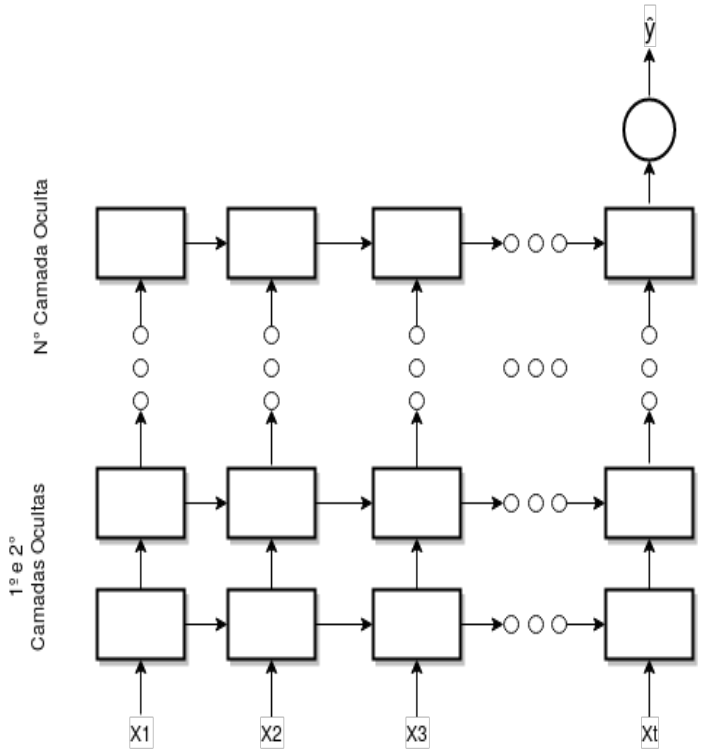
drão configurados pelo Keras, sendo estes os mesmos definidos no artigo original do algoritmo (KINGMA; BA, 2014): $alpha = 0.01$; $beta1 = 0.9$; $beta2 = 0.999$ e $epsilon = 1E - 08$.

4.6 MODELOS DE RNAS

As topologias modeladas neste trabalho possuem k sequências temporais com 4 atributos de entrada e 1 de saída cada, como entrada da rede; n camadas ocultas com m neurônios e um neurônio na camada de saída. A opção de utilizar mais de uma camada oculta se deu com base na pergunta de pesquisa inicialmente proposta, que consiste em verificar se os modelos profundos oferecem um poder representacional superior aos modelos rasos.

A estrutura de ambos os modelos recorrentes LSTM e GRU são semelhantes, inclusive os parâmetros alterados durante o treinamento tais como, quantidades de células e de camadas, número de subamostras utilizadas por atualização dos pesos (*batch*) e a mesma quantidade de sequências temporais. Sendo a única diferença entre os modelos a estrutura interna da célula que compõe sua configuração. Desta forma a Figura 4.6 exibe uma ilustração que resume como ambos os modelos foram configurados:

Figura 24 – Estrutura dos modelos recorrentes LSTM e GRU com n camadas e m células.



Fonte – Elaborado pelo autor.

Cada entrada X na diz respeito a uma observação do conjunto de dados, onde cada

observação possuí os dados de 4 atributos de entrada (Tabela 8) e o conjunto de observações por *batch* são as sequências temporais. O mesmo esquema de entrada é utilizado nas RNAs híbridas.

As RNAs híbridas desenvolvidas neste trabalho possuem n camadas convolucionais com 128 filtros cada e n camadas *Max Pooling* ou *Average Pooling* com *pool size* 1x2; n camadas recorrentes com m células; e n camadas de redes neurais *Multilayer Perceptron* com m neurônios nas camadas ocultas e um neurônio na camada de saída.

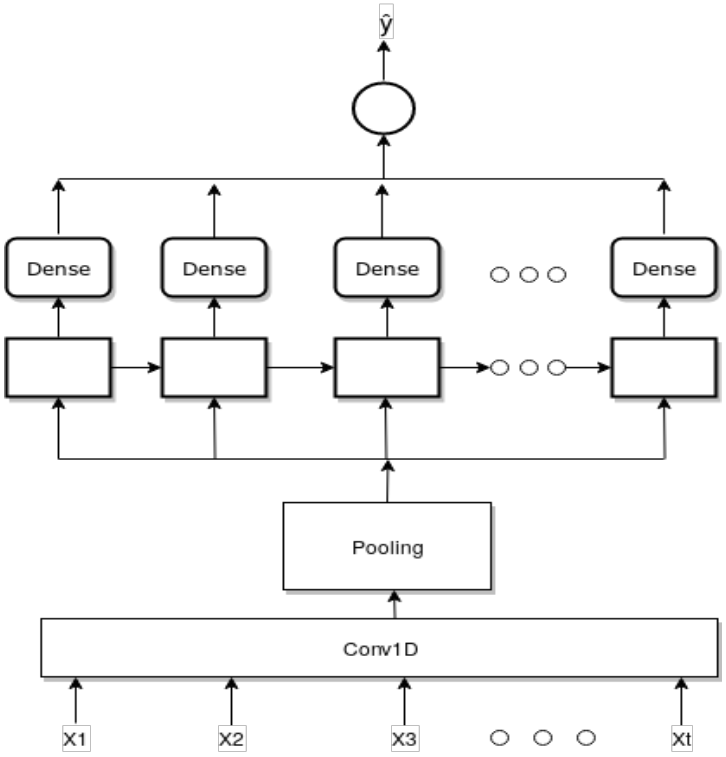
As RNAs convolucionais representam as primeiras camadas das RNAs híbridas. O objetivo de usar esse tipo de camada é extrair informações locais importantes dos dados, tais como padrões temporais, permitindo assim destilar os elementos mais salientes e descartar o restante. Além disso a dimensionalidade da rede é reduzida e o processo de treinamento se torna mais rápido, permitindo assim que um número maior de camadas sejam usadas.

Nas camadas centrais encontram-se as RNNs, que são utilizadas com o objetivo de capturar dependencias temporais dos dados. Nos modelos usados neste trabalho, a RNA híbrida possuí uma das duas vertentes LSTM ou GRU como camadas intermediárias, porém não ambas. Desta forma dois modelos híbridos foram elaborados: um usando células LSTM nas camadas intermediárias e o outro usando-se células GRU.

Os dados que não possuem dependências temporais, porém possuem padrões que lhes atribuem alguma relação dos atributos de entrada com o de saída, são capturados pela rede MLP situada nas camadas finais da rede. De maneira análoga aos modelos recorrentes, ambos os modelos híbridos podem ser ilustrados pela mesma figura (Figura 25), ou seja, apenas a estrutura interna das células recorrentes que mudam.

Por motivos de simplicidade, a RNA Híbrida da Figura 25 possuí apenas uma camada de cada topologia, porém como já mencionado a quantidade de camadas foi variada durante os experimentos realizados. Além disso a camada com neurônios denominados *Dense*, que é uma nomenclatura utilizada pelo *Keras* para denominar neurônios de redes MLP, também por motivos de econômia tais neurônios não estão ligados uns aos outros como é comum nas ilustrações de redes *feedforward* de múltiplas camadas. A nomencaltura *Conv1D* também é uma nomencaltura utilizada pelo software e diz respeito a camada convolucional de 1 dimensão.

Figura 25 – Estrutura dos modelos Híbridos usados nesta pesquisa.



Fonte – Elaborado pelo autor.

Uma rede com mais camadas ou neurônios pode realizar melhores previsões dos dados de treinamento, porque representa com precisão todos os recursos específicos dos dados. Mas essas propriedades específicas podem parecer bem diferentes em dados que não foram usados durante o treinamento do modelo (MEHLIG, 2019).

Neste sentido, uma rede com muitos parâmetros pode levar uma situação conhecida como *overfitting* (memorização excessiva). Inversamente ao se diminuir a complexidade da rede pode surgir o problema conhecido como *underfitting* onde o modelo não tem a capacidade de aprendizado. Para resolver esta situação diversas estrátégias de regularização foram desenvolvidas, dentre elas, tem-se a regularização *Dropout*, que foi adotada pelos modelos desenvolvidos neste trabalho.

De acordo com Mehlig (2019), a regularização *Dropout* consiste em ignorar neurônios das camadas ocultas durante o treinamento. Para isso, a cada etapa do treinamento da rede, uma fração p de neurônios da camada escondida a qual o *Dropout* é aplicado, é selecionada aleatóriamente e os valores gerados pelas ativações dos neurônios são alterados para 0. Desta forma os pesos associados aos neurônios desativados não são atualizados, consequentemente nem seus valores de saída. Uma vez que a fase de treinamento é completada, todos os neurônios

escondidos são ativados, e suas saídas são multiplicadas por p.

A aplicação deste tipo de regularização é utilizada como uma camada extra na rede neural, situada logo após a camada a qual se deseja ter neurônios desativados. Desta forma os neurônios desativados em uma camada não tem efeito na camada seguinte, consequentemente a complexidade da rede diminui e o processo de treinamento se torna mais robusto e os problemas latentes de memorização ou incapacidade de aprendizado podem ser superados.

A escolha das topologias modeladas para realizar este trabalho, foi tomada com base na revisão bibliográfica realizada no decorrer da pesquisa. No entanto, salienta-se que a escolha das topologias CNN e MLP utilizadas na configuração das RNAs híbridas foi realizada de forma empírica. Para realizar a avaliação dos resultados produzidos pelas redes, métricas de avaliação foram utilizadas e são descritas na seção seguinte.

4.7 ÍNDICES PARA ANÁLISE DE QUALIDADE DA PREVISÃO

Os índices de avaliação da qualidade da previsão adotados nesta pesquisa foram realizados com base na revisão bibliográfica e são utilizados com o objetivo de julgar a qualidade de previsão dos modelos empregados.

Os índices utilizados foram o RMSE (*Root Mean Square Error* – Raíz quadrada do erro médio); o MAPE (*Mean Average Percentage Error* – Percentual do Erro médio); e o NSE (*Nash–Sutcliffe Model Efficiency Coefficient* – Coeficiente de Eficiência de Nash e Sutcliffe). As equações para o cálculo dos índices são apresentadas a seguir, juntamente com uma breve descrição do seu significado.

- ***Root Mean Square Error* – RMSE**

De acordo com Stephanie (2016), RMSE representa o desvio padrão dos resíduos (erros de previsão). Os resíduos são uma medida da distância dos pontos de dados da linha de regressão – dada pela diferença entre os valores reais e os previstos; A métrica produz uma medida de quão espalhados esses resíduos estão. Resumidamente indica a concentração dos dados na linha de melhor ajuste;

De acordo com Otto (2019), o RMSE pode ser calculado através da equação 4.5:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y})^2}{n}} \tag{4.5}$$

Onde y_i é a iéssima observação de y e $\hat{y}$ é o valor previsto pelo modelo. Se os valores previstos estiverem muito próximos dos valores verdadeiros, o RMSE será pequeno. Porém, se as respostas previstas diferirem das verdadeiras substancialmente o RMSE será grande. Neste sentido um valor zero indica um ajuste perfeito para os dados.

- ***Nash–Sutcliffe Model Efficiency Coefficient* – NSE**

Segundo Nash e Sutcliffe (1970), o índice de eficiencia de Nash-Sutcliffe é uma estatística normalizada que determina a magnitude relativa da variação residual (ruído) em comparação com a variação dos dados medidos.

De acordo com Soares (2014) esse índice é utilizado para avaliar o poder preditivo de modelos hidrológico, sendo o resultado produzido semelhante ao do coneficiente de determinação R^2 utilizado na regressão. Seu valor varia dentro do intervalo $(-\infty, 1]$. Quanto mais próximo de 1 mais preciso é o modelo, onde uma eficiencia de 1 (NSE = 1) corresponde a um ajuste perfeito entre as previsões realizadas pelo modelo e os dados observados.

O Coeficiente de Eficiência de Nash e Sutclife (NSE) é definido de acordo com a equação 4.6:

$$NSE = 1 - \frac{\sum_{i=1}^{n} \left(Qo_i - Qp_i\right)^2}{\sum_{i=1}^{n} \left(Qo_i - \bar{Q}o\right)^2} \tag{4.6}$$

Onde Qo_i é o valor observado, Qp_i o valor previsto e $\bar{Q}o$ a média dos valores observados.

- ***Mean Average Percentage Error* – MAPE**

O MAPE é a média dos erros percentuais absolutos das previsões. O erro é definido como valor real ou observado menos o valor previsto. Os erros de porcentagem são somados sem considerar o sinal . Essa medida é fácil de analisar porque fornece o erro em termos de porcentagens. Além disso, como erros percentuais absolutos são usados, evita-se o problema de erros positivos e negativos cancelando-se mutuamente. Consequentemente, é uma medida comumente usada na previsão. Quanto mais próximo de 0 o MAPE, melhor a previsão (SWAMIDASS, 2000).

Segundo Stephanie (2017), o MAPE é definido pela equação 4.7:

$$\frac{1}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right| \tag{4.7}$$

Onde A_t é o valor real e F_t o valor observado. Resumidamente o desempenho de um modelo de previsão é bom quando os índices das métricas NSE, RMSE e MAPE estão próximos de 1, 0 e 0 respectivamente. Um ponto a se notar é que estes índices são utilizados para avaliar os resultados produzidos pelas RNAs após o treinamento. Durante o treinamento os modelos são avalidados pela métrica MSE que é usada como função de erro pelo algoritmo de treinamento.

4.8 DEFINIÇÃO DA MELHOR CONFIGURAÇÃO DE REDE NEURAL ARTIFICIAL

Os modelos de RNAs possuem múltiplos parâmetros de ajuste, que modificam suas características e consequentemente levam a rede a produzir resultados diversos em cada treinamento realizado. Durante o treinamento dos modelos o número de camadas escondidas e seus neurônios foram alterados e para cada camada escondida foram realizados ensaios com a variação de diversos parâmetros.

Desta forma os parâmetros tais como funções de ativação, taxas da regularização *Dropout*, quantidade de sequências temporais e tamanho do *batch* usado por época de treino, foram variados. Em contrapartida alguns outros parâmetros não foram alterados, como algoritmo de treinamento e seus parâmetros internos e o uma quantidade máxima de 200 épocas que foi mantida constante.

No entanto, cabe salientar que se utilizou do recurso de parada antecipada provida pela *Keras*, ou seja, se a partir de uma determinada quantidade de iterações não houver melhoras no desempenho do erro produzido pelo conjunto de validação, o treinamento é abruptamente finalizado, onde a quantidade determinada neste trabalho são de 10 iterações.

Dentro desse contexto, a escolha da melhor configuração foi realizada em 4 fases:

1. As redes LSTM são treinadas variando-se o número de camadas ocultas entre 1 e 5 camadas no total, onde para cada camada um conjunto de configurações diferentes foram experimentadas. Em seguida, para cada conjunto foram escolhidos os ensaios que possuíam o menor RMSE, menor MAPE e maior NSE; Na sequência, os ensaios escolhidos são comparados entre si, porém desta vez, sem distinção do número de camadas, assim elegendo-se o modelo LSTM que possuí o menor RMSE, menor MAPE e maior NSE. Essa mesma sistemática é adotada para as redes GRU

2. Os melhores modelos LSTM e GRU selecionados durante a primeira fase, são novamente comparados de modo a selecionar um dos dois modelos recorrentes para serem

comparados na fase final com o modelo híbrido escolhido.

3. Dois modelos de RNAs híbridas foram treinadas com configurações diferentes e comparadas entre si com o objetivo de identificar, novamente, a rede com menor RMSE, menor MAPE e maior NSE; Nessa fase, inicialmente, diferentes configurações do modelo que possuí camadas LSTM, são comparadas entre sí para se definir a melhor configuração híbrida com LSTM. Da mesma forma, diferentes configurações das redes híbridas com camadas GRU são comparadas entre sí para se obter a melhor configuração. Por último, o melhor modelo híbrido com LSTM é comparado com o melhor modelo híbrido com GRU de modo a se eleger a melhor rede híbrida.

4. Por último o melhor modelo recorrente eleito é comparado com o melhor híbrido eleito; deste modo o modelo com a configuração mais aderente à área de estudo selecionada é identificado.

5 RESULTADOS

O capítulo de resultados foi subdividido em 8 seções principais: Iniciando-se na seção 5.1, os parâmetros alterados durante a fase de treinamento, bem como os parâmetros mantidos constantes, são aclarados.

A partir da seção 5.2 o treinamento dos modelos recorrentes LSTM e os resultados obtidos por essa topologia são descritos; Na seção 5.3, semelhantemente o treinamento e resultados das redes recorrentes GRU são discutidos e analisados; Ao final das seções 5.2 e 5.3 as melhores configurações das redes LSTM e GRU são respectivamente apresentadas; Por último na seção 5.4 são comparadas as configurações de LSTM e GRU que obtiveram os melhores desempenhos, finalizando com a eleição da melhor configuração de RNA recorrente.

Nas seções 5.5, 5.6 e 5.7, a mesma sistemática é adotada para as redes neurais Híbridas. Inicialmente são descritos o treinamento e resultados da rede híbrida com LSTM nas camadas ocultas e na sequência a rede híbrida com GRU nas camadas ocultas. Na seção 5.7 são comparadas as configurações Híbridas que obtiveram os melhores desempenhos, finalizando com a eleição da melhor configuração de RNA híbrida.

Por último, na 5.8, são comparadas as configurações recorrentes e híbridas que obtiveram os melhores desempenhos, finalizando com a eleição da melhor configuração de RNA profunda, visando escolher a melhor rede para a previsão do rio Itajaí Açu, no local de estudo selecionado, e responder a pergunta de pesquisa inicialmente proposta.

5.1 PARÂMETROS

Ao se realizar o treinamento de uma RNA, existem diversos parâmetros que podem ser alterados em sua configuração. Neste trabalho alguns parâmetros foram alterados e outros foram mantidos constantes.

