UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA

GUSTAVO RYUJI TAIRA

Redes Neurais Bayesianas para Calibração de Sensores de Poluição do Ar e Detecção de Falhas em Processos Químicos

São Paulo
2022

GUSTAVO RYUJI TAIRA

Redes Neurais Bayesianas para Calibração de Sensores de Poluição do Ar e Detecção de Falhas em Processos Químicos

Versão Original

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

São Paulo
2022

GUSTAVO RYUJI TAIRA

Redes Neurais Bayesianas para Calibração de Sensores de Poluição do Ar e Detecção de Falhas em Processos Químicos

Versão Original

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Área de Concentração:
Engenharia Química

Orientador:
Prof. Dr. Song Won Park

São Paulo
2022

Catalogação-na-publicação

Taira, Gustavo Ryuji
Redes Neurais Bayesianas para Calibração de Sensores de Poluição do Ar e Detecção de Falhas em Processos Químicos / G. R. Taira -- São Paulo, 2022.
132 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Química.

1.Aprendizado de Máquina 2.Rede Neural Bayesiana 3.Calibração de Sensor 4.Detecção de Falhas em Processo Químico I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Química II.t.

Dedico esse trabalho a minha família, em especial ao meu querido avô Shigeru Taira.

AGRADECIMENTOS

Agradeço ao professor Dr. Song Won Park pela orientação e ensinamentos.

Agradeço à minha família, em especial aos meus pais, pelo apoio, suporte e principalmente pela paciência.

Agradeço aos meus colegas do bloco 21 pelo companheirismo, conversas, reuniões e cafés!

Agradeço a todos os meus amigos que me apoiaram nessa fase da minha vida.

Por fim, agradeço à FAPESP pelo apoio financeiro fornecido durante o desenvolvimento deste trabalho — Processo 2019/08280-9, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

"As opiniões, hipóteses e conclusões ou recomendações expressas neste material são de responsabilidade do(s) autor(es) e não necessariamente refletem a visão da FAPESP".

RESUMO

O desenvolvimento de técnicas de aprendizado de máquina é considerado uma das áreas de estudo que vem revolucionando a sociedade nas últimas décadas. O aprendizado de máquina é uma subárea do desenvolvimento da inteligência artificial que visa capacitar computadores a realizar tarefas de forma autônoma por meio de processos de aprendizado baseados em análise de dados. Desse modo, o desenvolvimento das técnicas de aprendizado de máquina tem proporcionado grandes transformações na sociedade ao possibilitar que computadores aprendam a realizar tarefas e funções que até pouco tempo só eram possíveis de serem realizadas por seres humanos. Além disso, o uso de técnicas de aprendizado de máquina tem se tornado uma tendência em diversas áreas científicas pelo fato de possibilitar o treinamento por meio do uso de dados de modelos capazes de realizar predições para sistemas e fenômenos difíceis de serem modelados analiticamente. Entre os diversos tipos de modelos de aprendizado de máquina existentes, as redes neurais artificiais são modelos que recentemente voltaram a ganhar grande destaque na comunidade científica. As redes neurais artificiais são os modelos de aprendizado de máquina reconhecidos por sua capacidade de extrair padrões abstratos e profundos presentes em conjunto de dados, e que recentemente, sob o contexto do desenvolvimento das técnicas de aprendizado profundo (*deep learning*), têm ganhado notoriedade pelo seu sucesso na solução de problemas complexos de diversas áreas científicas. Apesar desse sucesso, os modelos de redes neurais artificiais ainda apresentam alguns problemas que necessitam ser solucionados. Críticas comumente realizadas às redes neurais artificiais são a tendência desses modelos sofrerem sobreajuste e a incapacidade de fornecerem suas predições junto de seus respectivos valores de incerteza. Esses problemas prejudicam a capacidade de generalização desses modelos e podem gerar problemas de segurança durante a utilização de suas predições. Nesse contexto, o desenvolvimento de modelos de rede neurais artificiais bayesianas tem se estabelecido como uma solução promissora para esses problemas. Modelos de rede neurais artificiais bayesianas são modelos de redes neurais estocásticos que possuem mecanismos naturais de regularização dos modelos que evitam o problema de sobreajuste e que são capazes de diretamente representar valores de incerteza de suas predições. Estudos sobre aplicações de redes neurais bayesianas para solução de problemas reais, no entanto, ainda são bem escassos na literatura comparado ao número de estudos sobre aplicações de redes neurais tradicionais. Sendo assim, esse trabalho tem como objetivo provar a aplicabilidade de modelos de redes neurais bayesianas para solução de problemas reais referentes a dois estudos de caso. O primeiro caso consiste no estudo da aplicação de um modelo de rede neural bayesiana para a solução de um problema de calibração de sensores de poluição do ar de baixo-custo. Já o segundo caso consiste no estudo da aplicação de um modelo de rede neural bayesiana para a solução de um problema de detecção de falhas em um processo químico real. Para cada estudo de caso, o desempenho da solução baseada em rede neural bayesiana é avaliado, assim como é realizada uma análise das vantagens e limitações do uso de rede neural bayesiana para solução do problema.

Palavras-Chave: Aprendizado de Máquina, Rede Neural Bayesiana, Calibração de Sensor, Detecção de Falhas em Processo Químico.

ABSTRACT

The development of machine learning techniques is considered one of the study areas that has revolutionized society in recent decades. Machine learning is a subarea of the artificial intelligence development area that aims to enable computers to perform tasks autonomously through learning processes based on data analysis. In this way, the development of machine learning techniques has provided great transformations in society by enabling computers to learn how to perform tasks and functions that until recently were only possible to be performed by humans. In addition, the use of machine learning techniques has become a trend in several scientific areas since it enables the training of data-driven models capable of making predictions for systems and phenomena that are difficult to model analytically. Among the various types of existing machine learning models, artificial neural networks are models that have recently gained great prominence in the scientific community. Artificial neural networks are machine learning models recognized for their ability to extract abstract and deep patterns present in data sets, and that recently, under the context of the development of deep learning techniques, have gained notoriety for their success in the solution of complex problems from different scientific areas. Despite this success, the models of artificial neural networks still have some problems that need to be solved. Commonly held criticisms of artificial neural networks are the tendency of these models to overfit and the inability to provide their predictions along with their respective uncertainty values. These problems impair the generalizability of these models and can generate security problems when using their predictions. In this context, the development of Bayesian artificial neural network models has established itself as a promising solution to these problems. Bayesian artificial neural network models are stochastic neural network models that have natural mechanisms of model regularization that avoid the problem of overfitting and that can directly represent the uncertainty values of their predictions. Studies on applications of Bayesian neural networks for solving real problems, however, are still very scarce in the literature compared to the number of studies on applications of traditional neural networks. Therefore, this work aims to prove the applicability of Bayesian neural network models to solve real problems addressed in two case studies. The first case is a study of the application of a Bayesian neural network model for the solution of a problem of calibration of low-cost air pollution sensors. The second case is a study of the application of a Bayesian neural network model for the solution of a problem of fault detection in a real chemical process. For each case study, the performance of the solution based on a Bayesian neural network is evaluated, as well as an analysis of the advantages and limitations of using a Bayesian neural network to solve the problem is performed.

Keywords – Machine Learning, Bayesian Neural Network, Sensor Calibration, Chemical Process Fault Detection.

LISTA DE FIGURAS

LISTA DE TABELAS

ABREVIATURAS

ADALINE	Adaptative Linear Neuron.
Bayes-LSTM	Bayesian Long Short-Term Memory.
BPTT	Back-propagation Through Time.
BRNN	Bayesian Recurrent Neural Network.
DBN	Deep Belief Network.
DPCA	Dynamic Principal Component Analysis.
EWMA	Exponentially-Weighted Moving Average.
FCC	Fluid Catalytic Cracking.
FDA	Fisher Discriminant Analysis.
FN	False Negative.
FP	False Positive.
GLP	Gás Liquefeito de Petróleo.
GPR	Gaussian Process Regression.
GPS	Global Positioning System.
GRU	Gated Recurrent Unit.
HMC	Hamiltonian Monte Carlo.
ICA	Independent Component Analysis.
KL	Kullback-Leibler.
KPCA	Kernel Principal Component Analysis.
LCO	Light Cycle Oil.
LED	Light-Emitting Diode.
LSTM	Long Short-Term Memory.
MAE	Mean Absolute Error.
MCMC	Markov Chain Monte Carlo.
MLP	Multilayer Perceptron.
MLR	Multiple Linear Regressor.
PCA	Principal Component Analysis.
PLS	Partial Least Squares.
ReLU	Rectified Linear Unit.
SGHMC	Stochastic Gradient Monte Carlo.
SVM	Support Vector Machine.

SWA	Stochastic Weight Averaging.
SWAG	Stochastic Weight Averaging - Gaussian.
TN	True Negative.
TNR	True Negative Rate.
TP	True Positive.
TPR	True Positive Rate.

SUMÁRIO

1 INTRODUÇÃO

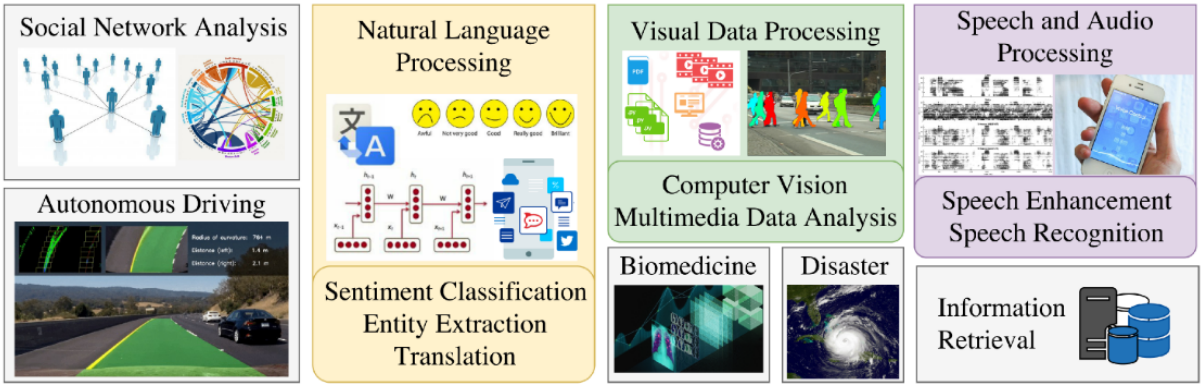
Tal como a Internet, que no final dos anos 90 e início dos anos 2000 transformou o modo como as pessoas vivem e trabalham, o aprendizado de máquina é considerado uma das atuais tecnologias que vem revolucionando a sociedade (LEE; SHIN; REALFF, 2018). O aprendizado de máquina, também conhecido como *machine learning*, consiste em uma subárea do estudo de inteligência artificial que busca desenvolver técnicas que capacitem os computadores a obter conhecimento a partir de experiências, tal como os seres humanos. No caso dos computadores, no entanto, suas experiências são descritas por meio de dados e a obtenção de conhecimento ocorre pela extração de padrões presentes neles. Desse modo, as técnicas de aprendizado de máquina consistem basicamente em algoritmos que visam capacitar os computadores a analisar dados de forma automatizada, extrair padrões dos dados e utilizá-los para a construção de modelos capazes de realizar predições para sistemas complexos (GOODFELLOW; BENGIO; COURVILLE, 2016).

Aplicações baseadas em técnicas de aprendizado de máquina já fazem parte do cotidiano da população, sendo utilizadas, por exemplo, na geração de resultados em sites de busca, na filtragem de conteúdo em redes sociais, em sistemas de recomendação de produtos de sites de *e-commerce*, em aplicações de tradução de idiomas, em sistemas de reconhecimento de fala, e em diversas outras aplicações comerciais (LECUN; BENGIO; HINTON, 2015). Além disso, aplicações de aprendizado de máquina tem recentemente ganhado grande notoriedade pelo fato de tornarem possível que computadores executem tarefas complexas que até pouco tempo só eram possíveis de serem realizadas por seres humanos, tal como dirigir automóveis, realizar o diagnóstico médicos, ou até mesmo derrotar jogadores humanos em jogos de tabuleiro tais como o xadrez e o *Go* (POUYANFAR et al., 2018).