Em função dos parâmetros alterados, o número de camadas ocultas, células e de neurônios, bem como as funções de ativação, as taxas de inclinação da função *LeakyReLU*, as probabilidades da regularização *Dropout*, Sequências Temporais (*Timesteps*), e conjunto de amostras (*Batch*) usado na atualização dos pesos e limiares por iteração, e as camadas de Pooling foram variados durante a fase de treinamento. Sendo as configurações alteradas as seguintes:

- Número de células e neurônios: 32, 64 e 128.

- Número de camadas nos modelos Recorrentes: de 1 a 5 camadas.
- Número de camadas nos modelos Híbridos: de 4 a 16 camadas.
- Taxas de inclinação α da função *LeakyReLU*: 0.1, 0.2, e 0.3.
- Probabilidades da regularização *Dropout*: 0.01, 0.05 e 0.1;
- Sequências temporais (*timesteps*) e *batch size*: 32, 64 e 128 – equivalente a 8, 16 e 32 horas de observações;
- Funções de Ativação: *Tanh*, *Relu*, *LeakyReLU*;
- Camadas *Pooling*: Variando entre *Max Pooling* e *Average Pooling*

Vários parâmetros foram mantidos constantes, dentre eles: quantidade e comprimento dos filtros utilizados nas camadas convolucionais; método de preenchimento (*padding*); o tamanho da janela (*pool size*) nas camadas de *Pooling*; o algoritmo de treinamento ADAM e seus parâmetros internos; o método de inicialização dos pesos; número máximo de épocas; valor mínimo de parada antecipada; e função de erro. As configurações mantidas constantes são as seguintes:

- *filters*: 128;
- *kernel_size* : 3;
- *padding*: *same* – resulta no preenchimento do vetor de entrada de modo que a saída tenha o mesmo comprimento que a entrada original. O mesmo preenchimento é utilizado em ambas camadas da rede CNN;
- *pool size*: 2;
- Parâmetros do algoritmo de treinamento Adam:
  - *learning_rate*: 0.01;
  - *beta1*: 0.9;
  - *beta2*: 0.999;
  - *epsilon*: 1E-08.
- Método de Inicialização dos pesos: *xavier_uniform*;
- Quantidade máxima de épocas: 200;

• *EarlyStopping* (Parada antecipada): 10 – Se dentro de 10 iterações o modelo não mostrar melhoras em relação a taxa de erro produzida pelo conjunto de validação, o treinamento é abruptamente terminado.

O método de inicialização dos pesos pode ser alterado através do parâmetro *kernel_initializer*, presente nas camadas de qualquer uma das topologias adotadas neste trabalho. O método de inicialização padrão no *Keras* é o *Xavier Uniform*. Esse método inicializa os pesos a partir de uma distribuição probabilística (normal ou uniforme) com base na variância do conjunto de dados. Desta forma a variância é mantida constante nas camadas da rede, em ambas as fases de propagação para frente e de retro propagação. Esse esquema ajuda a evitar que as funções de ativação saturem e inibe a explosão e o desaparecimento dos gradientes (GLOROT; BENGIO, 2010).

A avaliação do desempenho das RNAs durante a fase de treinamento é realizada através do MSE, que deve ser configurado explicitamente no *Keras*. Selecionado como função de erro a ser minimizada pelo *Backpropagation*, deve-se salientar que quanto mais próximo de zero o valor produzido por tal função, melhor será o desempenho da rede.

5.2 TREINAMENTO E RESULTADOS DA RNN LSTM

As subseções seguintes descrevem o processo de treinamento e os resultados obtidos com a utilização das redes neurais recorrentes *Long Short Term Memory* (LSTM).

5.2.1 Treinamento

Um total de 47 configurações de redes LSTM foram treinadas. Desse total, as redes com menos camadas tiveram mais configurações treinadas em relação aos modelos com mais camadas, pois quanto mais camadas se tinha, mais dispendioso era o tempo de treinamento. Por exemplo, enquanto as LSTMs com 1 camada levavam em torno de 1 hora à 1,5 horas para treinar, as redes com 5 camadas levavam em torno de 10 à 12 horas.

De maneira similar ao se aumentar o número de células, também mais tempo demorava o treinamento. Desta forma, as configurações com 5 camadas não foram treinadas com 128 células, pois se fossem, o tempo total para realizar o treinamento excederia o limite disponível no *Google Colaboratory*

O ajuste dos pesos e limiares da rede durante o treinamento foram realizados conforme a função de erro escolhida, que no caso deste trabalho foi a função *Mean Square Root*

(MSE), calculado a partir da saída produzida pela rede e as saídas desejadas. Quanto mais próximo de zero for o MSE durante o treinamento, melhor o desempenho do modelo. Neste sentido, tomando-se como base o MSE produzido pela rede durante tal fase, na Tabela 9 é apresentada a configuração que obteve o melhor desempenho.

Tabela 9 – Melhor desempenho LSTM durante o treinamento.

N° Camadas	**N° Células**	**Função de Ativação**	**Regularização**	**Épocas**	**MSE**
5	64	*LeakyReLU* – 0.03	*Dropout* – 0.1	130	4.4801E-05

Fonte: Elaborado pelo autor.

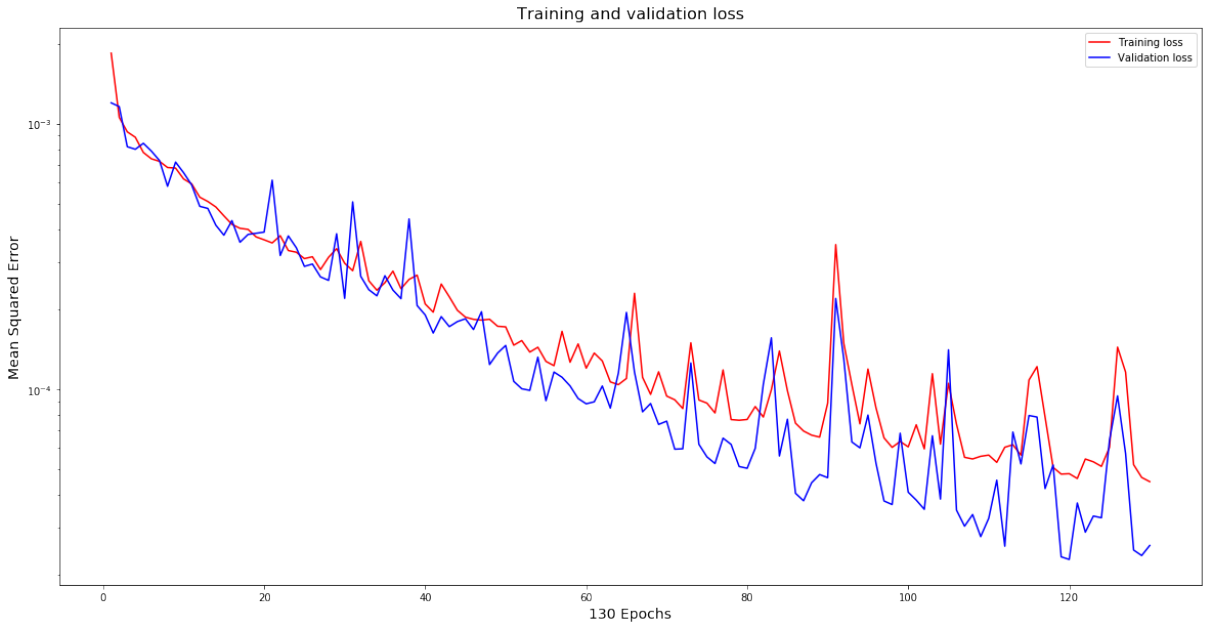
Conforme se pode observar na Tabela 9, a rede LSTM que obteve o menor MSE no conjunto de treinamento, chegou a tal resultado com 5 camadas ocultas, 64 células em cada camada, função de ativação *LeakyReLU* com configurado para 0.03, regularização *Dropout* com probabilidade de 10% e 130 épocas de treinamento. Coincidentemente, essa configuração foi a que obteve o melhor desempenho nas outras métricas durante a fase de testes.

A Figura 1 exibe a evolução do erro durante o treinamento da rede com o melhor desempenho, pode-se observar que o treinamento foi interrompido na época 130, onde o máximo que foi estipulado foi de 200 épocas. Conforme mostra a figura, pode-se perceber que os ruídos se manifestaram mais nas últimas épocas, onde a rede deixou de exibir melhoras significativas levando o treinamento a interrupção. Cabe salientar que o MSE exibido na Tabela 9 diz respeito ao erro calculado sob o conjunto de treino, já o conjunto de validação que foi usado para ajustar os parâmetros da rede durante o treinamento exibiu o menor MSE que foi de 2.2818E-05 na época 120.

O valor do MSE alcançado pelas RNAs durante o treinamento nos conjuntos de treinamento e validação, exibem a rede que obteve o melhor ajuste nesta fase, porém o interesse maior é pelos resultados produzidos durante a fase de testes, que é quando são apresentadas amostras que não foram consideradas no treinamento para o modelo com os parâmetros ajustados com base no conjunto de validação.

As previsões realizadas utilizando as amostras de teste ajudaram a definir a rede LSTM que fez as previsões com menor erro, para posteriormente ser comparada com a GRU que também obteve o menor erro no conjunto de testes considerando-se os dados da área de estudo selecionada. Na subseção a seguir são apresentados os resultados produzidos pela rede LSTM na fase de testes.

Figura 26 – Evolução do erro do modelo LSTM com melhor desempenho durante o treinamento.



Fonte – Acervo do autor.

5.2.2 Resultados

Os resultados produzidos pelas configurações LSTM quando apresentadas ao conjunto de testes são apresentados nas tabelas a seguir. A configuração dos modelos é identificada na primeira coluna da tabela, com o nome do modelo e através da seguinte definição de siglas:

- (i)C: RNN com i camadas ocultas;
- (n)CE: RNN com n células nas camadas ocultas;
- A: Função de ativação usada na saída das células das camadas ocultas
- (m)TS: Vetor de entrada com m sequências temporais (*timesteps*);
- (k)B: Vetor de entrada com k amostras por *batch*.
- Dprt: RNN com *Dropout*.

Por exemplo, o modelo com a configuração *LSTM _2C _64CE _A _leakyrely_0.2 _32TS _64B _Drpt _0.05* tem o seguinte significado: Modelo LSTM com 2 camadas ocultas, 64 células nas camadas ocultas, função de ativação *LeakyReLU* com inclinação α 0.2, 64 sequências temporais, 64 amostras por *batch* e *Dropout* de 5%.

Tabela 10 – Resultados de cada configuração da rede recorrente LSTM com 1 camada oculta.

LSTM – 1 CAMADA				
Modelo	Epochs	RMSE	MAPE %	NSE
LSTM_1C_32CE_A_tanh_32TS_ 32B	94	4.699E-02	12.421	8.004E-01
LSTM_1C_32CE_A_relu_32TS_ 32B	100	4.863E-02	13.412	7.862E-01
LSTM_1C_32CE_A_leakyrelu_ 0.1_32TS_32B	55	4.689E-02	12.034	8.012E-01
LSTM_1C_32CE_A_leakyrelu_ 0.2_32TS_32B	43	5.097E-02	13.161	7.652E-01
LSTM_1C_32CE_A_leakyrelu_ 0.3_32TS_32B	76	5.028E-02	14.753	7.715E-01
LSTM_1C_64CE_A_tanh_32TS_ 64B	100	4.931E-02	13.101	7.802E-01
LSTM_1C_32CE_A_tanh_64TS_ 32B	100	1.785E-02	**6.378**	9.762E-01
LSTM_1C_64CE_A_tanh_64TS_ 64B	24	2.546E-02	7.672	9.515E-01
LSTM_1C_32CE_A_tanh_128TS_ 128B	45	2.333E-02	8.248	9.552E-01
LSTM_1C_64CE_A_tanh_128TS_ 128B	59	2.319E-02	7.116	9.593E-01
LSTM_1C_64CE_A_leakyrelu_0.3_128TS _128B	28	**1.705E-02**	6.508	**9.779E-01**
LSTM_1C_128CE_A_tanh_128TS_ 128B	88	2.887E-02	9.583	9.356E-01
LSTM_1C_128CE_A_leakyrelu_0.3_ 128TS_128B_Drpt_0.1	37	2.076E-02	6.932	9.659E-01

Fonte: Elaborado pelo autor.

O menor MAPE (**6.378%**) foi obtido pela configuração treinada 32 células na camada oculta, função de ativação *Tanh*, 64 sequências temporais e 32 amostras por *batch*. No entanto, nos outros dois índices uma única configuração exibiu o menor RMSE (**1.705E-02**) e o maior NSE (**9.779E-01**). Tal modelo possuí 64 células na camada oculta, função de ativação *LeakyReLU* com α 0.3, 128 sequências temporais e 128 amostras por *batch*.

As LSTMs de uma camada oculta com os piores NSE, RMSE, e MAPE, ou seja, os modelos que exibiram os resultados com maior erro, foram os modelos menos complexos (com menos parâmetros), onde o modelo *LSTM_1C_32CE_A_leakyrelu_0.2_32TS_32B* exibiu o pior RMSE (**5.097E-02**) e o pior NSE (**7.715E-01**); já o pior MAPE (**14.753**) foi exibido pelo modelo *LSTM_1C_32CE_A_leakyrelu_0.3_32TS_32B*.

Ao se comparar os modelos que exibiram os melhores resultados com os piores resultados, é possível observar que aumentado a capacidade de rede com mais células, ou com um conjunto maior de sequências temporais e/ou maior quantidade de amostras por *batch*, melhores eram os resultados. Essas observações foram valiosas ao se treinar as redes com mais camadas, uma vez que essas levam mais tempo, os parâmetros que melhor se adaptaram nas redes com menos camadas tiveram uma maior preferência nos testes realizados nas redes com mais camadas.

Seguindo-se a sistemática adotada nesta pesquisa, apenas um modelo de cada conjunto de camadas deve ser eleito o melhor, para que ao final seja comparado com os melhores modelos de mais camadas. Dessa forma considerando-se os modelos LSTM de 1 camada que obtiveram os melhores desempenhos, optou-se por eleger o modelo *LSTM_1C_64CE_A_leakyrelu _0.3_128TS_128B* como o melhor, pois este exibiu os melhores resultados em duas das três métricas utilizadas (RMSE e NSE), enquanto que o outro modelo apresentou o melhor resultado em apenas uma métrica (MAPE).

Na sequência a Tabela 11 exibe os resultados obtidos pelos modelos com 2 camadas ocultas.

Tabela 11 – Resultados de cada configuração da rede recorrente LSTM com 2 camadas ocultas.

LSTM – 2 CAMADAS				
Modelo	Epochs	RMSE	MAPE %	NSE
LSTM_2C_32CE_A_leakyrelu_0.1_32TS_32B	92	1.809E-02	6.157	9.745E-01
LSTM_2C_64CE_A_tanh_32TS_64B_Drpt_0.05	71	4.699E-02	12.281	8.004E-01
LSTM_2C_64CE_A_leakyrelu_0.1_64TS_64B	100	4.564E-02	12.253	8.096E-01
LSTM_2C_64CE_A_leakyrelu_0.3_128TS_128B	136	**6.783E-03**	**3.055**	**9.965E-01**
LSTM_2C_64CE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	80	1.950E-02	8.301	9.701E-01
LSTM_2C_64CE_A_tanh_128TS_128B_Drpt_0.1	108	1.715E-02	5.981	9.771E-01
LSTM_2C_128CE_A_tanh_128TS_128B_Drpt_0.01	140	1.365E-02	5.203	9.853E-01
LSTM_2C_128CE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	89	1.110E-02	4.922	9.903E-01

Fonte: Elaborado pelo autor.

Conforme mostra a Tabela 11, o modelo LSTM com 2 camadas ocultas, 64 célu-

las em cada camada, função de ativação *LeakyReLU* com 0.3, 128 sequências temporais e 128 amostras por *batch*, coincidentemente foi o modelo com melhor desempenho em todas as métricas dentre os outros modelos de duas camadas. Tal modelo obteve o menor RMSE (**6.783E-03**); menor MAPE (**3.055%**); e o maior NSE (**9.965E-01**). Desta forma sendo eleito unanimemente como melhor modelo de duas camadas ocultas.

Similarmente, apenas um modelo apresentou o pior desempenho em todas as métricas em relação aos outros modelos. A configuração que obteve o maior RMSE (**4.699E-02**); maior MAPE (**12.281%**); e o menor NSE (**8.004E-01**); chegou a esse resultado com 64 células nas camadas ocultas, função de ativação *Tanh*, 32 sequências temporais, 64 amostras por *batch* e *Dropout* com 5% de probabilidade.

Através dos resultados obtidos, é possível observar que ambos os piores e melhores resultados dos modelos de 2 camadas foram superiores aos melhores e piores resultados dos modelos de 1 camada, ou seja, o melhor resultado do modelo raso foi inferior ao melhor resultado do modelo com mais profundidade em todas as métricas. Semelhantemente, os piores índices do modelo de 1 camada, foram piores que os índices do modelo de 2 camadas.

Seguindo-se a mesma metodologia, a Tabela 12 exibe o desempenho dos modelos com 3 camadas ocultas nas métricas adotadas neste trabalho.

Tabela 12 – Resultados de cada configuração da rede recorrente LSTM com 3 camadas ocultas.