Entre os diversos tipos de técnicas de aprendizado de máquina em desenvolvimento, as técnicas de aprendizado profundo, mais conhecidas como técnicas de *deep learning*, têm recentemente ganhado grande destaque na comunidade científica devido ao seu sucesso na solução de diversos problemas complexos da área de inteligência artificial (LECUN; BENGIO; HINTON, 2015). Tal como ilustrado na Figura 1, aplicações de técnicas de *deep learning* têm se estabelecido, por exemplo, como o estado da arte na solução de problemas como os de visão computacional (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; SZEGEDY et al., 2015), reconhecimento de fala (DAHL et al., 2011; HINTON et al., 2012), processamento de linguagem natural (BAHDANAU; CHO; BENGIO, 2014; CHO et al., 2014; SUTSKEVER; VINYALS; LE, 2014), e vêm contribuindo também

para o desenvolvimento inovações, tais como sistemas computacionais capazes de realizar diagnósticos médicos (KER et al., 2017) e carros autônomos (CHEN et al., 2015). Além disso, dado esse sucesso, aplicações de *deep learning* têm se tornado uma tendência em diversas áreas científicas, pelo fato de possibilitarem a construção de modelos capazes de realizar predições para sistemas e fenômenos complexos e difíceis de serem modelados analiticamente, como por exemplo na área de previsão de desastres naturais (HERNÁNDEZ et al., 2016), na análise de dados de aceleradores de partículas (CIODARO et al., 2012), na predição de efeitos de mutações na expressão genética (LEUNG et al., 2014), entre outras.

Figura 1: Exemplo de aplicações de *deep learning*



Fonte: Adaptado de Pouyanfar et al. (2018)

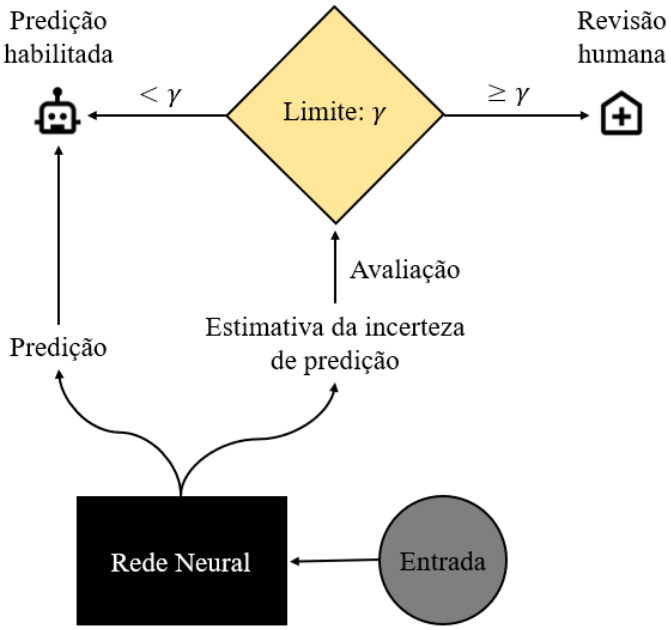
Os modelos base das técnicas de *deep learning* são as chamadas redes neurais artificiais, as quais são modelos matemáticos de aprendizado de máquina que foram criados com objetivo de imitar o processo de aprendizado do cérebro humano e que são reconhecidos por sua capacidade de extrair padrões abstratos e profundos presentes em dados referentes a sistemas complexos, e utilizá-los para realizar a predição do comportamento desses sistemas de forma precisa. As redes neurais artificiais são modelos de aprendizado de máquina utilizados na prática desde os anos 80, mas que somente recentemente alcançaram seu auge de performance na solução de problemas complexos, superando a performance de diversas outras técnicas de aprendizado de máquina. Até o início dos anos 2000, as redes neurais artificiais eram consideradas difíceis de serem treinadas que exigiam um longo período para alcançar desempenho satisfatórios. Esse panorama, no entanto, mudou drasticamente a partir da segunda metade dos anos 2000, devido principalmente ao surgimento de fatores do atual contexto tecnológico, tais como a maior disponibilidade de dados (*big data*), a maior capacidade de processamento dos atuais recursos computacionais e o desenvolvimento de técnicas de treinamento mais avançadas, os quais têm possibilitado que modelos de redes neurais mais complexos e profundos sejam facilmente

treinados e capacitados para solução de problemas de grande complexidade com elevada precisão (GOODFELLOW; BENGIO; COURVILLE, 2016).

Apesar do recente sucesso alcançado pelas redes neurais artificiais na solução de problemas de grande complexidade, esses modelos de aprendizado de máquina ainda apresentam alguns problemas que necessitam ser solucionados. Uma crítica comumente realizada às redes neurais artificiais é a tendência desses modelos sofrerem sobreajuste. O problema de sobreajuste, mais conhecido pelo seu nome em inglês *overfitting*, consiste um problema frequentemente enfrentado no processo de treinamento de redes neurais, no qual o modelo acaba memorizando os dados de treinamento. Esse problema gera grandes danos à capacidade de generalização do modelo, ou seja, ao analisar qualquer dado que não esteja presente no conjunto de dados de treinamento, a predição gerada a partir da análise desse dado acaba não sendo precisa, afetando assim a aplicabilidade do modelo do treinado (JOSPIN et al., 2022).

Além disso, uma outra crítica feita aos modelos de redes neurais é que eles tendem a realizar predições com excesso de confiança, ou seja, suas predições são fornecidas sem intervalos de confiança, impossibilitando quantificar sua confiança. A realização de predições com excesso de confiança por modelos pode gerar problemas de segurança na solução de problemas que envolvem tomadas de decisão críticas, tais como diagnóstico médicos e controle de veículos autônomos, pois predições pouco confiáveis e errôneas podem acabar sendo utilizadas e causar graves consequências indesejáveis, como por exemplo o fornecimento de diagnósticos errados para pacientes e a causa de acidentes de trânsito (GAL, 2016; KENDALL; GAL, 2017). Uma excelente solução para esse problema seria o fornecimento das predições do modelo junto aos seus valores de incerteza, os quais são informações que permitem avaliar a confiança e segurança das predições. Tal como ilustrado na Figura 2, o ideal é que o valor de incerteza de toda predição fornecida por um modelo de rede neural seja avaliado e que a utilização da predição seja habilitada apenas quando esse valor esteja abaixo de um valor limite aceitável. Caso o valor de incerteza ultrapasse o valor limite aceitável, o ideal é que a predição do modelo seja encaminhada para revisão de um ser humano, o qual deve então autorizar ou negar a sua utilização. Entretanto, no modo no qual as tradicionais modelos de redes neurais artificiais são estruturadas e treinadas, os valores de incerteza de suas predições não podem ser fornecidos de forma direta, exigindo mecanismos extras para seu cálculo (GAL, 2016).