LSTM – 3 CAMADAS				
Modelo	Epochs	RMSE	MAPE %	NSE
LSTM_3C_64CE_A_leakyrely_0.1_64TS_64B_Drpt_0.1	193	6.175E-03	2.362	9.971E-01
LSTM_3C_64CE_A_leakyrely_0.3_128TS_128B_Drpt_0.01	113	6.482E-03	2.814	9.968E-01
LSTM_3C_128CE_A_tanh_128TS_128B_Drpt_0.01	27	2.136E-02	7.873	9.651E-01
LSTM_3C_64CE_A_leakyrely_0.2_128TS_128B_Drpt_0.01	153	**5.453E-03**	**2.026**	**9.977E-01**
LSTM_3C_64CE_A_relu_128TS_128B_Drpt_0.01	107	9.001E-03	3.317	9.939E-01
LSTM_3C_32CE_A_leakyrely_0.2_64TS_64B_Drpt_0.01	108	1.483E-02	5.202	9.829E-01
LSTM_3C_32CE_A_leakyrely_0.3_128TS_128B_Drpt_0.01	94	1.944E-02	6.599	9.701E-01

Fonte: Elaborado pelo autor.

De maneira similar a configuração com 2 camadas, conforme é destacado na Tabela

12, uma única configuração com 3 camadas ocultas superou todas as outras configurações de 3 camadas nas três métricas utilizadas, ou seja, o melhor modelo obteve o menor RMSE (**5.453E-03**), menor MAPE (**2.026%**) e o maior NSE (**9.977E-01**).

O modelo em questão chegou a tal resultado com 64 células nas camadas ocultas, função de ativação *LeakyReLU* com α 0.2, 128 sequências temporais, 128 amostras por *batch* e 1% de *Dropout*.

Dentre os modelos de 3 camadas, apenas uma configuração obteve o pior desempenho em todas as métricas, tendo sido o treinamento interrompido na época 22, exibiu o maior RMSE (**2.136E-02**), maior MAPE (**7.873%**) e o menor NSE (**9.651E-01**). Chegando a esse resultado com 128 células nas camadas ocultas, função de ativação *Tanh*, 128 sequências temporais, 128 amostras por *batch* e 1% de*Dropout*.

O modelo com o pior desempenho deveria apresentar resultados melhores, pois em tese, por possuí uma quantidade maior de células em relação a configuração com o melhor desempenho, maior seria seu poder representacional. Através dos resultados obtidos pode-se observar que todas as configurações que obtiveram os piores desempenhos nos índices, faziam o uso da função *Tanh*. Esses resultados refletem o intervalo em que os dados foram normalizados, uma vez que este se difere do intervalo da função *Tanh*.

A Tabela 13 a seguir exibe os resultados obtidos pelas configurações com 4 camadas ocultas.

Tabela 13 – Resultados de cada configuração da rede recorrente LSTM com 4 camadas ocultas.

LSTM – 4 CAMADAS				
Modelo	Epochs	RMSE	MAPE %	NSE
LSTM_4C_32CE_A_relu_128TS_128B_Drpt_0.01	93	6.531E-03	2.659	9.968E-01
LSTM_4C_32CE_A_leakyrely_0.1_128TS_128B	56	1.944E-02	7.862	9.719E-01
LSTM_4C_64CE_A_leakyrely_0.1_64TS_64B_Drpt_0.1	93	1.034E-02	3.617	9.919E-01
LSTM_4C_64CE_A_leakyrely_0.3_64TS_64B_Drpt_0.1	100	1.089E-02	4.864	9.908E-01
LSTM_4C_64CE_A_leakyrely_0.3_128TS_128B_Drpt_0.1	102	8.648E-03	**2.576**	9.945E-01
LSTM_4C_128CE_A_tanh_128TS_128B_Drpt_0.01	47	2.369E-02	7.653	9.583E-01
LSTM_4C_64CE_A_relu_128TS_128B_Drpt_0.01	103	**5.996E-03**	2.644	**9.973E-01**

Fonte: Elaborado pelo autor.

Nas configurações com 4 camadas ocultas, o menor MAPE (**2.576%**) foi obtido pelo modelo com 64 células, função de ativação *LeakyReLU* com α 0.3, 128 sequências temporais e 128 amostras por *batch* e 10% de *Dropout*. Em contrapartida, de maneira similar as configurações com 1 camada oculta, um único modelo com 4 camadas sucedeu os outros modelos nas outras duas métricas.

O modelo em questão obteve o menor MSE (**5.996E-03**) e o maior NSE (**9.973E-01**), chegando a este resultado com 64 células nas camadas ocultas, função de ativação *ReLU*, 128 sequências temporais, 128 amostras por *batch* e 1% de *Dropout*.

Em relação aos piores resultados, de forma similar as redes com o melhor desempenho, o modelo *LSTM_4C_32CE_A_leakyrely_0.1_128TS_128B* exibiu o pior MAPE (**7.862E%**) e o modelo *LSTM_4C_128CE_A_tanh_128TS_128B_Drpt_0.01* os piores RMSE (**2.369E-02**) e NSE (**9.583E-01**).

Seguindo-se a mesma estratégia, apenas um modelo deve ser selecionado para a comparação final que visa escolher o modelo LSTM mais adequado para realizar as previsões. Desta forma, o modelo *LSTM_4C_64CE_A_relu_128TS_128B_Drpt_0.01* é eleito o melhor modelo dentre os modelos com 4 camadas ocultas, pois foi superior a todos os outros em, pelo menos, duas das métricas utilizadas.

Na sequência, a Tabela 14 exibe os resultados obtidos pelas configurações com 5 camadas ocultas.

Tabela 14 – Resultados de cada configuração da rede recorrente LSTM com 5 camadas ocultas.

LSTM – 5 CAMADAS				
Modelo	Epochs	RMSE	MAPE %	NSE
LSTM_5C_32CE_A_leakyrelu_0.1_128TS_128B_Drpt_0.1	88	9.672E-03	3.995	9.930E-01
LSTM_5C_64CE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	130	**5.040E-03**	**2.032**	**9.980E-01**
LSTM_5C_64CE_A_tanh_128TS_128B_Drpt_0.01	57	1.847E-02	6.429	9.735E-01
LSTM_5C_64CE_A_leakyrelu_0.2_128TS_128B_Drpt_0.1	78	7.775E-03	3.254	9.952E-01
LSTM_5C_64CE_A_relu_128TS_128B_Drpt_0.1	67	9.798E-03	4.451	9.927E-01
LSTM_5C_32CE_A_relu_128TS_128B_Drpt_0.1	58	1.688E-02	5.488	9.776E-01
LSTM_5C_64CE_A_leakyrelu_0.1_128TS_128B_Drpt_0.1	110	6.319E-03	2.642	9.970E-01

Fonte: Elaborado pelo autor.

De maneira similar aos modelos com 2 e 3 camadas ocultas, uma única configuração com 5 camadas ocultas obteve os melhores desempenhos nas três métricas utilizadas, em relação as outras configurações de mesma profundidade. O modelo em questão obteve o menor RMSE (**5.040E-03**), menor MAPE (**2.032%**) e o maior NSE (**9.980E-01**).

Os melhores resultados foram obtidos pelo modelo configurado com 64 células nas camadas ocultas, função de ativação *LeakyReLU* com α 0.3, 128 sequências temporais, 128 amostras por *batch* e 10% de *Dropout*.

Em relação aos índices que apresentaram os maiores erros, uma única configuração obteve o maior RMSE (**1.847E-02**), maior MAPE (**6.429%**) e menor NSE (**9.735E-01**); chegando a esses resultados com 64 células nas camadas ocultas, função de ativação *Tanh*, 128 sequências temporais, 128 amostras por *batch* e 1% de *Dropout*.

Uma vez que as melhores configurações para cada grupo de camadas foram selecionadas, as tais são dispostas na Tabela 15, os resultados são comparados a fim de eleger a melhor rede LSTM.

Tabela 15 – Melhores desempenhos obtidos por grupos de LSTMs com *n* camadas ocultas.

LSTM – Melhores Resultados				
Modelo	Epochs	RMSE	MAPE %	NSE
LSTM_1C_64CE_A_leakyrelu_0.3 _128TS_128B	28	1.705E-02	6.508	9.779E-01
LSTM_2C_64CE_A_leakyrelu_0.3 _128TS_128B	136	6.783E-03	3.055	9.965E-01
LSTM_3C_64CE_A_leakyrely_0.2 _128TS_128B_Drpt_0.01	153	5.453E-03	**2.026**	9.977E-01
LSTM_4C_64CE_A_relu_128TS_1 28B_Drpt_0.01	103	5.996E-03	2.644	9.973E-01
LSTM_5C_64CE_A_leakyrelu_0.3 _128TS_128B_Drpt_0.1	130	**5.040E-03**	2.032	**9.980E-01**

Fonte: Elaborado pelo autor.

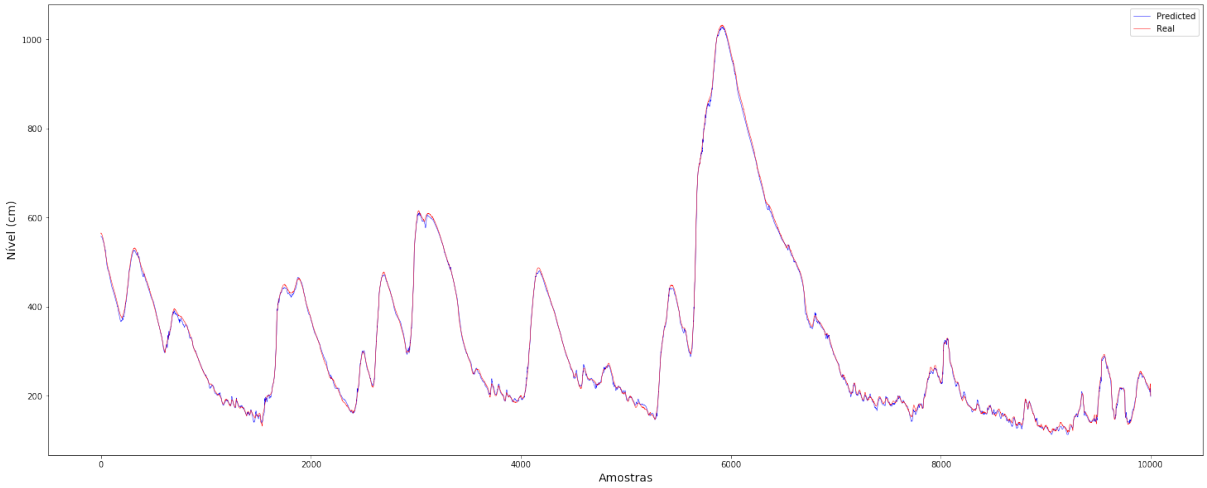
O menor MAPE (**2.026%**) foi obtido pela configuração treinada com 3 camadas ocultas, 64 células em cada camada, função de ativação *LeakyReLU* com α 0.2, 128 sequências temporais, 128 amostras por *batch* e 1% de *Dropout*. No entanto, nos outros dois índices, uma única configuração exibiu o menor RMSE (**5.040E-03**) e o maior NSE (**9.980E-01**); chegando a tal resultado com 5 camadas ocultas, 64 células em cada camada, função de ativação *LeakyReLU* com α 0.3, 128 sequências temporais, 128 amostras por batch e 1% de Dropout.

Em contrapartida o maior RMSE (**1.705E-02**), maior MAPE (**6.508E%**) e menor NSE (**9.779E-01**), foram obtidos pelo modelo menos complexo, ou seja, o modelo com apenas 1 camada oculta. Coincidentemente o modelo possuí a mesma configuração que o modelo com 5 camadas, apenas diferindo-se em função da profundidade.

Embora a configuração com 5 camadas ocultas tenha obtido um desempenho melhor em duas das três métricas utilizadas, o modelo com 3 camadas ocultas, embora tenha sido melhor em uma única métrica, porém apresenta um bom desempenho nas outras métricas, que exibem um baixo erro de previsão.

Desta forma, antes de selecionar a melhor rede LSTM, os gráficos com os resultados dos valores do nível do rio calculado e observado para as duas redes que obtiveram os melhores RMSE, MAPE e NSE, são apresentados através da Figura 27 e Figura 28.

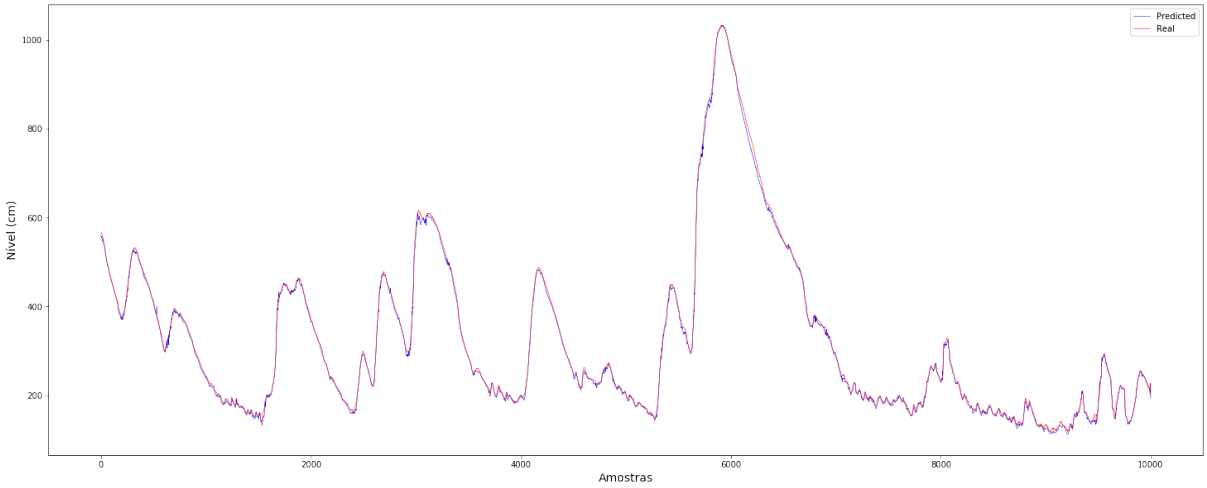
Figura 27 – Gráfico do nível calculado e observado do modelo com 3 camadas ocultas *LSTM_3C_64CE_A_leakyrely_0.2_128TS_128B_Drpt_0.01*.



Fonte – Acervo do autor.

Conforme pode ser observado na Figura 27, os níveis calculados pela LSTM com 3 camadas ocultas foram muito próximos dos observados. As maiores diferenças ocorreram para os níveis entre 4 e 6 metros, onde o modelo calculou valores menores do que os observados. No entanto, o maior interesse desta pesquisa são as previsões do nível do rio para valores acima de 6,5 metros, que para a região de estudo selecionada são considerados altos. Para essa faixa, os níveis calculados quase não apresentam diferença em relação aos níveis observados.

Figura 28 – Gráfico do nível calculado e observado do modelo com 5 camadas ocultas *LSTM_5C_64CE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1*.



Fonte – Acervo do autor.

Assim como o modelo de 3 camadas, o modelo com 5 camadas produziu resultados muito parecidos com os níveis observados, conforme pode-se observar na Figura 28. A imprecisão também ocorreu em torno de 4 e 6 metros, porém nesta configuração o erro é menos visível. Acima de 6 metros os valores calculados estão bem próximos aos observados.

Ambas as redes apresentaram bons resultados, no entanto, conforme destacado anteriormente, apenas uma configuração deve ser eleita a melhor LSTM, para então ser comparada com a melhor configuração de GRU. Assim sendo, na comparação entre as redes com 3 e 5 camadas ocultas, a rede *LSTM_3C_64CE_A_leakyrely_0.2_128TS_128B_Drpt_0.01* obteve o menor MAPE (**2.026%**), enquanto a rede *LSTM_5C_64CE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1* obteve o menor RMSE (**5.040E-03**) e o maior NSE (**9.980E-01**). Desta forma, a LSTM que produziu os reusltados com menores erros de acordo com as métricas utilizadas, foi a rede com 5 camadas ocultas e, portanto, é eleita a melhor LSTM.

5.3 TREINAMENTO E RESULTADOS DA RNN GRU

As próximas subseções descrevem o processo de treinamento e os resultados obtidos com a utilização das redes neurais recorrentes *Gated Recurrent Unit* (GRU).

5.3.1 Treinamento

Um total de 41 configurações de redes GRU foram treinadas. De forma análoga aos modelos LSTM, as redes com menos camadas tiveram mais configurações treinadas em relação aos modelos com mais camadas, pois quanto mais camadas se tinha, mais dispendioso era o tempo de treinamento.

O ajuste dos pesos e limiares da rede durante o treinamento também foram realizados através da função de erro *Mean Square Root* (MSE). Desta forma, tomando-se como base o MSE produzido pela rede durante tal fase, na Tabela 16 é apresentada a configuração que obteve o melhor desempenho.

Tabela 16 – Melhor desempenho GRU durante o treinamento.

N° Camadas	**N° Células**	**Função de Ativação**	**Regularização**	**Épocas**	**MSE**
4	64	*Tanh*	*Dropout* – 0.01	210	1.0924E-05

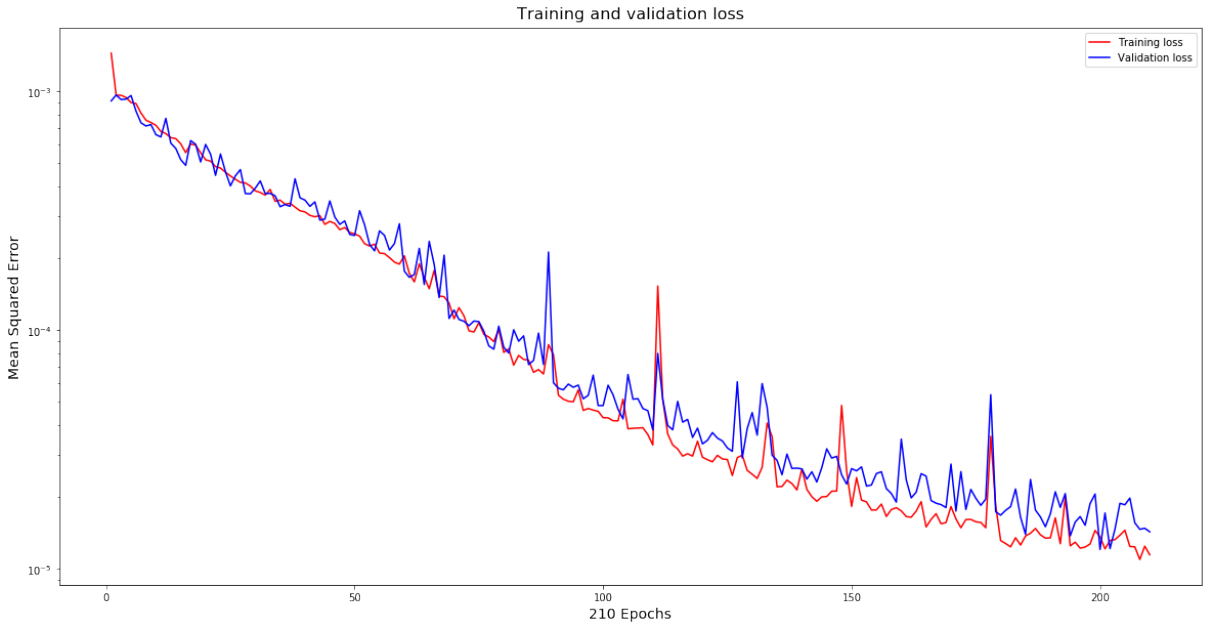
Fonte: Elaborado pelo autor.