Figura 2: Avaliação do valor de incerteza de uma predição



Fonte: Adaptado de Band et al. (2021)

Nesse contexto, o desenvolvimento de modelos de rede neurais artificiais bayesianas tem se estabelecido como solução promissora para esses problemas das redes neurais artificiais tradicionais. As redes neurais bayesianas consistem em modelos de redes neurais artificiais estocásticas treinadas utilizando inferência bayesiana que permitem representar valores de incerteza de suas predições de forma direta e possuem mecanismos naturais de regularização dos modelos que evitam o problema de sobreajuste (GAL, 2016; JOSPIN et al., 2022). O conceito de rede neural foi inicialmente proposto no início da década de 90, porém por um longo tempo a aplicação desses modelos foi considerada impraticável devido a problemas de escalabilidade e custo computacional do processo de inferência bayesiana utilizado no seu treinamento. Com sucesso das aplicações de redes neurais, recentemente o estudo desses modelos voltou a ser um tema de grande interesse e grandes avanços em relação a praticabilidade desses modelos foram alcançados. Apesar desses avanços, comparado ao número de estudos sobre aplicações de redes neurais tradicionais, estudos sobre aplicações de redes neurais bayesianas para solução de problemas reais ainda são bem escassos na literatura (JOSPIN et al., 2022).

1.1 Objetivo

Dado a escassez de estudos sobre aplicações de redes neurais bayesianas na literatura, esse trabalho tem como objetivo provar a aplicabilidade de modelos de redes neurais bayesianas para solução de problemas reais por meio de dois estudos de caso:

- **Estudo de caso 1 - Calibração de sensores de poluição do ar**

  Estudo de caso com objetivo específico de apresentar a aplicação de um modelo de rede neural bayesiana para solução do problema de calibração de sensores de poluição do ar de baixo custo, os quais são dispositivos de medição conhecidos por perderem precisão ao longo do tempo e necessitarem serem submetidos a procedimentos de calibração periodicamente.

- **Estudo de caso 2: Detecção de falhas em processos químicos**

  Estudo de caso com objetivo específico de apresentar a aplicação de um modelo de rede neural bayesiana para solução do problema de detecção de falhas em um processo químico real, visando o desenvolvimento de sistemas de monitoramento inteligentes capazes de realizar a detecção de falhas de forma autônoma e precisa.

Para cada estudo de caso, o desempenho da solução baseada em rede neural bayesiana é avaliado, assim com uma análise das vantagens e limitações do uso de rede neural bayesiana para solução do problema é realizada.

1.2 Estrutura da dissertação

Essa dissertação é composta por 6 capítulos. No capítulo 1 é realizada uma introdução sobre o aprendizado de máquina, em especial destaca-se o sucesso das técnicas de *deep learning*, os problemas dos modelos de redes neurais artificiais, e a motivação para uso de redes neurais bayesianas. Além disso, são apresentados o objetivo geral da dissertação e os estudos de casos a serem desenvolvidos na dissertação. No capítulo 2 é descrita a história das redes neurais artificiais, são descritas as abordagens de aprendizado das redes neurais, e por fim são apresentados os principais tipos de redes neurais artificiais. No capítulo 3 é apresentado o conceito teórico das redes neurais bayesianas assim como os métodos de inferência bayesiana que permitem sua implementação prática atualmente. Além disso, é apresentado um estudo prático dos principais métodos de aproximação de inferência bayesiana para redes neurais bayesianas. No capítulo 4 é descrito o desenvolvimento do estudo de caso sobre a aplicação de redes neurais bayesianas para a solução do problema de calibração de sensores de poluição do ar. No capítulo 5 é descrito

o desenvolvimento do estudo de caso sobre a aplicação de rede neural bayesiana para detecção de falhas em um processo químico. No capítulo 6 é apresentado as conclusões finais do trabalho e é sugerido assuntos para continuidade do estudo.

1.3 Lista de publicações resultantes do trabalho

SOUSA, R.; TAIRA, G. R.; PARK, S. W. Integração do Sistema Ciber-Físico para Sistema de Programação, Intertravamento e Controle de um Reator Batelada. The Journal of Engineering and Exact Sciences, v. 5, n. 5, p. 0424-0432, 2019.
DOI: https://doi.org/10.18540/jcecvl5iss5pp0424-0432

SANTOS, A. S.; FREITAS, L. G.; TEIXEIRA, I. C.; GAVA, V. L.; TAIRA, G. R.; QUILLE, R. V. E.; BRAGHETTO, K. R. Desafios e oportunidades da aplicação de Sistemas Ciberfísicos no monitoramento da poluição urbana. In: Anais do IV Workshop de Computação Urbana. SBC, p. 276-289, 2020.
DOI: https://doi.org/10.5753/courb.2020.12369

TAIRA, G. R.; LEAL, A. G.; SANTOS, A. S.; PARK, S. W. Bayesian Neural Network-Based Calibration for Urban Air Quality Sensors. In: Computer Aided Chemical Engineering. Elsevier, p. 1549-1554, 2022.
DOI: https://doi.org/10.1016/B978-0-323-95879-0.50259-9

TAIRA, G. R.; PARK, S. W.; ZANIN, A. C., PORFIRIO, C. R. Fault Detection in a Fluid Catalytic Cracking Process using Bayesian Recurrent Neural Network. IFAC-PapersOnLine, v. 55, n. 7, p. 715-720, 2022.
DOI: https://doi.org/10.1016/j.ifacol.2022.07.528

2 REDES NEURAIS ARTIFICIAIS

2.1 História das redes neurais artificiais

Dado o sucesso do *deep learning*, técnicas baseadas em modelos de redes neurais artificiais são atualmente consideradas um dos principais tipos de técnicas de aprendizado de máquina. Entretanto, modelos de redes neurais artificiais não são novos, e eles existem pelo menos desde os anos 40. O seu processo de desenvolvimento ao longo das décadas foi caracterizado por vários por altos e baixos, podendo-se destacar três ondas principais de grande interesse no estudo desses modelos (GOODFELLOW; BENGIO; COURVILLE, 2016).