A configuração GRU que exibiu o menor erro durante a fase de treinamento conforme exibe a Tabela 16, chegou a tal resultado durante a época 209, com 64 células nas camadas ocultas, função de ativação *Tanh*, regularização *Dropout* com probabilidade de 1%. Deve-se reforçar que esse resultado foi obtido através do conjunto de dados de treinamento, em relação ao conjunto de validação o menor MSE obtido foi de 1.2028E-05 durante a época 201.

alienta-se que, embora tenha se estipulado um máximo de 200 épocas para o treinamento das RNAs, este modelo foi o único a chegar na época 200 sem interrupções, além disso, os resultados mostravam que a rede poderia melhorar ainda mais com mais épocas de treino, dessa forma tal configuração foi novamente treinada, desta vez com um número maior de épocas, sendo esta quantidade de 300 épocas.

A Figura 29 ilustra a evolução do erro durante o treinamento da rede com o melhor desempenho. Pode-se observar que o treinamento foi interrompido abruptamente durante a época 210. Em contraste a rede LSTM que obteve o melhor desempenho durante a fase de treinamento, a rede GRU apresentou menos ruídos nas curvas dos erros. Além disso, embora nas últimas épocas as curvas se distanciam um pouco, a rede não mostrou sinais de sobre ajuste, da mesma forma que é observado na Figura 26, onde as curvas apresentam uma distância maior entre si.

Figura 29 – Evolução do erro do modelo GRU com melhor desempenho durante o treinamento.



Fonte – Acervo do autor.

O erro exibido através do MSE calculado pelas RNAs durante o treinamento nos conjuntos de treinamento e validação, exibem a rede que obteve o melhor ajuste nesta fase, porém assim como nas redes LSTM, o interesse maior é pelos resultados produzidos durante a fase de testes. As previsões realizadas utilizando o conjunto de teste ajudaram a definir a rede GRU que fez as previsões com menor erro, que posteriormente foram comparadas com a melhor LSTM, quando então se pode eleger a melhor configuração de RNN para a área de estudo selecionada.

5.3.2 Resultados

Os resultados produzidos pelas configurações GRU quando apresentadas ao conjunto de testes são apresentados nas tabelas a seguir. A configuração dos modelos é identificada através do mesmo sistema de siglas descrito na seção 5.2.2.

O menor RMSE (**6.772E-03**), menor MAPE (**2.669%**) e maior NSE (**9.965E-01**), foram obtidos pelo modelo GRU de uma camada oculta com 128 células nesta camada, função de ativação *ReLU*, 128 sequências temporais, 128 amostras por *batch*, e regularização *Dropout* com 10% de probabilidade.

Tabela 17 – Resultados de cada configuração da rede recorrente GRU com 1 camada oculta.

GRU – 1 CAMADA				
Modelo	Epochs	RMSE	MAPE %	NSE
GRU_1C_32CE_A_tanh_32TS_32B	26	2.867E-02	7.853	9.360E-01
GRU_1C_32CE_A_tanh_32TS_64B	63	4.627E-02	12.751	8.065E-01
GRU_1C_32CE_A_tanh_64TS_32B	100	1.678E-02	7.214	9.773E-01
GRU_1C_64CE_A_leakyrelu_0.1_64TS_32B_Drpt_0.1	100	1.652E-02	5.700	9.780E-01
GRU_1C_32CE_A_tanh_128TS_128B	186	1.375E-02	5.051	9.862E-01
GRU_1C_64CE_A_tanh_128TS_128B	83	8.934E-02	49.961	3.657E-01
GRU_1C_64CE_A_leakyrelu_0.2_128TS_128B_Drpt_0.1	70	1.701E-02	5.985	9.773E-01
GRU_1C_64CE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	189	9.340E-03	3.821	9.932E-01
GRU_1C_64CE_A_relu_128TS_128B_Drpt_0.1	145	1.231E-02	4.596	9.883E-01
GRU_1C_128CE_A_relu_128TS_128B_Drpt_0.1	164	**6.772E-03**	**2.669**	**9.965E-01**
GRU_1C_128CE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	125	8.312E-03	3.752	9.946E-01

Fonte – Acervo do autor.

Analogamente ao modelo LSTM, a configuração GRU que mostrou o melhor resultado possuí um número maior de parâmetros em relação as outras configurações de 1 camada oculta. A configuração que exibiu o maior erro, chegou ao maior RMSE (**8.934E-02**), maior MAPE (**49.961%**) e menor NSE (**3.657E-01**) com 64 células na camada oculta, função de ativação *Tanh*, 128 sequências temporais, 128 amostras por *batch*.

O modelo GRU de 1 camada que exibiu o menor erro em todas as métricas, é posteriormente comparado com os outros modelos com mais camadas selecionados durante a análise dos resultados. A Tabela 18 a seguir exibe os resultados obtidos pelo modelo GRU com 2 camadas ocultas.

Tabela 18 – Resultados de cada configuração da rede recorrente GRU com 2 camadas ocultas.

GRU – 2 CAMADAS				
Modelo	Epochs	RMSE	MAPE %	NSE
GRU_2C_32CE_A_tanh_32TS_ 32B	125	1.882E-02	6.142	9.724E-01
GRU_2C_32CE_A_tanh_64TS_ 32B	44	2.085E-02	6.418	9.650E-01
GRU_2C_64CE_A_leakyrelu_0.1_64TS_32 B_Drpt_0.1	92	1.200E-02	3.737	9.884E-01
GRU_2C_64CE_A_tanh_128TS_ 128B	164	5.363E-03	2.195	9.977E-01
GRU_2C_64CE_A_tanh_128TS_128B_Drpt _0.01	249	**5.018E-03**	**2.184**	**9.981E-01**
GRU_2C_64CE_A_leakyrelu_0.3_128TS_12 8B_Drpt_0.1	185	7.556E-03	3.176	9.955E-01
GRU_2C_64CE_A_leakyrelu_0.2_128TS_12 8B_Drpt_0.1	136	9.735E-03	3.289	9.927E-01
GRU_2C_64CE_A_relu_128TS_128B_Drpt_ 0.1	186	8.891E-03	2.703	9.939E-01

Fonte – Acervo do autor.

De maneira similar ao modelo com uma camada oculta, apenas uma configuração GRU com duas camadas ocultas obteve o melhor desempenho em todas as métricas. Tal configuração exibiu o menor RMSE (**5.018E-03**), menor MAPE (**2.184%**) e maior NSE (**9.981E-01**); obtendo tal resultado com 64 células nas camadas ocultas, função de ativação *Tanh*, 128 sequências temporais, 128 amostras por *batch*, e 1% de *Dropout*.

A rede que apresentou tal desempenho chegou a esse resultado sendo treinada durante 249 épocas, onde o treinamento foi interrompido. Em contrapartida, o modelo com duas camadas ocultas que apresentou o pior desempenho teve seu treinamento interrompido durante a época 44, pois já não apresentava melhoras significativas durantes as dez últimas épocas. Tal modelo apresentou o maior RMSE (**2.085E-02**), maior MAPE (**6.418E%**) e menor NSE (**9.650E-01**) configurado com 32 células nas camadas ocultas, função de ativação *Tanh*, 64 sequências temporais e 32 amostras por *batch*.

De forma geral, as redes que apresentaram os melhores resultados foram treinadas por mais épocas devido ao bom desempenho exibido durante o treinamento, inibindo assim a interrupção abrupta e permitindo o aperfeiçoamento do conhecimento adquirido pelo modelo através das atualizações dos pesos e limiares, realizadas no decorrer desta fase.

Evidentemente os modelos que até aqui apresentaram o melhor desempenho em

todas as métricas, são automaticamente selecionados para a comparação final. Desta forma, segue-se a análise das configurações com mais camadas, assim sendo, na Tabela 19 são exibidas as configurações treinadas com 3 camadas ocultas.

Tabela 19 – Resultados de cada configuração da rede recorrente GRU com 3 camadas ocultas.

GRU – 3 CAMADAS				
Modelo	Epochs	RMSE	MAPE %	NSE
GRU_3C_64CE_A_leakyrelu_0.1_64TS_32B_Drpt_0.1	100	6.478E-03	2.631	9.966E-01
GRU_3C_64CE_A_leakyrelu_0.1_64TS_64B_Drpt_0.1	100	7.187E-03	3.031	9.959E-01
GRU_3C_64CE_A_tanh_128TS_128B_Drpt_0.01	192	8.084E-03	**1.969**	9.950E-01
GRU_3C_64CE_A_leakyrelu_0.2_128TS_128B_Drpt_0.1	143	**6.302E-03**	2.570	**9.969E-01**
GRU_3C_64CE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	147	6.644E-03	2.785	9.965E-01
GRU_3C_32CE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	109	1.274E-02	4.868	9.873E-01
GRU_3C_128CE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	131	6.765E-03	2.734	9.964E-01
GRU_3C_128CE_A_relu_128TS_128B_Drpt_0.1	104	7.643E-03	2.979	9.954E-01

Fonte – Acervo do autor.

Nas configurações que possuem 3 camadas ocultas, um único modelo apresentou o menor MAPE (**1.969%**). Tal modelo chegou a esse resultado configurado com 64 células nas camadas ocultas, função de ativação *Tanh*, 128 sequências temporais, 128 amostras por batch, e 1% de *Dropout*.

No entanto, uma única configuração se mostrou superior nas outras duas métricas utilizadas, ou seja, exibiu a melhor performance obtendo o menor RMSE (**6.302E-03**) e o maior NSE (**9.969E-01**). O modelo em questão chegou a este resultado com 4 células nas camadas ocultas, função de ativação *LeakyReLU* com α 0.2, 128 sequências temporais, 128 amostras por *batch*, e 10% de *Dropout*.

Embora o MAPE do primeiro modelo seja bem superior ao do segundo, o posterior por ter exibido o menor erro em duas das métricas utilizadas, logo foi selecionado para a comparação final que visa escolher o modelo GRU com o melhor desempenho.

Em relação a configuração com o pior desempenho, um único modelo exibiu os piores resultados em todas as métricas utilizadas. Obtendo o maior RMSE (**1.274E-02**), maior MAPE (**4.868%**) e menor NSE (**9.873E-01**); obteve tal resultado configurado com 32 células

nas camadas ocultas, função de ativação *LeakyReLU* com α 0.3, 128 sequências temporais, 128 amostras por *batch*, e 10% de *Dropout*.

Na sequência a Tabela 20 exibe os resultados obtidos pelas configurações GRU com 4 camadas ocultas.

Tabela 20 – Resultados de cada configuração da rede recorrente GRU com 4 camadas ocultas.

GRU – 4 CAMADAS				
Modelo	Epochs	RMSE	MAPE %	NSE
GRU_4C_64CE_A_leakyrelu_0.1_64TS_64B_Drpt_0.1	97	7.933E-03	3.210	9.951E-01
GRU_4C_64CE_A_tanh_128TS_128B_Drpt_0.01	210	**3.392E-03**	**1.403**	**9.991E-01**
GRU_4C_64CE_A_leakyrelu_0.1_128TS_128B_Drpt_0.1	70	6.617E-03	2.725	9.965E-01
GRU_4C_32CE_A_leakyrelu_0.1_128TS_128B_Drpt_0.1	66	1.081E-02	5.137	9.903E-01
GRU_4C_64CE_A_leakyrelu_0.1_128TS_64B_Drpt_0.1	75	7.682E-03	3.107	9.955E-01
GRU_4C_64CE_A_leakyrelu_0.2_128TS_64B_Drpt_0.1	119	5.605E-03	2.263	9.975E-01
GRU_4C_64CE_A_leakyrelu_0.3_128TS_64B_Drpt_0.1	93	8.546E-03	3.734	9.942E-01

Fonte – Acervo do autor.

Assim como a maioria das outras configurações analisadas, um único modelo com 4 camadas apresentou o menor erro em todas as métricas durante a fase de testes. Tal modelo obteve o menor RMSE (**3.392E-03**), menor MAPE (**1.403%**) e maior NSE (**9.991E-01**) configurado com 64 células nas camadas ocultas, função de ativação *Tanh*, 128 sequências temporais, 128 amostras por *batch*, e 1% de *Dropout*.

A configuração em questão foi treinada por 210 épocas, além de ter apresentado os menores erros nas métricas, também exibiu o melhor desempenho durante a fase de treinamento, conforme destacado no ínicio desta subseção.

O modelo com 4 camadas ocultas que exibiu o pior desempenho durante a fase de testes, obteve o maior RMSE (**1.081E-02**), maior MAPE (**5.137%**) e menor NSE (**9.903E-01**), configurado com 32 células nas camadas ocultas, função de ativação *LeakyReLU* com α 0.1, 128 sequências temporais, 128 amostras por *batch*, e 10% de *Dropout*.

A configuração que mostrou o pior desempenho teve o treinamento interrompido durante a época 66, ou seja, 144 épocas a menos que o modelo que apresentou o menor erro. De forma geral, as redes que foram treinadas por mais tempo, apresentaram os melhores resultados.

A Tabela 21 exibe os resultados obtidos pelas configurações com 5 camadas ocultas. Os resultados são analisados e um único modelo é selecionado seguindo-se a sistemática adotada neste trabalho. Na sequência os melhores modelos para cada conjunto de camadas são analisados e discutidos. Por último um único modelo é selecionado como o melhor modelo GRU.

Tabela 21 – Resultados de cada configuração da rede recorrente GRU com 5 camadas ocultas.

GRU – 5 CAMADAS				
Modelo	Epochs	RMSE	MAPE %	NSE
GRU_5C_32CE_A_leakyrelu_0.1_128TS_128B_Drpt_0.1	98	8.200E-03	3.194	9.948E-01
GRU_5C_32CE_A_tanh_128TS_128B_Drpt_0.1	71	1.075E-02	4.369	9.908E-01
GRU_5C_64CE_A_leakyrelu_0.1_128TS_128B_Drpt_0.1	103	**5.311E-03**	**2.080**	**9.978E-01**
GRU_5C_64CE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	108	6.201E-03	2.540	3.162E+00
GRU_5C_64CE_A_relu_128TS_128B_Drpt_0.1	74	7.603E-03	3.348	9.954E-01
GRU_5C_64CE_A_leakyrelu_0.2_128TS_128B_Drpt_0.1	90	7.264E-03	2.866	9.962E-01
GRU_5C_64CE_A_tanh_128TS_128B_Drpt_0.1	123	7.005E-03	3.162	9.962E-01

Fonte – Acervo do autor.

A configuração com 5 camadas ocultas que exibiu o melhor desempenho dentre as outras configurações de mesma profundidade, obteve o menor RMSE (**5.311E-03**), menor MAPE (**2.080%**) e maior NSE (**9.978E-01**). Sendo treinada por 103 épocas, obteve tais resultados com 64 células nas camadas ocultas, função de ativação *LeakyReLU* com α 0.1, 128 sequências temporais, 128 amostras por *batch*, e 10% de *Dropout*.

O maior RMSE (**1.075E-02**), maior MAPE (**4.369%**) e menor NSE (**9.908E-01**), são resultantes do modelo de 5 camadas que exibiu o pior desempenho dentre os outros, chegando a este resultado com 32 células nas camadas ocultas, função de ativação *Tanh*, 128 sequências temporais, 128 amostras por *batch*, e 10% de *Dropout*.

Em vista de todas as configurações avaliadas até aqui, não se observou nenhuma tendência nos resultados em relação a função de ativação utilizada nas camadas ocultas. Diferentemente do modelo LSTM que exibiu os piores resultados nas configurações com a função *Tanh*, os modelos GRU, por outro lado, mostraram bom desempenho nas configurações com tal função nas camadas ocultas.

Uma vez que as melhores configurações para cada grupo de camadas foram selecionadas, as tais são dispostas na Tabela 22, os resultados são comparados a fim de eleger a melhor rede GRU.

Tabela 22 – Melhores desempenhos obtidos por grupos de GRUs com *n* camadas ocultas.

GRU – Melhores Resultados				
Modelo	Epochs	RMSE	MAPE %	NSE
GRU_1C_128CE_A_relu_128TS_128B_Drpt_0.1	164	6.772E-03	2.669	9.965E-01
GRU_2C_64CE_A_tanh_128TS_128B_Drpt_0.01	249	5.018E-03	2.184	9.981E-01
GRU_3C_64CE_A_leakyrelu_0.2_128TS_128B_Drpt_0.1	143	6.302E-03	2.570	9.969E-01
GRU_4C_64CE_A_tanh_128TS_128B_Drpt_0.01	210	**3.392E-03**	**1.403**	**9.991E-01**
GRU_5C_64CE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	103	5.311E-03	2.080	9.978E-01

Fonte – Acervo do autor.

Uma vez que todas as configurações para cada grupo com n camadas ocultas foram analisadas, os modelos GRU que exibiram os melhores desempenho dentro de seus grupos foram dispostas nas Tabela 22 semelhantemente ao que foi feito com os melhores modelos LSTM. No entanto, desta vez um modelo com menos camadas ocultas obteve os menores erros nas métricas adotadas na pesquisa.

O modelo com 4 camadas ocultas, 64 células em cada camada, função de ativação *Tanh*, 128 sequências temporais, 128 amostras por batch, e 1% de *Dropout*; obteve o menor RMSE (**3.392E-03**), menor MAPE (**1.403%**) e o maior NSE (**9.991E-01**).

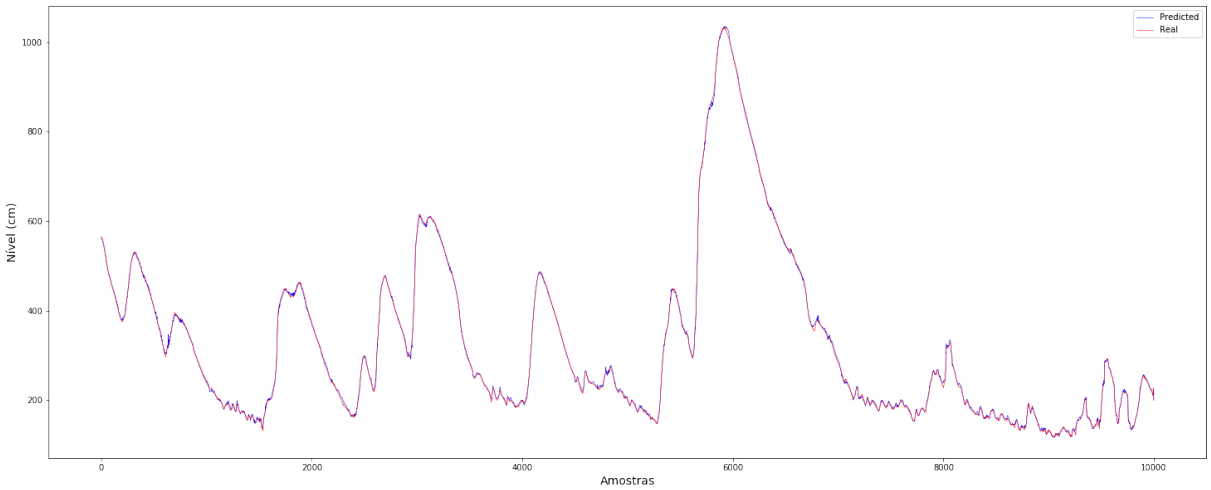
Dentre todos os modelos, este em particular se sobressaiu exibindo o menor erro em todas as métricas em relação a todas as outras configurações, desta forma não exigindo a necessidade de ser comparado e/ou usar outros critérios de avaliação e portanto é eleito a melhor configuração GRU.

Em relação a rede com o pior desempenho, assim como ocorreu com as melhores configurações LSTM, a rede GRU que teve o pior desempenho também foi a rede com a menor quantidade de camadas. O modelo com 1 camada oculta *GRU_1C_128CE_A_relu_128TS_128B_Drpt_0.1* exibiu o maior RMSE (**6.772E-03**), maior MAPE (**2.669%**) e menor NSE (**9.965E-01**) dentre todas as melhores configurações GRU para cada conjunto de camadas.

O gráfico com os resultados dos valores do nível do rio calculado e observado para

a configuração que o obteve os melhores RMSE, MAPE e NSE, é apresentado através da Figura 30.

Figura 30 – Gráfico do nível calculado e observado do modelo com 4 camadas ocultas *GRU_4C_64CE_A_tanh_128TS_128B_Drpt_0.01*.



Fonte – Acervo do autor.

Conforme se pode observar na Figura 30, os níveis calculados pela GRU com 4 camadas ocultas foram muito próximos dos observados. Algumas pequenas diferenças ocorreram entre 2 e 4.5 metros, onde a rede previu resultados acima dos observados. No entanto, para valores acima de 4.5 as diferenças são imperceptíveis, ou seja, os valores estão com praticamente nenhuma diferença.

Esses resultados refletem que o modelo em questão é adequado para realizar previsão de inundações no local de estudo selecionado devido ao baixo erro apresentado. Até aqui os modelos profundos, com mais níveis de profundidade tem-se mostrado mais adequados ao apresentarem um índice de erro menor em relação aos modelos mais rasos.

5.4 ESCOLHA DA MELHOR CONFIGURAÇÃO DE REDE NEURAL RECORRENTE

Após o treinamento e testes das redes LSTM e GRU, cada rede neural foi analisada de acordo com seu nível de profundidade e de maneira individual, ou seja, para cada conjunto com n camadas cada modelo de uma determinada topologia foi comparado com configurações de mesma topologia e de mesma quantidade de camadas.

As configurações que apresentaram o melhor desempenho nas métricas de erro para cada conjunto de n camadas, foram novamente agrupadas, analisadas e comparadas entre si, de

modo a se eleger a melhor configuração para cada uma das vertentes de RNNs. Desse modo, analisando as métricas de erro e os gráficos de comparação entre nível do rio previsto e observado, foram definidas as configurações LSTM e GRU que produziram os resultados mais satisfatórios.

A melhor configuração LSTM, ou seja, o modelo desta topologia que produziu os resultados com menores erros, foi a alternativa com 5 camadas ocultas, 64 células em cada camada, função de ativação *LeakyReLU* com α 0.2, 128 sequências temporais, 128 amostras por *batch* e 1% de *Dropout*. Esta configuração obteve o melhor desempenho dentre todas as configurações LSTM, exibindo o menor RMSE, menor MAPE e maior NSE, com valores de **5.040E-03**, **2.026%** e **9.980E-01** respectivamente.

O modelo GRU que exibiu o melhor desempenho, ou seja, a configuração que apresentou os menores índices de erro nas métricas adotadas, possuí 4 camadas ocultas, 64 células em cada camada, função de ativação *Tanh*, 128 sequências temporais, 128 amostras por *batch*, e 1% de *Dropout*. Analisando os três parâmetros estatísticos RMSE, MAPE e NSE, esta configuração se mostrou a mais adequada dentre todas as outras configurações GRU, com valores **3.392E-03**, **1.403%** e **9.991E-01**, respectivamente.

Em vista do desempenho apresentado pelas duas redes com os resultados mais satisfatórios e, levando-se em consideração as três métricas de erro utilizadas, a rede GRU foi a que obteve o melhor desempenho em todas as métricas em comparação aos resultados obtidos pela rede LSTM. Desta forma, não há necessidade de adotar um outro critério de avaliação, assim sendo a rede GRU é então eleita a RNN mais adequada para a comparação final que visa eleger a RNA mais adequada para as previsões do nível do rio no local de estudo selecionado e também para responder à pergunta de pesquisa inicialmente proposta.

5.5 TREINAMENTO E RESULTADOS DA RNA HÍBRIDA COM CAMADAS LSTM

As próximas subseções descrevem o processo de treinamento e os resultados obtidos com a utilização das redes neurais híbridas com camadas LSTM. A rede em questão possuí a seguinte estrutura: *CNN-Pooling-LSTM-MLP*.

5.5.1 Treinamento

As redes híbridas empregadas nesta pesquisa, se diferenciam através do tipo de rede recorrente que é usada nas camadas ocultas deste tipo de rede. Desta forma, a RNA híbrida

abordada nesta seção, possuí apenas a RNN LSTM nas camadas ocultas em conjunto com as outras camadas das topologias CNN e MLP.

Assim, um total de 63 configurações de redes híbridas com LSTM foram treinadas. Da mesma forma que as RNNs, as redes híbridas com menos complexidade, tiveram mais configurações treinadas em relação as redes com maior complexidade, devido à dispendiosidade do tempo de treino gasto com as redes com um maior número de parâmetros.

A sistemática utilizada para avaliar as redes híbridas é similar a usada para avaliar as redes recorrentes, no entanto, de maneira inversa, não são avaliadas por nível de profundidade, pois: (i) essa arquitetura possuí no mínimo 4 camadas ocultas, ou seja, já é inerentemente profunda. (ii) Além disso, as RNNs já foram o suficiente para mostrar que os modelos com maior profundidade se mostraram mais representativos em relação as modelos rasos. Portanto, são utilizadas como forma de fornecer um comparativo entre as duas topologias RNN e Híbrida, utilizadas nesta pesquisa.

O ajuste dos pesos e limiares da rede durante o treinamento, de forma similar as RNNs, também foram realizados através da função de erro MSE. Desta forma, tomando-se como base o MSE produzido pela rede durante a fase de treinamento, na Tabela 23 é apresentada a configuração que obteve o melhor desempenho.

Tabela 23 – Melhor desempenho Híbrido - LSTM durante o treinamento.

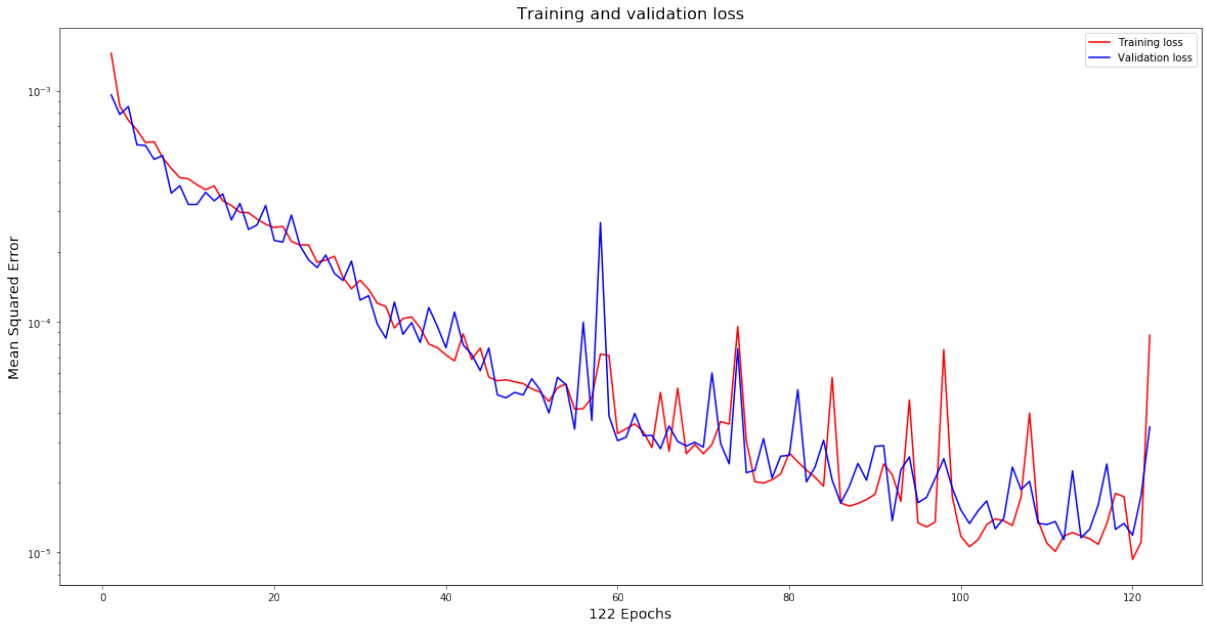
N° Camadas	**N° Células**	**Função de Ativação**	**Regularização**	**Épocas**	**MSE**
5	64	*LeakyRelu* 0.3	N/A	122	1.0088E-05

Fonte – Elaborado pelo autor.

Conforme se pode observar na Tabela 23, a rede Híbrida com LSTM nas camadas ocultas que obteve o menor MSE no conjunto de treinamento, chegou a tal resultado com 1 camada convolucional, 1 camada *Average Pooling*, 2 camadas LSTM com 64 células em cada uma e 1 camada *Dense* com 64 neurônios, totalizando 5 camadas ocultas; função de ativação *LeakyReLU* com α 0.3; 128 sequências temporais para cada uma das 128 amostras por *batch*.

A rede foi treinada até ser interrompida abruptamente durante a época 122. O MSE em questão foi obtido pela rede durante a época 111, avaliado com base no conjunto de treinamento. Em relação ao conjunto de validação, o menor MSE obtido foi de 1.1557e-05 durante a época 114. A evolução do erro durante o treinamento da rede com o melhor desempenho é exibida pela Figura 31.

Figura 31 – Evolução do erro do modelo Híbrido - LSTM com melhor desempenho durante o treinamento



Fonte – Acervo do autor.

Embora o treinamento tenha sido interrompido durante a época 122 de um total de 200 épocas por não exibir melhorias significativas no conjunto de validação, mesmo não utilizando regularização, conforme se pode observar na Figura 31, o gráfico não apresenta indícios de sobre ajuste durante tal fase.

O erro exibido através do MSE calculado durante o treinamento nos conjuntos de treinamento e validação, exibem a rede que obteve o melhor ajuste nesta fase, porém, o interesse maior é pelos resultados produzidos durante a fase de testes. As previsões realizadas utilizando o conjunto de teste ajudaram a definir a rede Híbrida com LSTM que fez as previsões com menor erro, que posteriormente foram comparadas com a melhor Híbrida com GRU, quando então se pode eleger a melhor configuração Híbrida para a área de estudo selecionada.

5.5.2 Resultados

Os resultados produzidos pelas configurações Híbridas com LSTM quando apresentadas ao conjunto de testes são apresentados na tabela 24 a seguir. Apenas as 10 configurações mais relevantes são apresentadas A configuração dos modelos é identificada através do mesmo sistema de siglas descrito na seção 5.2.2, porém com a seguinte adição:

- (j)NE: Dense com j neurônios nas camadas ocultas.

Por exemplo, o modelo com a configuração *7C_CNN_1C_MaxPool_1C_LSTM_3C_128CE_DENSE_2C_128NE_A_leakyrelu_0.3_128TS_128B*, tem o seguinte significado: Modelo Híbrido com um total de 7 camadas ocultas; contendo 1 camada Convolucional, 1 camada *Max Pooling*, 3 camadas LSTM com 128 células em cada camada, 2 camadas Dense com 128 neurônios; função de ativação *LeakyReLU* com 0.3, 128 sequências temporais e 128 amostras por *batch*.

Tabela 24 – Resultados para cada configuração da rede Híbrida LSTM com camadas ocultas variando de 4 a 14 camadas.

HÍBRIDO LSTM				
Modelo	Epochs	RMSE	MAPE %	NSE
4C_CNN_1C_MaxPool_1C_LSTM_1C_128CE_DENSE_1C_128NE_A_tanh_128TS_128B_Drpt_0.05	129	4.864E-03	1.784	9.982E-01
5C_CNN_1C_MaxPool_1C_LSTM_2C_64CE_DENSE_1C_64NE_A_tanh_128TS_128B_Drpt_0.05	100	5.263E-03	2.134	9.979E-01
6C_CNN_1C_AvgPool_1C_LSTM_2C_128CE_DENSE_2C_128NE_A_leakyrelu_0.1_128TS_128B	81	4.288E-03	1.280	9.985E-01
7C_CNN_1C_MaxPool_1C_LSTM_3C_128CE_DENSE_2C_128NE_A_leakyrelu_0.3_128TS_128B	59	4.842E-03	1.962	9.982E-01
8C_CNN_1C_MaxPool_1C_LSTM_3C_128CE_DENSE_3C_128NE_A_tanh_128TS_128B_Drpt_0.1	112	**3.702E-03**	1.434	**9.990E-01**
8C_CNN_2C_MaxPool_2C_LSTM_2C_128CE_DENSE_2C_128NE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	98	3.981E-03	**1.269**	9.988E-01
9C_CNN_2C_MaxPool_2C_LSTM_3C_64CE_DENSE_2C_64NE_A_tanh_128TS_128B_Drpt_0.05	82	4.653E-03	1.560	9.983E-01
10C_CNN_2C_AvgPool_2C_LSTM_3C_64CE_DENSE_3C_64NE_A_tanh_128TS_128B_Drpt_0.1	77	4.635E-03	1.964	9.984E-01
11C_CNN_1C_MaxPool_1C_LSTM_5C_128CE_DENSE_4C_128NE_A_tanh_128TS_128B_Drpt_0.1	70	6.390E-03	2.736	9.967E-01
14C_CNN_2C_MaxPool_2C_LSTM_5C_128CE_DENSE_5C_128NE_A_tanh_128TS_128B_Drpt_0.1	31	1.160E-01	42.910	-5.579E-03

Fonte – Elaborado pelo autor.

O menor MAPE (**1.269%**) foi obtido pela configuração treinada com 8 camadas ocultas, constituídas de: 2 camadas convolucionais; 2 camadas Max Pooling; 2 camadas LSTM

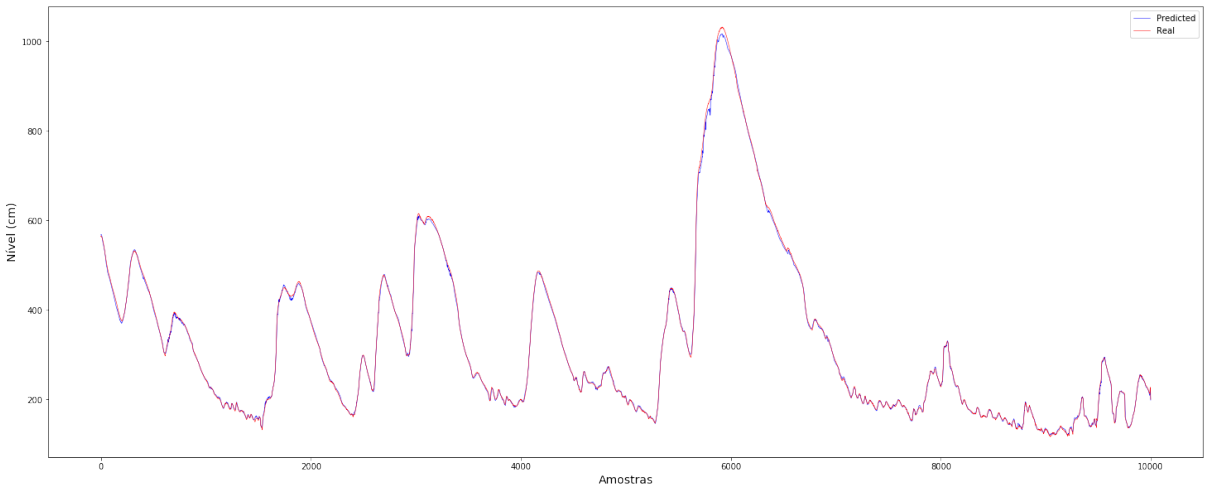
com 128 células cada; 2 camadas Dense com 128 neurônios cada; função de ativação *Leaky-ReLU* com α 0.3; 128 sequências temporais; 128 amostras por batch; e 10% de *Dropout*.