A primeira onda de desenvolvimento das redes neurais artificiais ocorreu no período entre 1940 e 1960 com o surgimento da ideia de criar modelos computacionais capazes de imitar o processo de aprendizado que ocorre em cérebros humanos. O primeiro protótipo de rede neural artificial foi proposto por McCulloch e Pitts (1943). Esse protótipo consistia basicamente em um modelo matemático que descrevia o funcionamento de um neurônio de um cérebro humano. Esse modelo, denotado como uma função linear $f(x, w)$, simplesmente realizava classificações binárias (positivo ou negativo) de dados de entrada x de acordo com os valores desses dados e os valores de seus parâmetros pesos w. Esse modelo, entretanto, apenas imitava o processamento de informações em um neurônio do cérebro humano, não sendo capaz imitar o seu processo de aprendizado. Os parâmetros pesos do modelo, por exemplo, necessitavam ser selecionados manualmente por humanos para que o modelo realizasse as classificações de forma correta. Já no final da década de 50, o primeiro modelo matemático de um neurônio artificial com capacidade de autoaprendizagem dos valores de seus parâmetros pesos foi proposto por Rosenblatt (1958). Chamado de *perceptron*, esse modelo realizava classificações binárias igualmente ao modelo de McCulloch e Pitts (1943), porém ele já era capaz de aprender de forma autônoma os valores de seus parâmetros peso para realizar as classificações de forma correta, por meio da análise de exemplos rotulados de entradas da classe positiva e da classe negativa. Nesse mesmo período, um outro modelo de neurônio com capacidade de autoaprendizagem seus parâmetros pesos chamado de ADALINE (*Adaptative Linear Neuron*) foi proposto por Widrow e Hoff (1960). Diferentemente do modelo *perceptron* que realizava classificações binárias, o modelo ADALINE consistia em uma função capaz de aprender a realizar regressões e predizer valores numéricos contínuos a partir da análise de dados rotulados. O desenvolvimento do modelo ADALINE foi de grande contribuição para o desenvolvimento

das redes neurais artificiais ao apresentar o algoritmo de treinamento gradiente descendente estocástico, o qual é um algoritmo até hoje utilizado para treinamento de redes neurais. Com esses avanços, durante a década de 60 houve muito interesse no estudo desses modelos, no entanto, esse interesse foi interrompido com a publicação do trabalho de Minsky e Papert (1969), em que limitações relevantes desses modelos lineares foram identificadas ao tentar aplicar o modelo *perceptron* para o aprendizado da função XOR.

Já a segunda onda do desenvolvimento das redes neurais artificiais ocorreu no período entre 1980 e 1990, motivada pelo movimento de estudo da área de conexionismo (TOURETZKY; HINTON, 1985). O conexionismo se baseia na ideia central de que um grande número de unidades computacionais simples pode atingir um comportamento inteligente quando interligadas em rede. Essa percepção, que também se aplica aos neurônios em sistemas nervosos biológicos, motivou a criação de modelos de redes neurais artificiais tal como são conhecidos atualmente, baseados em redes interconectadas de unidades computacionais simples, tal como os neurônios artificiais *perceptron*, capazes de aprender e identificar padrões mais complexos presentes em dados. Além disso, uma das principais motivações para o interesse no estudo de redes neurais artificiais nesse período foi o desenvolvimento do algoritmo *back-propagation* (RUMELHART; HINTON; WILLIAMS, 1985), o qual permitiu o treinamento desses novos modelos de redes neurais na época e que até hoje é o algoritmo predominante para o treinamento de redes neurais. Dentre as grandes contribuições dessa segunda onda, pode-se destacar também o desenvolvimento de novos tipos de redes neurais capazes de modelar, por exemplo, dados sequenciais (HOCHREITER; SCHMIDHUBER, 1997) e imagens (LECUN et al., 1998). Essa segunda onda durou até o final da década de 90 e ela foi frustrada devido a uma combinação de fatores tais como o avanço no desenvolvimento de outras técnicas de aprendizado de máquina, como métodos de kernel e modelos gráficos, a falta de investimento e a necessidade de recursos computacionais com maior capacidade de processamento (GOODFELLOW; BENGIO; COURVILLE, 2016).

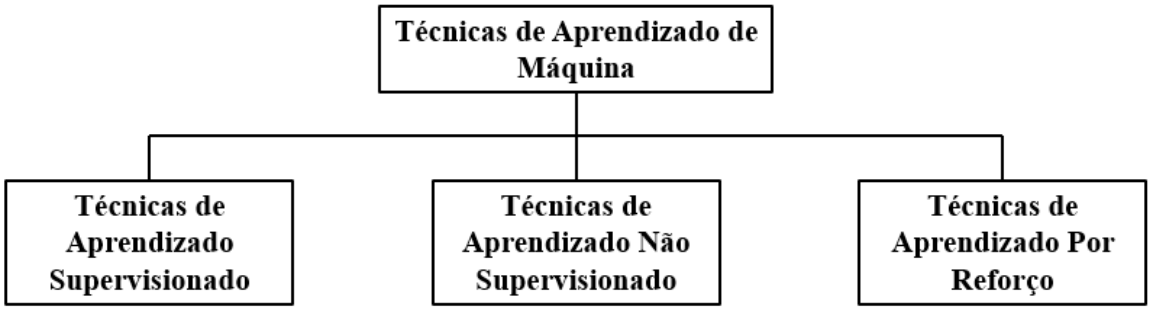
A terceira onda de desenvolvimento das redes neurais artificiais, por sua vez, se iniciou em 2006, com a publicação do trabalho de Hinton, Osindero e Teh (2006) em que é apresentado uma inovadora estratégia para treinamento de redes neurais chamada de *greedy layer-wise pre-training*, o qual envolve um pré-treinamento não supervisionado separado de cada camada de uma rede neural, a fim de melhorar eficiência do processo treinamento. Esse trabalho foi considerado disruptivo, pois mostrou a possibilidade de se treinar de forma eficiente redes neurais profundas e complexas que até então não podiam ser treinadas. Desse modo, essa terceira onda popularizou o termo *deep learning* para se referir ao uso de redes neurais para o aprendizado de máquina com objetivo de