O menor RMSE (**3.702E-03**) e maior NSE (**9.990E-01**) foram obtidos também por uma configuração com 8 camadas ocultas, especificamente configurada com: 1 camada convolucional; 1 camada *Max Pooling*; 3 camadas LSTM com 128 células cada; 3 camadas *Dense* com 128 neurônios cada; função de ativação *Tanh*; 128 sequências temporais; 128 amostras por *batch*; e 10% de *Dropout*.

Em relação ao pior desempenho, uma única configuração com 14 camadas ocultas exibiu o maior RMSE (**1.160E-01**), maior MAPE (**42.910%**) e menor NSE (**-5.579E-03**), configurada com: 2 camadas convolucionais; 2 camadas *Max Pooling*; 5 camadas LSTM com 128 células cada; 5 camadas *Dense* com 128 neurônios cada; função de ativação *Tanh*; 128 sequências temporais; 128 amostras por *batch*; e 10% de *Dropout*.

A rede que exibiu o menor MAPE (**1.269E**), obteve resultados nas métricas RMSE (**3.981E-03**) e NSE (**9.988E-01**), próximos dos resultados da rede que obteve os menores RMSE e NSE. Assim, como forma também de comparação a Figura 32 e Figura 33 exibem, respectivamente, os gráficos com os resultados dos valores do nível do rio calculado e observado para a rede que exibiu o menor MAPE e a rede que apresentou o menor RMSE e maior NSE.

Figura 32 – Gráfico do nível calculado e observado do modelo com 8 camadas ocultas que obteve o menor MAPE.



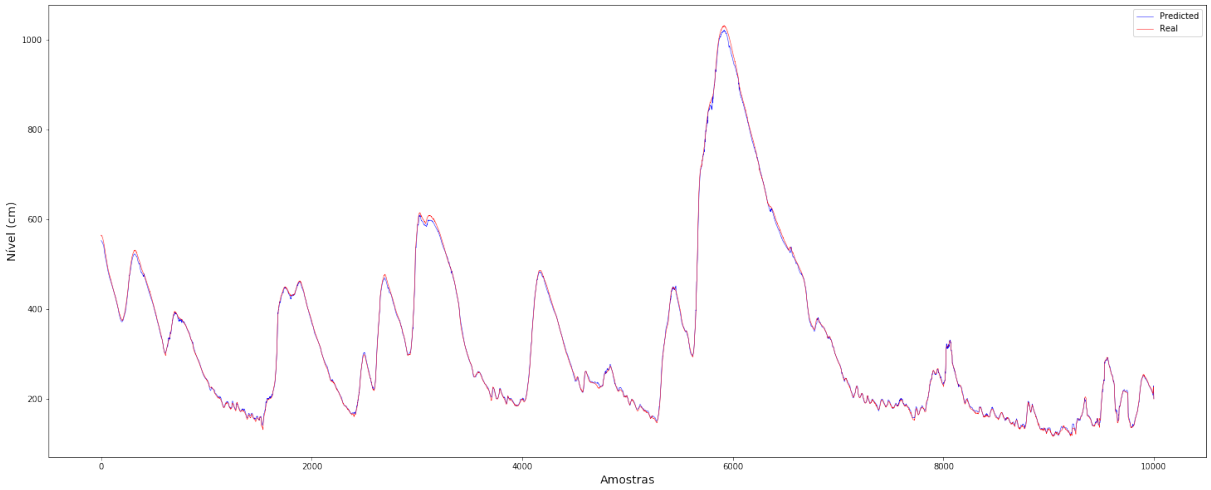
Fonte – Acervo do autor.

Conforme se pode observar na Figura 32, os níveis calculados pela rede com o menor MAPE, foram próximos dos observados, no entanto, é possível observar algumas diferenças

em torno de 4.5 e 6 metros, onde a rede calculou valores um pouco abaixo dos observados. Mais adiante, por volta dos 8 metros, é possível observar algumas discrepâncias, porém em torno dos 10 metros é onde aparece o maior erro cometido por essa rede, que produziu resultados alguns centímetros abaixo do observado.

Em contrapartida, na Figura 33 onde são exibidos os níveis calculados pela rede com o menor RMSE e maior NSE, a rede apresenta um erro um pouco maior em torno dos 6 metros, porém por volta de 8 e 10 metros, o erro exibido é muito menor, que o erro mostrado pela rede com o menor MAPE. Os níveis calculados estão bem próximos dos níveis observados.

Figura 33 – Gráfico do nível calculado e observado do modelo com 8 camadas ocultas que obteve o menor RMSE e maior NSE.



Fonte – Acervo do autor.

Em vista dos resultados obtidos nas métricas de erro que, foram exibidos por ambas as redes, ainda se considerando a análise realizada através dos gráficos de ambas resultantes da previsão de ambos os modelos em questão. O modelo que mostrou o melhor desempenho nas métricas RMSE e NSE, também realizou melhores previsões do nível do rio, desta forma tal modelo é eleito a melhor configuração Híbrida LSTM a ser utilizada na comparação com a melhor rede Híbrida GRU.

5.6 TREINAMENTO E RESULTADOS DA RNA HÍBRIDA COM CAMADAS GRU

As próximas subseções descrevem o processo de treinamento e os resultados obtidos com a utilização das redes neurais híbridas com camadas GRU. A rede em questão possuí a seguinte estrutura: *CNN-Pooling-GRU-MLP*.

5.6.1 Treinamento

A RNA híbrida abordada nesta seção, possuí apenas a RNN GRU nas camadas ocultas em conjunto com as outras camadas das topologias CNN e MLP.

Um total de 73 configurações de redes híbridas com GRU foram treinadas. De forma análoga as outras RNAs usadas na pesquisa, as redes com maior complexidade, foram treinadas em número reduzido, enquanto que as redes com menos complexidade, tiveram mais configurações treinadas e avaliadas.

O ajuste dos pesos e limiares da rede durante o treinamento também foram realizados através da função de erro MSE. Desta forma, tomando- se como base o MSE produzido pela rede durante tal fase, na Tabela 25 é apresentada a configuração que obteve o melhor desempenho.

Tabela 25 – Melhor desempenho Híbrido - GRU durante o treinamento.

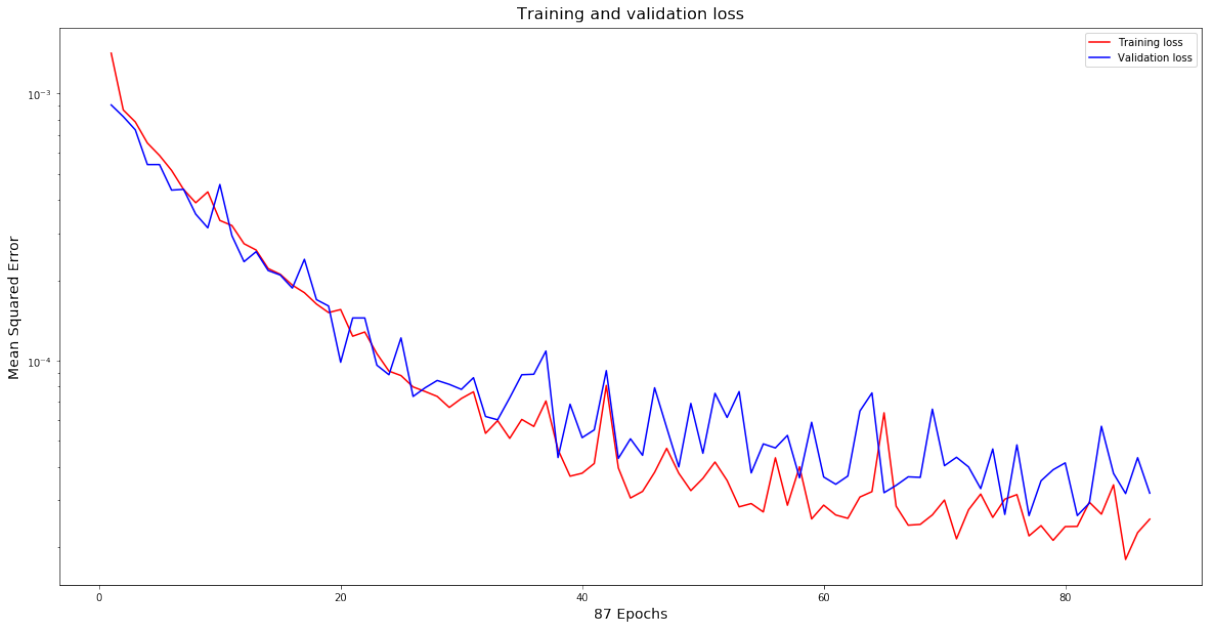
N° Camadas	**N° Células e Neurônios**	**Função de Ativação**	**Regularização**	**Épocas**	**MSE**
9	128	*LeakyReLU* 0.3	*Dropout* 0.1	87	1.8001E-05

Fonte – Elaborado pelo autor.

Conforme pode se observar na Tabela 25, a rede Híbrida com GRU nas camadas ocultas que obteve o menor MSE no conjunto de treinamento, chegou a tal resultado com 2 camadas convolucionais, 2 camadas *Max Pooling*, 3 camadas LSTM com 128 células em cada uma e 2 camadas *Dense* com 128 neurônios, totalizando 9 camadas ocultas; função de ativação *LeakyReLU* com α 0.3; 128 sequências temporais para cada uma das 128 amostras por *batch* e 10% de *Dropout*.

A rede foi treinada até ser interrompida abruptamente durante a época 87. O MSE em questão foi obtido pela rede durante a época 85, avaliado com base no conjunto de treinamento. Em relação ao conjunto de validação, o menor MSE obtido foi de 2.6264e-05 durante a época 77. A evolução do erro durante o treinamento da rede com o melhor desempenho é exibida pela Figura 34.

Figura 34 – Evolução do erro do modelo Híbrido - GRU com melhor desempenho durante o treinamento.



Fonte – Acervo do autor.

Embora o treinamento tenha sido interrompido durante a época 87 de um total de 200 épocas, a curva plotada na Figura 34 não apresenta indícios de sobre ajustes durante a fase do treinamento. Além disso, coincidentemente, essa configuração em específico foi a que exibiu melhor performance dentre todas as redes treinadas, pois obteve o menor erro em duas das três métricas estatísticas utilizadas no trabalho.

Desta forma, a subseção seguinte exibe os resultados obtidos pelas configurações obtidas durante a fase de teste. Os resultados são analisados e discutidos, ao final o modelo Híbrido – GRU com o melhor desempenho é eleito.

5.6.2 Resultados

Os resultados produzidos pelas configurações Híbridas com GRU quando apresentadas ao conjunto de testes são mostrados na tabela 26 a seguir. A configuração dos modelos é identificada através do mesmo sistema de siglas descrito na subseção 5.2.2 e complementado na subseção 5.5.2.

A mesma forma que foi feito na subseção 5.5.2, nesta subseção, apenas os 10 resultados mais relevantes são exibidos.

Tabela 26 – Resultados para cada configuração da rede Híbrida GRU com camadas ocultas variando de 4 a 14 camadas.

HÍBRIDO GRU				
Modelo	Epochs	RMSE	MAPE %	NSE
4C_CNN_1C_MaxPool_1C_GRU_1C_128CE_DENSE_1C_128NE_A_leakyrelu_0.3_128TS_128B_Drpt_0.05	167	1.151E-02	4.533	9.895E-01
5C_CNN_1C_AvgPool_1C_GRU_2C_64CE_DENSE_1C_64NE_A_leakyrelu_0.3_128TS_128B	129	4.335E-03	1.735	9.985E-01
6C_CNN_1C_MaxPool_1C_GRU_2C_32CE_DENSE_2C_32NE_A_leakyrelu_0.1_32TS_32B_Drpt_0.1	91	1.713E-02	6.542	9.768E-01
7C_CNN_1C_MaxPool_1C_GRU_3C_64CE_DENSE_2C_64NE_A_tanh_0.3_tanh_64TS_64B_Drpt_0.1	115	7.138E-03	2.467	9.962E-01
8C_CNN_2C_MaxPool_2C_GRU_2C_128CE_DENSE_2C_128NE_A_leakyrelu_0.3_128TS_128B_Drpt_0.05	80	4.084E-03	**1.249**	9.987E-01
9C_CNN_2C_MaxPool_2C_GRU_3C_128CE_DENSE_2C_128NE_A_leakyrelu0.3_128TS_128B_Drpt_0.1	87	**3.418E-03**	1.276	**9.991E-01**
10C_CNN_2C_MaxPool_2C_GRU_3C_128CE_DENSE_3C_128NE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	62	4.706E-03	1.765	9.984E-01
11C_CNN_1C_AvgPool_1C_GRU_5C_128CE_DENSE_4C_128NE_A_relu_128TS_128B_Drpt_0.1	44	7.341E-03	2.918	9.959E-01
12C_CNN_3C_AvgPool_3C_GRU_3C_128CE_DENSE_3C_128NE_A_relu_128TS_128B_Drpt_0.1	64	6.358E-03	2.373	9.968E-01
14C_CNN_3C_MaxPool_3C_GRU_4C_64CE_DENSE_4C_64NE_A_tanh_128TS_64B_Drpt_0.1	76	8.821E-03	3.313	9.942E-01

Fonte – Elaborado pelo autor.

O menor MAPE (**1.249%**) foi obtido pelo modelo treinado com 8 camadas ocultas, cuja configuração é constituída de: 2 camadas convolucionais; 2 camadas *Max Pooling*; 2 camadas GRU com 128 células cada; 2 camadas *Dense* com 128 neurônios cada; função de ativação *LeakyReLU* com α 0.3; 128 sequências temporais; 128 amostras por *batch*; e 5% de *Dropout*.

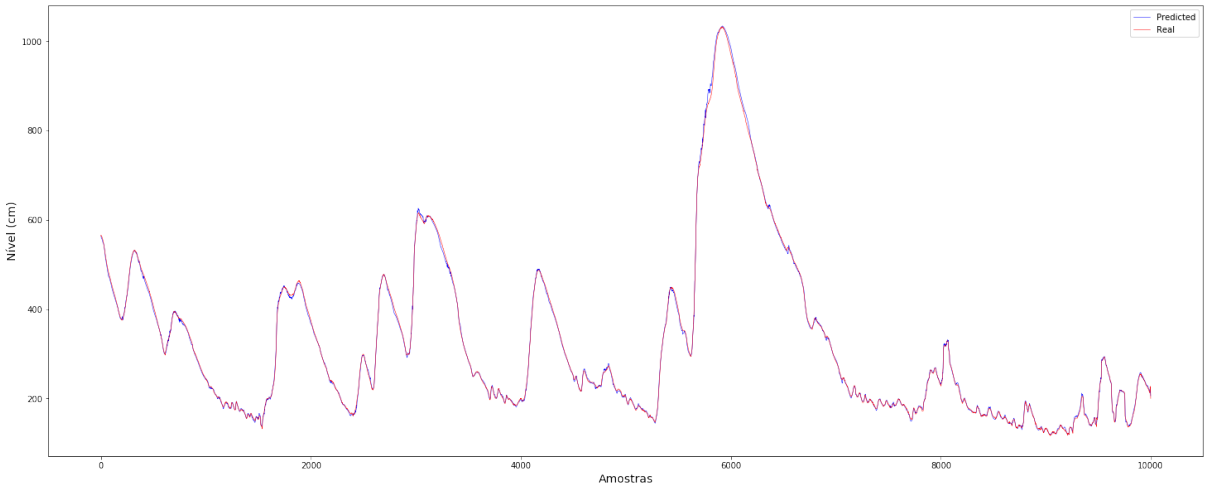
O menor RMSE (**3.418E-03**) e maior NSE (**9.991E-01**) foram obtidos por uma configuração com 9 camadas ocultas, especificamente configurada com: 2 camadas convolucionais; 2 camadas *Max Pooling*; 3 camadas GRU com 128 células cada; 2 camadas *Dense* com 128 neurônios cada; função de ativação *LeakyReLU* com α 0.3; 128 sequências temporais; 128

amostras por *batch*; e 10% de *Dropout*.

A configuração que obteve o melhor desempenho em duas das três métricas utilizadas, foi eleita como a melhor modelo, pois (i) o RMSE (**4.084E-03**) e NSE (**9.987E-01**) da rede que obteve o menor MAPE, ficaram mais distantes da rede que obteve o menor RMSE e maior NSE, em contraste aos resultados das redes Híbridas LSTM que obtiveram os menores erros de acordo com as métricas em questão; (ii) a análise realizada na seção 5.5.2, mostraram que a rede que demonstrou um erro menor em duas das três métricas utilizadas, foi a configuração que exibiu o melhor desempenho.

A Figura 35 a seguir exibe o gráfico com os resultados dos valores do nível do rio calculado e observado para a rede que exibiu o apresentou o menor RMSE e maior NSE.

Figura 35 – Gráfico do nível calculado e observado do modelo com 9 camadas ocultas que obteve o menor RMSE e maior NSE.



Fonte – Acervo do autor.

Conforme se pode observar através da Figura 35, os níveis calculados estão muito próximos dos observados. A única diferença ocorreu em torno dos 8 metros e 8.5 metros, onde o modelo calculou os níveis alguns centímetros acima dos observados. Para o restante, os níveis calculados não apresentam diferença em relação aos observados.

Em relação a rede que apresentou o pior desempenho, esta exibiu o maior RMSE (**1.713E-02**), maior MAPE (**6.542%**) e menor NSE (**9.768E-01**), chegando a esse resultado treinada por 91 épocas, com um total 6 camadas ocultadas, configurado com: 1 camada convolucional; 1 camada *Max Pooling*; 2 camadas GRU com 32 células cada; 2 camadas Dense com 32 neurônios cada; função de ativação *LeakyReLU* com α 0.1; 32 sequências temporais;

32 amostras por *batch*; e 10% de *Dropout*.

5.7 ESCOLHA DA MELHOR CONFIGURAÇÃO DE REDE NEURAL HÍBRIDA

Após o treinamento e testes das redes híbridas, cada configuração de mesma topologia foi analisada e comparada com as outras redes de mesma topologia, de modo a se definir a melhor configuração, para cada um dos dois tipos de RNAs híbridas.

Desta forma, as configurações híbridas com camadas LSTM, foram comparadas umas com as outras, da mesma forma, as híbridas com camadas GRU também foram comparadas umas com as outras. Desse modo, analisando as métricas de erro e os gráficos de comparação entre nível calculado e observado, foi definida a configuração híbrida com camadas LSTM que produziu os resultados mais satisfatórios e a configuração com camadas GRU que exibiu o melhor desempenho.

A melhor configuração Híbrida que produziu os resultados com os menores erros, foi a alternativa com 8 camadas ocultas, constituídas de 1 camada convolucional; 1 camada *Max Pooling*; 3 camadas LSTM com 128 células cada; 3 camadas *Dense* com 128 neurônios cada; função de ativação *Tanh*; 128 sequências temporais; 128 amostras por *batch*; e 10% de *Dropout*. Essa rede apresentou um RMSE, MAPE e NSE de **3.702E-03**, **1.434%**, **9.990E-01**, respectivamente.

A rede Híbrida com camadas recorrentes GRU que mostrou o melhor desempenho, exibiu um MSE, MAPE e NSE de **3.418E-03**, **1.276%** e **9.991E-01** respectivamente. Tal modelo chegou a esse resultado com 9 camadas ocultas, especificamente configurado com: 2 camadas convolucionais; 2 camadas *Max Pooling*; 3 camadas GRU com 128 células cada; 2 camadas *Dense* com 128 neurônios cada; função de ativação *LeakyReLU* com α 0.3; 128 sequências temporais; 128 amostras por *batch*; e 10% de *Dropout*.

Na comparação entre os dois modelos híbridos que obtiveram os resultados mais satisfatórios, levando-se em consideração as métricas de erro utilizadas, a rede híbrida com camadas GRU foi ligeiramente superior a rede híbrida com camadas LSTM, ou seja, os resultados exibidos pelas três métricas foram superiores, porém, razoavelmente próximos uns dos outros. Tendo em vista a análise dos gráficos comparativos do nível do rio previsto e observado, o modelo com camadas LSTM a presentou maiores erros em relação ao modelo com camadas GRU.

Desta forma, levando-se em consideração a superioridade da rede Híbrida com ca-

madas GRU nas três métricas utilizadas e uma quantidade menor de erros exibidos através dos gráficos de comparação, optou-se por selecionar esse modelo, como a configuração Híbrida mais adequada para as previsões do nível do rio no local de estudo selecionado.

5.8 ESCOLHA DA MELHOR CONFIGURAÇÃO DE REDE NEURAL PROFUNDA

As RNAs profundas treinadas e testadas no decorrer desta pesquisa, com os objetivos de selecionar a configuração mais adequada para realizar previsões de inundações na área de estudo selecionada, e responder a pergunta de pesquisa inicialmente proposta totalizaram 224 configurações.

As vertentes de redes recorrentes LSTM e GRU foram inicialmente avaliadas e testadas individualmente de acordo com o nível de profundidade. Os modelos que exibiram os menores erros em cada conjunto de n camadas ocultas foram reagrupados e novamente comparados, desta vez sem levar a quantidade de camadas ocultas em consideração. Ao final, o melhor modelo LSTM e o melhor modelo GRU foram selecionados e em seguida comparados um com o outro para se selecionar o modelo recorrente mais adequado para o problema em questão.

A estratégia utilizada para avaliar as RNAs híbridas não levou em consideração o nível de profundidade da rede. Desta forma, todas as configurações das redes híbridas com camadas LSTM foram avaliadas e comparadas de maneira geral umas com as e apenas uma única configuração foi selecionada. Da mesma forma, todas as configurações híbridas com camadas recorrentes GRU também foram comparadas e uma única configuração foi selecionada. Ao final, o melhor modelo híbrido LSTM foi comparado com o melhor híbrido GRU e o mais adequado dentre eles foi selecionado como modelo híbrido mais adequado para o problema em questão.

A Tabela 27 a seguir apresenta as melhores configurações eleitas de redes neurais recorrentes e híbridas.

Tabela 27 – Resultados das melhores configurações.

Modelo	Epochs	RMSE	MAPE %	NSE
GRU_4C_64CE_A_tanh_128TS_128B_Drpt_0.01	210	**3.392E-03**	1.403	9.991E-01
9C_CNN_2C_MaxPool_2C_GRU_3C_128CE_DENSE_2C_128NE_A_leakyrelu_0.3_128TS_128B_Drpt_0.1	87	3.418E-03	**1.276**	9.991E-01

Fonte – Elaborado pelo autor

Na comparação entre as duas redes com resultados mais satisfatórios, levando-se em consideração as três métricas de erro utilizadas, a rede GRU obteve o menor RMSE, enquanto que a rede Híbrida obteve o menor MAPE. Coincidentemente ambas as redes tiveram o mesmo NSE, desta forma acarretando um empate dos modelos.

De acordo com a sistemática adotada neste trabalho, apenas uma configuração de RNA deve ser eleita como a mais adequada. Em vista do empate entre as duas melhores configurações, como forma também de avaliar, na Tabela 28 são apresentados os níveis calculados em períodos de enchente/inundação pelas redes em comparação com alguns valores observados.

Tabela 28 – Previsões realizadas pelas melhores configurações de GRU e Híbrida – GRU..

Data	**Nível Observado (cm)**	**Nível previsto pela rede GRU (cm)**	**Erro GRU (cm)**	**Nível previsto pela rede Híbrida (cm)**	**Erro Híbrida (cm)**
20/09/2013	697.0	696.15	0.85	696.66	0.34
23/09/2013	1029.0	1026.59	2.41	1028.19	0.81
13/01/2014	619.0	615.02	3.98	615.99	3.01
14/01/2014	603.0	602.23	0.77	600.71	2.29
03/09/2018	685.0	683.19	1.81	683.71	1.29
04/09/2018	717.0	715.03	1.97	715.87	1.13

Fonte – Elaborado pelo autor

De acordo com a Tabela 28, em 4 ocasiões a rede Híbrida se mostrou superior a rede GRU, apresentando um erro mais baixo. No entanto, em outras duas ocasiões a rede GRU foi melhor. No geral, o erro acometido por ambas as redes não foi tão significativo, pois a diferença entre os valores calculados e observados foram baixos, ou seja, não passaram dos 4 centímetros de diferença.

Ambas as redes apresentaram resultados satisfatórios, no entanto, de acordo com a sistemática empregada neste trabalho, apenas uma rede deve ser eleita como a mais adequada para realizar as previsões do rio Itajaí Açu no local de interesse. Desta forma em vista dos resultados obtidos através das métricas de erro, e os erros de previsão exibidos na Tabela 19, conclui-se que a rede neural Híbrida com um total 9 camadas ocultas, configurada com 2 camadas convolucionais; 2 camadas *Max Pooling*; 2 camadas GRU com 128 células cada; 2 camadas *Dense* com 128 neurônios cada; função de ativação *LeakyReLU* com α 0.3; 128 sequências temporais; 128 amostras por *batch*; e 10% de *Dropout*; deve ser eleita a mais adequada para o problema em questão.

Os resultados apresentados por ambas as redes confirmam a hipótese desse traba-

lho, mostrando que as redes com maior profundidade fornecem um horizonte de previsão mais preciso em relação as redes mais rasas, e ademais, mostram também que é possível realizar a previsão de inundações em curto prazo com baixo erro no local de estudo selecionado.

Um ponto importante a se destacar, está no fato de que apenas aumentar a profundidade da rede, não fornece diretamente uma maior representatividade dos dados em relação a redes rasas. Um dos fatores que permitiram as RNAs com maior profundidade se destacarem neste trabalho, foi encontrar os parâmetros que mais se ajustaram ao modelo nas redes rasas e utilizá-los nas redes com maior profundidade.

Por último, destaca-se que de acordo com os testes realizados, não se evidenciou nenhuma tendência nos valores previstos, tanto para os modelos recorrentes, quanto para os modelos híbridos, serem acima ou abaixo dos valores observados na estação de Rio do Sul, ponto de interesse das previsões. Em alguns testes, ambos os resultados apresentados pelos modelos de ambas topologias, previram valores acima e em outros previram abaixo dos valores reais, desta forma não caracterizando nenhum viés neste sentido.

6 CONCLUSÃO

O objetivo inicial deste trabalho foi realizar a previsão de inundações em curto prazo em um ponto do rio Itajaí Açu. Especificamente, a área de estudo selecionada foi o município de Rio do Sul, devido ao histórico de enchentes e inundações que o acomete. Para realizar as previsões, decidiu-se empregar as Redes Neurais profundas, pois constituem o estado da arte atualmente.

Para que os objetivos do trabalho fossem alcançados, inicialmente realizou-se uma revisão bibliográfica a fim de identificar os principais modelos profundos que têm sido utilizados em previsões de inundações e as principais variáveis utilizadas. A partir disso, pôde-se evidenciar por viés da literatura, que o uso de Redes Neurais profundas têm sido infrequente no tange a previsão de inundações, principalmente na área de estudo selecionada.

Tal observação forneceu um arcabouço que deu origem à pergunta de pesquisa inicialmente proposta neste trabalho, gerando assim o segundo objetivo desta pesquisa, cuja consiste em verificar se a abordagem do aprendizado profundo pode fornecer um horizonte de previsão melhor em relação a abordagem tradicional.

Desta forma, para se atingir ambos os objetivos de realizar a previsão de inundações na área estudo selecionada, e, responder a pergunta de pesquisa em questão, uma vez que as principais variáveis foram identificadas, verificou-se a disponibilidade dos dados do rio Itajaí Açu coletados pela Agência Nacional das Águas. Assim, foi possível obter-se os dados de precipitação, nível do rio e vazão do ponto de interesse – Rio do Sul – e de dois pontos a montante local de estudo selecionado – Ituporanga e Taió.

A escolha dos pontos a montante possibilitaram desenvolver um modelo para previsões com 6 horas de antecedência, pois conforme explicado na literatura, um fator determinante para se obter tal antecedência situa-se no tempo de deslocamento da onda de água na rede de drenagem, que entre Ituporanga e Rio do Sul é de 6 horas; e de Taió a Rio do Sul é de 8 horas.

Por conseguinte, após a obtenção dos dados, antes de realizar a modelagem e treinamento das RNAs profundas, verificou-se a necessidade de se realizar as ações de tratamento e pré-processamento nos dados. Após essas etapas, com as amostras organizadas em um único arquivo no formato csv, e com o acoplamento temporal adequado entre as variáveis de entrada e saída, deu-se inicio a fase de treinamento dos modelos profundos.

Um total de 224 configurações de redes neurais profundas foram treinadas, valida-

das e testadas. Desse total, 47 foram modelos da topologia LSTM e 41 modelos da topologia GRU – vertentes das RNNs. O restante dos modelos eram híbridos, ou seja, uma junção de redes neurais convolucionais com redes recorrentes e redes MLP. Desta forma, os modelos híbridos foram distinguidos quanto ao tipo de rede recorrente empregada nas camadas ocultas, nesse sentido, 63 configurações de redes híbridas com LSTM nas camadas ocultas e 73 configurações com redes GRU foram construídas.

As redes recorrentes foram avaliadas de acordo com o nível de profundidade de cada configuração, e, os modelos que exibiram o melhor desempenho em cada grupo de n camadas, foram comparados entre si, de modo a verificar o desempenho das redes mais profundas em relação a redes mais rasas, com o objetivo de responder pergunta de pesquisa. Por outro lado, nos modelos híbridos não se levou em consideração o nível de profundidade pois estes já são inerentemente profundos, servindo apenas como modelos comparativos para com as vertentes recorrentes empregadas.

Para avaliar o desempenho dos modelos configurados, foram utilizadas as métricas estatísticas de erro RMSE, MAPE e NSE. Por meio dos resultados apresentados por estas métricas e através de gráficos de comparação, foi possível avaliar e selecionar as redes recorrentes e híbridas com os melhores desempenhos. As redes de cada topologia selecionadas foram comparadas em entre si para a escolha da rede mais adequada para realizar as previsões do nível do rio para a área de estudo selecionada.

Dentro desse contexto, o melhor desempenho dentre todas as configurações de redes neurais recorrentes treinadas foi obtido pela rede GRU com 4 camadas ocultas, 64 células em cada camada, função de ativação *Tanh*, 128 sequências temporais, 128 amostras por *batch* e 1% de *Dropout*. Esta rede produziu as previsões de melhor qualidade dentre todas as RNNs LSTM e GRU, exibindo RMSE, MAPE e NSE de 3.392E-03, 1.403% e 9.991E-01 respectivamente.

Em relação as redes híbridas, o modelo que exibiu o melhor desempenho foi a rede de 9 camadas ocultas, configurada com 2 camadas convolucionais, 2 camadas *Max Pooling*, 3 camadas GRU com 128 células cada, 2 camadas Dense com 128 neurônios cada, função de ativação *LeakyReLU* com α 0.3, 28 sequências temporais, 128 amostras por *batch* e 10% de *Dropout*. Essa configuração foi superior a todas as outras de mesma topologia segundo as métricas utilizadas ao exibir RMSE, MAPE e NSE de 3.418E-03, 1.276 e 9.991E-01 respectivamente.

Na comparação entre a melhor configuração recorrente e a melhor híbrida, levando-se em consideração as métricas de erro, ambas as redes chegaram a um empate. Com o objetivo

de selecionar apenas um modelo, decidiu-se realizar previsões para períodos de cheias e de inundações e comparar o erro acometido por ambas as redes. Ao se comparar o erro exibido por ambos os modelos, chegou-se a conclusão de que o modelo Híbrido com camadas recorrentes da topologia GRU realizou as previsões com maior qualidade e portanto é a rede mais aderente a área de estudo.

Através dos resultados obtidos, pode-se evidenciar que as redes neurais com um maior nível de profundidade forneceram um horizonte de previsão mais adequado em relação as redes rasas. Ao se encontrar os parâmetros mais adequados durante o treinamento das redes mais rasas, pode-se usar deste conhecimento favoravelmente para modelar e treinar os modelos mais profundos que exibiram um baixo percentual de erro de acordo com o observado nas métricas adotadas.

Os resultados satisfatórios obtidos através das redes profundas, não só permitiram responder a pergunta de pesquisa inicialmente proposta de forma positiva, mas também possibilitaram construir um modelo eficiente para se realizar a previsão de inundações, especificamente para a área de estudo seleciona, podendo servir como base para o desenvolvimento imediato de sistemas de alerta de enchentes e inundações para a cidade de Rio do Sul.

Deve-se salientar que, embora os resultados tenham sido positivos, as redes neurais profundas são muito dispendiosas computacionalmente, e por exigirem muitos recursos de processamento, o treinamento deste tipo de rede para grandes conjuntos de dados leva a necessidade de acesso a equipamentos mais robustos que nem sempre estão disponíveis para o pesquisador.

Dessa forma, para a realização de trabalhos futuros, propõe-se realizar novas investigações empregando-se outros modelos de aprendizado de máquina e através de comparativos averiguar se existem alternativas as redes profundas que sejam mais viáveis em termos de custo computacional. Também, desde que se disponha de recursos computacionais sem limite de uso e/ou tempo, poder-se-á realizar uma quantidade maior de testes, o que permitir-se-á ter-se uma confiabilidade estatística maior nos resultados, além disso, também dever-se-á se utilizar de uma quantidade maior de métricas estatísticas para a avaliação dos modelos de modo a se permitir uma interpretação mais ampla e mais confiável dos resultados em questão.

REFERÊNCIAS

ANA. **Agência Nacional de Águas**. 2019. Acesso em: 27 de Setembro de 2019. Disponível em: <https://www.ana.gov.br/>.

AREL, Itamar; ROSE, Derek C.; KARNOWSKI, Thomas P. Deep Machine Learning - A New Frontier in Artificial Intelligence Research [Research Frontier]. **IEEE Comp. Int. Mag.**, v. 5, n. 4, p. 13–18, 2010. Disponível em: <http://dblp.uni-trier.de/db/journals/cim/cim5.html#ArelRK10>.

BARRETO, Jorge M. **Indrodução às Redes Neurais Artificiais**. [S.l.]: Laboratório de Conexionismo e Ciências Cognitivas UFSC, 2002. <http://www.inf.ufsc.br/~j.barreto/tutoriais/Survey.pdf>.

BATTITI, Roberto; BRUNATO, Mauro. **The LION way. Machine Learning *plus* Intelligent Optimization**. LIONlab, University of Trento, Italy, 2017. Disponível em: <http://intelligent-optimization.org/LIONbook/>.

BENGIO, Yoshua. Learning Deep Architectures for AI. **Foundations**, v. 2, p. 1–55, Janeiro 2009.

BISHOP, Christopher M. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0-387-31073-8.