enfatizar a atual capacidade do treinamento de redes neurais mais complexas e profundas para solução de problemas de grande complexidade e assim impulsionar uma nova era do desenvolvimento das redes neurais artificiais. Além disso, outros importantes fatores que vem impulsionando essa terceira onda são a maior disponibilidade de dados em decorrência do fenômeno do *big data* e o elevado poder de processamento dos atuais recursos computacionais. Esses fatores têm possibilitado que redes neurais profundas e complexas sejam facilmente treinadas e capacitadas para realizar predições com elevada precisão, dado poder de processamento de dados dos atuais computadores e a elevada quantidade de conhecimentos que podes ser extraídos do *big data*. Atualmente, redes neurais artificiais têm superado a performance de outras técnicas de aprendizado de máquina na solução de diversos problemas da área de inteligência artificial e suas aplicações tem se tornado uma tendência em diversas áreas científicas (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.2 Abordagens de aprendizado das redes neurais artificiais

Em geral, tal como ilustrado na Figura 3, as técnicas de aprendizado de máquina podem ser divididas em três categorias (MURPHY, 2012): técnicas de aprendizado supervisionado, técnicas de aprendizado não supervisionado e técnicas de aprendizado por reforço.

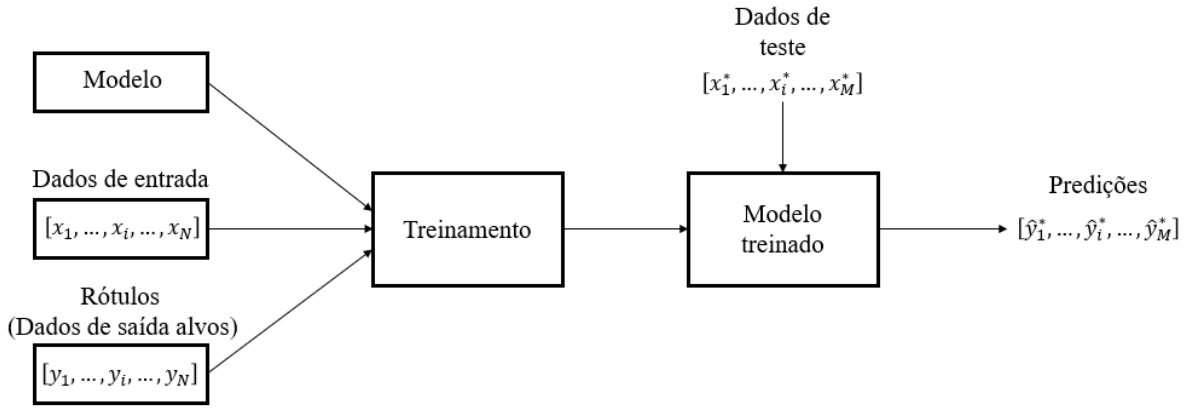
Figura 3: Tipos de técnicas de aprendizado de máquina



As técnicas de aprendizado supervisionado são utilizadas para o aprendizado a partir de conjuntos de dados rotulados. Conjuntos de dados desse tipo são compostos por N pares de dados (x, y), onde x denota um dado de entrada e y denota seu respectivo rótulo (também chamado de dado de saída alvo), e por essa razão são comumente denotados por $D = \{(x_i, y_i)\}_{i=1}^{N}$. As técnicas de aprendizado supervisionado têm como objetivo treinar modelos a realizar o mapeamento entre os dados de entrada e seus res-

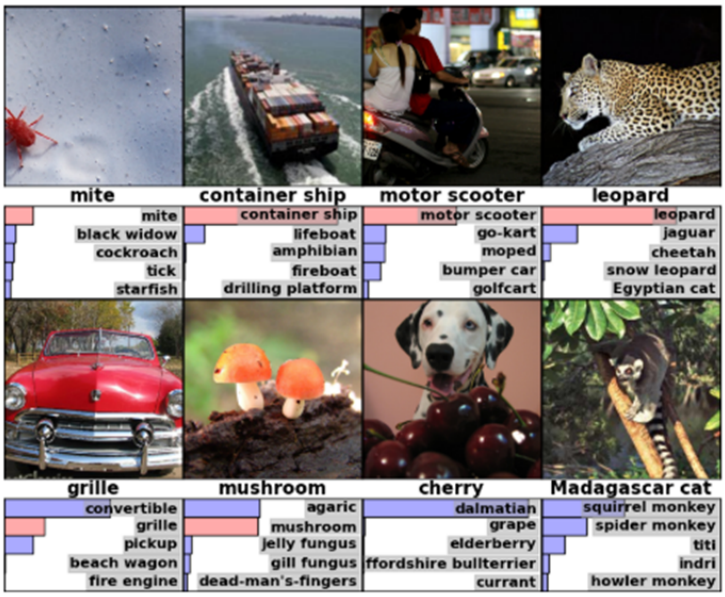
pectivos dados de saída, ou seja, elas visam capacitar modelos a predizer de forma correta o valor de y_i correspondente a um dado de entrada x_i (MURPHY, 2012). Para que isso seja possível, conjuntos de dados rotulados são fornecidos aos modelos em um processo de treinamento para que eles extraiam os padrões de mapeamento entre entradas e saídas, e assim aprendam a realizar as predições. Tal como ilustrado na Figura 4, o objetivo final dessa abordagem de aprendizado é que o modelo, após treinamento, ao receber um novo dado de entrada x_i^*, ele seja capaz de predizer corretamente o dado de saída $\hat{y}_i^*$ correspondente a esse novo dado de entrada. Essa capacidade desejada para o modelo é chamada de capacidade de generalização.

Figura 4: Abordagem de aprendizado supervisionado



As técnicas de aprendizado supervisionado são normalmente utilizadas para solução de problemas de classificação e regressão. No caso de um problema de classificação, um modelo é treinado de forma a fornecer como resposta uma predição da classe mais provável à qual um dado de entrada x_i^* pertence. Nesse caso a resposta $\hat{y}_i^*$ do modelo normalmente consiste em um vetor numérico contendo valores de probabilidade para cada classificação possível para o dado de entrada. Um exemplo bem conhecido de problema de classificação em que técnicas de aprendizado supervisionado são utilizadas para sua solução é o problema de reconhecimento de objetos em imagens, tal como ilustrado na Figura 5.

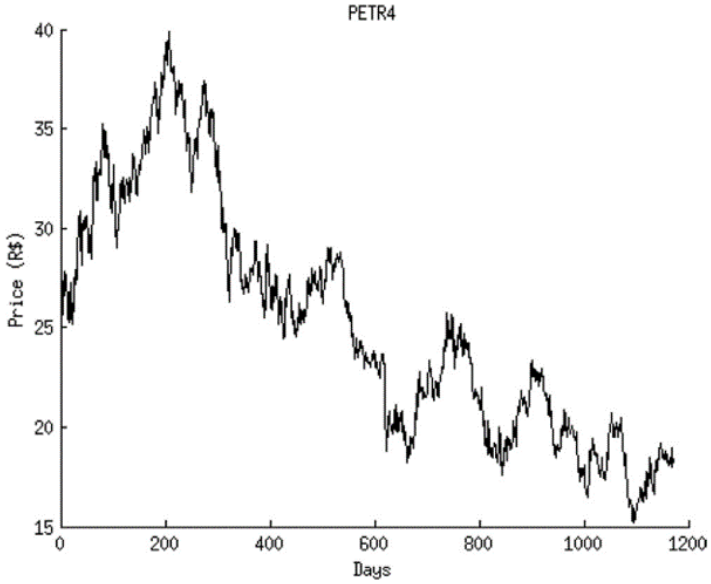
Figura 5: Exemplo de reconhecimento de objetos em imagens



Fonte: Krizhevsky, Sutskever e Hinton (2012)

Já no caso de um problema de regressão, um modelo é treinado para fornecer como resposta $\hat{y}_i^*$ uma predição o valor numérico real correspondente ao dado de entrada x_i^*. Um caso de problema de regressão em que técnicas de aprendizado supervisionado podem ser utilizadas para sua solução é por exemplo a previsão de séries temporais do preço de ações financeiras, tal como ilustrado na Figura 6.

Figura 6: Série temporal de preços de ações de uma empresa brasileira



Fonte: Cavalcante et al. (2016)

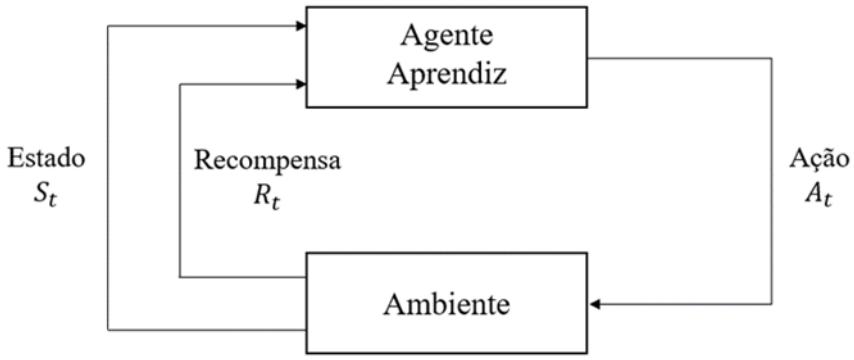
As técnicas de aprendizado não supervisionado, por sua vez, são normalmente aplicadas para o aprendizado a partir de conjuntos de dados não rotulados, que são compostos apenas por N dados de entrada x, sem rótulos associados a eles. Desse modos, esses conjuntos de dados são normalmente denotados por $D = \{x_i\}_{i=1}^{N}$. As técnicas de aprendizado não supervisionado são utilizadas para extração de padrões e estruturas interessantes presentes nesses conjuntos de dados e por essa razão são conhecidas como técnicas de descobrimento de conhecimento (*knowledge discovery*) (MURPHY, 2012). O exemplo mais popular de aplicação dessas técnicas é na solução de problemas de agrupamento de dados (*clustering*), tal como o ilustrado na Figura 7, mas também são utilizadas para solução de problemas de sumarização de dados, associação de dados e detecção de *outliers* (MURPHY, 2012).

Figura 7: Exemplo de agrupamento de dados

Por fim, as técnicas de aprendizado por reforço consistem em um grupo de técnicas que não utilizam um conjunto de dados fixo para realizar o aprendizado. Essas técnicas utilizam uma abordagem de aprendizado baseada na interação entre um agente aprendiz e seu ambiente de operação. Tal como ilustrado na Figura 8, essa abordagem de aprendizado é utilizada normalmente para o caso em que o agente aprendiz realiza ações sobre o ambiente visando alcançar um certo objetivo. Nesse esquema de aprendizado, de acordo com a execução de ações pelo agente, mudanças ocorrem no estado do ambiente e as informações dessas mudanças junto do seu respectivo valor de recompensa são fornecidos para o agente de aprendiz. De acordo com essas informações, o aprendiz então decide qual será a próxima ação a ser realizada sobre o ambiente, refinando as estratégias de qual ação deve ser escolhida para alcançar seu objetivo. Para isso é utilizado um sistema de maximização de recompensas, no qual ações realizadas pelo agente sobre o ambiente que o aproximem de seu objetivo recebem uma maior recompensa enquanto que ações que o distanciem de seu objetivo são menos recompensadas. Dessa forma, o agente acaba

aprendendo a identificar quais ações devem ser realizadas e quais devem ser evitadas para atingir seu objetivo (SUTTON; BARTO, 2018). Os métodos de aprendizado por reforço são normalmente aplicados para solução de problemas que exigem tomadas de decisão sequenciais, comumente encontrados na área de robótica e no desenvolvimento de sistemas computacionais capazes de jogar jogos de tabuleiro de forma autônoma, tal como o *AlphaGo* (SILVER et al., 2016).