BONAFÉ, Maria. **URBANIZAÇÃO E CONTROLE DE ENCHENTES O CASO DE SÃO PAULO: SEUS CONFLITOS E INTER-RELAÇÓES**. Dissertação (Mestrado) — UNIVERSIDADE DE SÃO PAULO, SÃO PAULO (SP), Julho 1989. Resumo da Dissertação apresentada à Escola Politécnica da Universidade de São Paulo no Departamento de Engenharia Urbana e Construção Civil – São Paulo (SP).

BRAGA, Antônio de Pádua; CARVALHO, André Carlos Ponce de Leon Ferreira de; LUDERMIR, Teresa Bernarda. **REDES NEURAIS ARTIFICIAIS: TEORIA E APLICAÇÕES**. 2.ed. ed. Rio de Janeiro - RJ: LTC, 2011. ISBN 978-85-216-1564-4.

BRILLIANT. **Backpropagation**. 2019. Acessado em: 02 de Outubro de 2019. Disponível em: <https://brilliant.org/wiki/backpropagation/>.

BROOKSHEAR, J. Glenn. **Computer Science: An Overview**. 10th. ed. USA: Addison-Wesley Publishing Company, 2008. ISBN 9780321524034.

BROWNLEE, Jason. **How To Backtest Machine Learning Models for Time Series Forecasting**. 2016. Acesso em: 27 de Setembro de 2019. Disponível em: <https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/>.

____. **Gentle Introduction to the Adam Optimization Algorithm for Deep Learning**. 2017. Acesso em: 11 de Outubro de 2019. Disponível em: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>.

CASTRO, Antônio Luiz Coimbra de. **Manual de Desastres - Desastres Naturais**. Brasília - DF: Ministério sa Integração Nacional, 2003. v. 1.

CHO, Kyunghyun et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. **CoRR**, abs/1406.1078, 2014. Disponível em: <http://arxiv.org/abs/1406.1078>.

CHOLLET, FRANÇOIS. **Deep Learning with Python**. [S.l.]: Manning Publications Co., 2018. ISBN 9781617294433.

CHUNG, Junyoung et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. **CoRR**, abs/1412.3555, 2014. Disponível em: <http://arxiv.org/abs/1412.3555>.

CORDERO, Ademar; MOMO, Marcos Rodrigo; SEVERO, Dirceu Luis. PREVISÃO DE CHEIA EM TEMPO ATUAL, COM UM MODELO ARMAX, PARA A CIDADE DE RIO DO SUL-SC. In: **XIX Simpósio Brasileiro de Recursos Hídricos**. [S.l.: s.n.], 2011.

DANCUCO. **RIO DO SUL ENCHENTE Setembro 2011**. 2011. Acesso em: 26 de Maio de 2019. Disponível em: <https://dancuco.blogspot.com/2011/09/rio-do-sul-enchente-2011_12.html>.

DATA SCIENCE ACADEMY. **Deep Learning Book**. 2019. Acessado em: 02 de Julho de 2019. Disponível em: <http://www.deeplearningbook.com.br>.

DEFESA CIVIL. **Quadro de cheias ocorridas em Rio do Sul**. 2019. Acessado em: 01 de Maio de 2019. Disponível em: <https://defesacivil.riodosul.sc.gov.br/index.php?r=externo%2Fplanilha>.

DENG, Li; YU, Dong. **Deep Learning: Methods and Applications**. Hanover, MA, USA: Now Publishers Inc., 2014. ISBN 1601988141, 9781601988140.

DESHPANDE, Mohit. **Introduction to Convolutional Neural Networks for Vision Tasks**. 2017. Acessado em: 06 de Outubro de 2019. Disponível em: <https://pythonmachinelearning.pro/introduction-to-convolutional-neural-networks-for-vision-tasks/>.

DICIO - DICIONÁRIO ONLINE DE PORTUGUÊS. **morfometria**. 2019. Acessado em: 02 de Julho de 2019. Disponível em: <https://www.dicio.com.br/morfometria/>.

ELSAFI, Sulafa Hag. Artificial Neural Networks (ANNs) for flood forecasting at Dongola Station in the River Nile, Sudan. **Alexandria Engineering Journal**, v. 53, n. 3, p. 655 – 662, 2014. ISSN 1110-0168. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1110016814000660>.

ESPÍNDOLA, Marcos Aurélio; NODARI, Eunice Sueli. Enchentes inesperadas? vulnerabilidades e políticas públicas em Rio do Sul - SC, Brasil. In: **Esboços: histórias em contextos globais**. [S.l.: s.n.], 2013. v. 20, n. 30, p. 9–34. ISSN 2175-7976.

FEW, R et al. **Floods, health and climate change: a strategic review**. [S.l.], 2004. Disponível em: <http://researchonline.lshtm.ac.uk/13333/>.

FINOCCHIO, Marco Antonio Ferreira. **NOÇÕES DE REDES NEURAIS ARTIFICIAIS**. [S.l.]: UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ, 2014.

FISCHER, Thomas; KRAUSS, Christopher. **Deep learning with long short-term memory networks for financial market predictions**. [S.l.], 2017. Disponível em: <https://EconPapers.repec.org/RePEc:zbw:iwqwdp:112017>.

GAMBOA, John Cristian Borges. Deep Learning for Time-Series Analysis. **CoRR**, abs/1701.01887, 2017. Disponível em: <http://arxiv.org/abs/1701.01887>.

GARCIA, Sara Iris. **An introduction to Gradient Descent Algorithm**. 2019. Acessado em: 02 de Outubro de 2019. Disponível em: <https://medium.com/@montjoile/an-introduction-to-gradient-descent-algorithm-34cf3cee752b/>.

GAUTAM, K.P.; HOEK, E.E. van der. Literature study on environmental impact of floods. **DC1-233-13**, Delft Cluster, Junho 2003. Disponível em: <http://resolver.tudelft.nl/uuid:4080519e-a46d-4e96-8524-62ee8fd93712>.

GÉRON, Aurélien. **Hands-On Machine Learning with Scikit-Learn and TensorFlow**: Concepts, Tools, and Techniques to Build Intelligent Systems. 1st. ed. USA: O'Reilly Media, Inc, 2017. ISBN 978-1-491-96229-9.

GLOROT, Xavier; BENGIO, Yoshua. Understanding the difficulty of training deep feedforward neural networks. **Journal of Machine Learning Research - Proceedings Track**, v. 9, p. 249–256, 01 2010.

GOERL, Roberto Fabris; KOBIYAMA, Masato. **Considerações sobre as inundações no Brasil**. 2005. Disponível em: <http://www.labhidro.ufsc.br/Artigos/ABRH2005_inundacoes.pdf>.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GOOGLE COLAB. **Google Colaboratory.** 2019. Acesso em: 10 de Outubro de 2019. Disponível em: <https://colab.research.google.com/notebooks/welcome.ipynb/>.

HADDAD, Eduardo A.; TEIXEIRA, Eliane. **ECONOMIC IMPACTS OF NATURAL DISASTERS IN MEGACITIES: THE CASE OF FLOODS IN SÃO PAULO, BRAZIL**. São Paulo, SP: NEREUS - Núcleo de Economia Regional e Urbana da Universidade de São Paulo, 2013.

HAGAN, Martin T. et al. **Neural Network Design**. 2nd. ed. USA: Martin Hagan, 2014. ISBN 0971732116, 9780971732117.

HAYKIN, Simon S. **Neural networks and learning machines**. Third. Upper Saddle River, NJ: Pearson Education, 2009.

HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HU, Caihong et al. Deep Learning with a Long Short-Term Memory Networks Approach for Rainfall-Runoff Simulation. **Water**, v. 10, p. 1543, Outubro 2018.

IBGE. **Cidades**. 2019. Acessado em: 01 de Maio de 2019. Disponível em: <https://cidades.ibge.gov.br/brasil/sc/rio-do-sul/panorama>.

JOZEFOWICZ, Rafal; ZAREMBA, Wojciech; SUTSKEVER, Ilya. An Empirical Exploration of Recurrent Network Architectures. In: **Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37**. JMLR.org, 2015. (ICML'15), p. 2342–2350. Disponível em: <http://dl.acm.org/citation.cfm?id=3045118.3045367>.

JUPYTER. **Jupyter**. 2019. Acessado em: 03 de Dezembro de 2019. Disponível em: <https://jupyter.org/>.

KARN, Ujjwal. **An Intuitive Explanation of Convolutional Neural Networks**. 2016. Acessado em: 06 de Outubro de 2019. Disponível em: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>.

KERAS. **Keras: The Python Deep Learning library.** 2019. Acesso em: 10 de Outubro de 2019. Disponível em: <https://keras.io/>.

KINGMA, Diederik; BA, Jimmy. Adam: A Method for Stochastic Optimization. **International Conference on Learning Representations**, 12 2014.

KOSTADINOV, Simeon. **Understanding GRU networks**. 2017. Acessado em: 05 de Outubro de 2019. Disponível em: <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>.

KOVÁCS, Zsolt László. **Redes Neurais Artificiais**. 4 edição. ed. São Paulo - SP: Editora Livraria da Física, 2006. ISBN 85-88325-14-4.

KRIESEL, David. **A Brief Introduction to Neural Networks** . [s.n.], 2007. Disponível em: <http://www.dkriesel.com>.

KRON, Wolfgang. Flood risk = hazard x values x vulnerability. **Flood Defence**, Science Press, New York Ltd, Janeiro 2002.

LÄNGKVIST, Martin; KARLSSON, Lars; LOUTFI, Amy. A Review of Unsupervised Feature Learning and Deep Learning for Time-Series Modeling. **Pattern Recognition Letters**, v. 42, Junho 2014.

LAURENTINO, Gabriel Schmitt. **MEDIDAS DE CONTROLE DE ENCHENTES E INUNDAÇÕES NA CIDADE DE RIO DO SUL/SC**. 2016.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep Learning. **Nature**, v. 521, n. 7553, p. 436–444, Maio 2015. Disponível em: <https://www.nature.com/articles/nature14539>.

LECUN, Yann et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, Nov 1998.

LIU, Fan; XU, Feng; YANG, Sai. A Flood Forecasting Model Based on Deep Learning Algorithm via Integrating Stacked Autoencoders with BP Neural Network. In: **2017 IEEE Third International Conference on Multimedia Big Data (BigMM)**. [S.l.: s.n.], 2017. p. 58–61.

LOHMANN, Marciel. **REGRESSÃO LOGÍSTICA E REDES NEURAIS APLICADAS À PREVISÃO PROBABILÍSTICA DE ALAGAMENTOS NO MUNICÍPIO DE CURITIBA, PR**. Tese (Doutorado) — Universidade Federal do Paraná, Curitiba (PR), 2011. PROGRAMA DE PÓS - GRADUAÇÃO EM GEOGRAFIA – Curitiba (PR).

MATPLOTLIB. **Matplotlib.** 2019. Acesso em: 10 de Outubro de 2019. Disponível em: <https://matplotlib.org/>.

MEHLIG, B. Artificial Neural Networks. **CoRR**, abs/1901.05639, 2019. Disponível em: <http://arxiv.org/abs/1901.05639>.

MENDIONDO, Eduardo Mario. Flood risk management of urban waters in humid tropics: early warning, protection and rehabilitation. In: **Workshop on Integrated Urban Water Management in Humid Tropics**. [S.l.: s.n.], 2005.

MOSAVI, Amir; OZTURK, Pinar; CHAU, Kwok-wing. Flood Prediction Using Machine Learning Models: Literature Review. **Water**, v. 10, p. 1536, Outubro 2018.

NASH, J.E.; SUTCLIFFE, J.V. River flow forecasting through conceptual models part I — A discussion of principles. **Journal of Hydrology**, v. 10, n. 3, p. 282 – 290, 1970. ISSN 0022-1694. Disponível em: <http://www.sciencedirect.com/science/article/pii/0022169470902556>.

NIELSEN, Michael A. **Neural Networks and Deep Learning**. 2019. Disponível em: <http://neuralnetworksanddeeplearning.com/>.

NUMPY. **Numpy.** 2019. Acesso em: 09 de Outubro de 2019. Disponível em: <https://numpy.org/>.

OLAH, Christopher. **Understanding LSTM Networks**. 2015. Acessado em: 03 de Outubro de 2019. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

OTTO, Saskia A. **How to normalize the RMSE**. 2019. Acesso em: 12 de Outubro de 2019. Disponível em: <https://www.marinedatascience.co/blog/2019/01/07/normalizing-the-rmse/>.

PANDAS. **The pandas project.** 2019. Acesso em: 09 de Outubro de 2019. Disponível em: <https://pandas.pydata.org/about.html/>.

PATTERSON, Josh; GIBSON, Adam. **Deep Learning: A Practitioner's Approach**. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2017. ISBN 1491914254, 9781491914250.

PETNEHÁZI, Gábor. Recurrent Neural Networks for Time Series Forecasting. **CoRR**, abs/1901.00069, 2019. Disponível em: <http://arxiv.org/abs/1901.00069>.

PINZÓN, Tito Morales; RESTREPO, Juan David Céspedes; CALDERÓN, Manuel Tiberio Flórez. Daily river level forecast based on the development of an artificial neural network: case study in La Virginia -Risaralda. **Revista Facultad de Ingenierá Universidad de Antioquia**, scieloco, p. 46 – 57, Setembro 2015. ISSN 0120-6230. Disponível em: <http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-62302015000300006&nrm=iso>.

RASCHKA, Sebastian; MIRJALILI, Vahid. **Python Machine Learning**. Second edition. [S.l.]: Packt Publishing, 2017. ISBN 978-1-78712-593-3.

ROJAS, Raúl. **Neural Networks: A Systematic Introduction**. Berlin, Heidelberg: Springer-Verlag, 1996. ISBN 3-540-60505-3.

SCIKIT-LEARN. **Machine Learning in Python.** 2019. Acesso em: 09 de Outubro de 2019. Disponível em: <https://scikit-learn.org/>.

SHAH, Tarang. **About Train, Validation and Test Sets in Machine Learning.** 2017. Acessado em: 30 de Setembro de 2019. Disponível em: <https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7/>.

SHEN, Chaopeng. A trans-disciplinary review of deep learning research for water resources scientists. **Water Resources Research**, Dezembro 2017.

SILVA, Ivan Nunes da; SPATTI, Danilo Hernane; FLAUZINO, Rogério Andrade. **Redes neurais artificiais para engenharia e ciências aplicadas**. [S.l.]: Artliber Editora, 2010.

SILVA, Pedro Reis da; SOUZA, Fabiano de. INUNDAÇÕES NO MUNICÍPIO DE RIO DO SUL: UMA ANÁLISE DOS EVENTOS DE 2011 E 2013 À LUZ DA GESTÃO DE RISCO DE DESASTRES. **Revista Ordem Pública e Defesa Social**, v. 9, n. 1, p. 163–179, Janeiro 2016. ISSN 2237-6380. Disponível em: <http://www.acors.org.br/rop.emnuvens.com.br/rop>.

SIT, Muhammed; DEMIR, Ibrahim. Decentralized Flood Forecasting Using Deep Neural Networks. **CoRR**, abs/1902.02308, 2019. Disponível em: <http://arxiv.org/abs/1902.02308>.

SOARES, Daniel Gomes. **PREVISÃO DE CHEIAS DO RIO ITAJAÍ-AÇU UTILIZANDO REDES NEURAIS ARTIFICIAIS**. Dissertação (Mestrado) — UNIVERSIDADE DO VALE DO ITAJAÍ, ITAJAÍ (SC), Outubro 2014. Curso de mestrado acadêmico em computação aplicada – Itajaí (SC).

SOARES, Daniel Gomes; TEIVE, Raimundo Celeste Ghizoni. Estudo Comparativo entre as Redes Neurais Artificiais MLP e RBF para Previsão de Cheias em Curto Prazo. In: **Rita - Revista de Informática Teórica e Aplicada**. [s.n.], 2015. v. 22, n. 2, p. 67–86. ISSN 21752745. Disponível em: <https://seer.ufrgs.br/rita/article/view/RITA-VOL22-NR2-67>.

STEPHANIE. **What is Root Mean Square Error (RMSE)?** 2016. Acesso em: 12 de Outubro de 2019. Disponível em: <https://www.statisticshowto.datasciencecentral.com/rmse/>.

____. **Mean absolute percentage error (MAPE).** 2017. Acesso em: 12 de Outubro de 2019. Disponível em: <https://www.statisticshowto.datasciencecentral.com/mean-absolute-percentage-error-mape/>.

Mape (mean absolute percentage error)mean absolute percentage error (mape). In: SWAMIDASS, Paul M. (Ed.). **Encyclopedia of Production and Manufacturing Management**. Boston, MA: Springer US, 2000. p. 462–462. ISBN 978-1-4020-0612-8. Disponível em: <https://doi.org/10.1007/1-4020-0612-8_580>.

TUCCI, Carlos E. M. **Gestão da drenagem urbana**. Brasília, DF: Instituto de Pesquisa Econômica Aplicada – IPEA, 2012. v. 48. ISSN 2179-5495.

TUCCI, Carlos E. M.; BERTONI, Juan Carlos. **Inundações urbanas na América do Sul**. Porto Alegre, RS: Ed. dos Autores, 2003. ISBN 85-88686-07-4.

VALDENEGRO-TORO, Matias. Deep Neural Networks for Marine Debris Detection in Sonar Images. **CoRR**, abs/1905.05241, 2019. Disponível em: <http://arxiv.org/abs/1905.05241>.

VAROONCHOTIKUL, Pichaid. Flood forecasting using artificial neural networks. In: . [S.l.: s.n.], 2003.

WANG, Jianjin et al. Application of BP Neural Network Algorithm in Traditional Hydrological Model for Flood Forecasting. **Water**, v. 9, p. 48, Janeiro 2017.

ZHANG, Aston et al. **Dive into Deep Learning**. [s.n.], 2019. Disponível em: <http://www.d2l.ai>.