Figura 8: Abordagem do aprendizado por reforço



As redes neurais artificiais são bem versáteis e podem ser utilizadas nas três abordagens de aprendizado de máquina. No entanto, esses modelos são mais reconhecidos pelo seu sucesso quando utilizados para solução de problemas de aprendizado supervisionado (LECUN; BENGIO; HINTON, 2015). Dado que os estudos de casos a serem desenvolvidos nesse trabalho envolvem problemas de aprendizado supervisionado, apenas tipos de redes neurais utilizadas para solução de aprendizado supervisionado serão descritas na sessão seguinte.

2.3 Tipos de redes neurais artificiais

Diferentes tipos de redes neurais artificiais são utilizados atualmente. Em especial, três tipos devem ser destacados: as redes neurais *feedforward*, as redes neurais recorrentes e as redes neurais convolucionais.

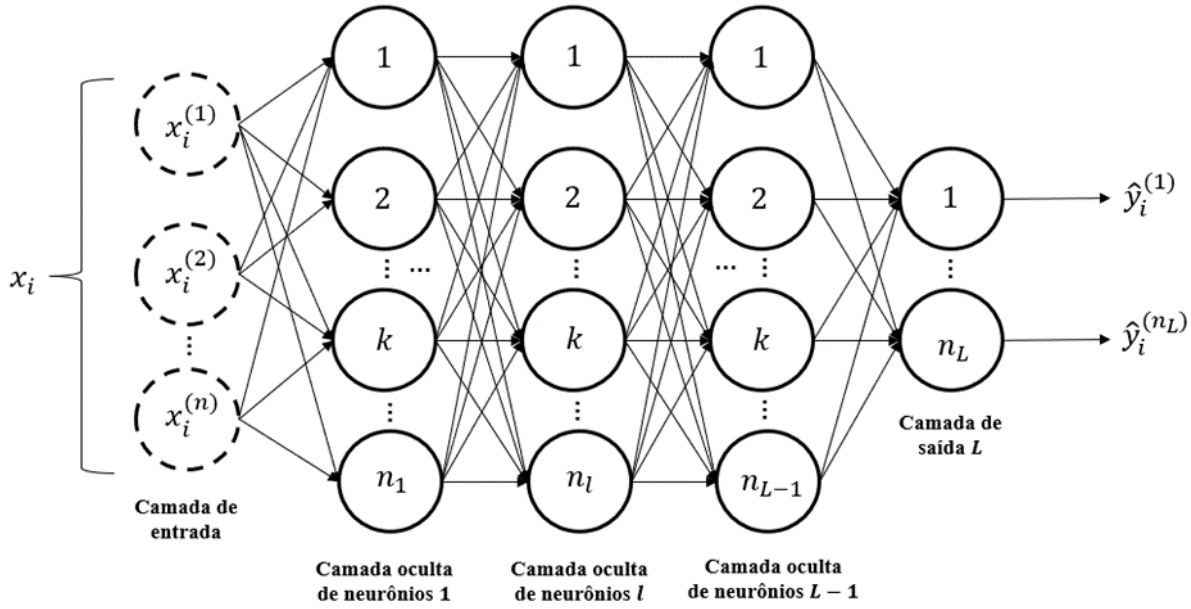
2.3.1 Redes neurais *feedforward*

As redes neurais *feedforward*, também chamadas de redes *multilayer perceptron* (MLP), são consideradas os exemplos mais clássicos de redes neurais artificiais (GOODFELLOW; BENGIO; COURVILLE, 2016). Elas foram desenvolvidas no contexto do estudo do conexionismo, durante a segunda onda de pesquisa sobre redes neurais. Elas são consideradas a base de outros tipos de redes neurais artificiais, tais como as redes

neurais recorrentes e as redes neurais convolucionais, e por essa razão, muitos conceitos de sua estrutura, métodos de treinamento e problemas também são válidos para esses outros dois tipos de redes neurais.

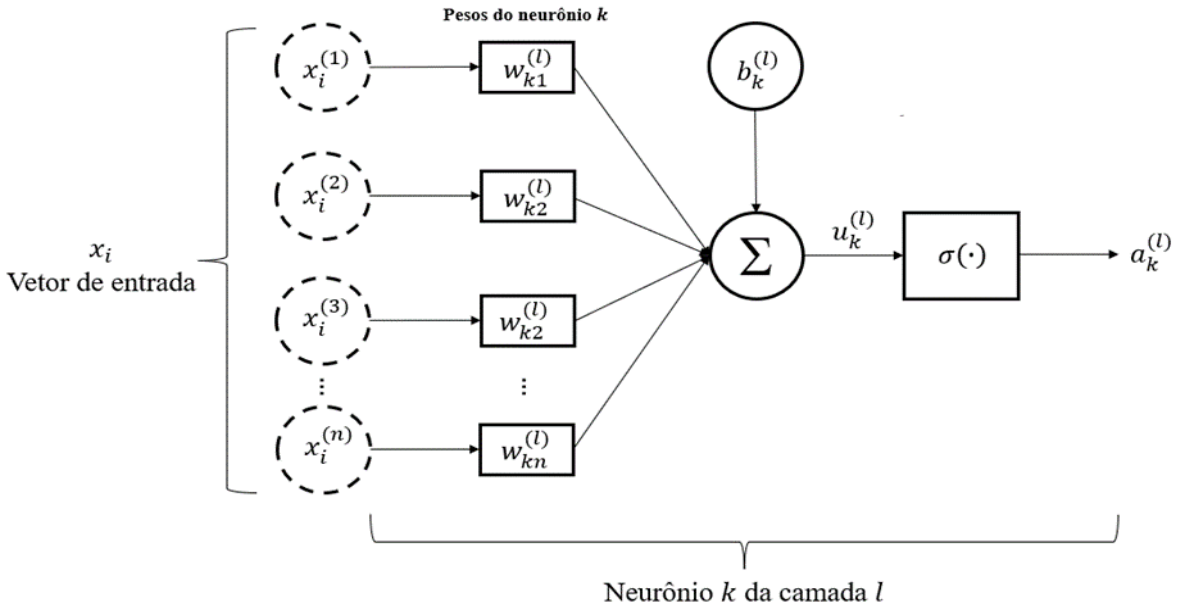
As redes neurais *feedforward* são comumente descritos como redes de múltiplas camadas de neurônios artificiais interconectados. Tal como ilustrado na Figura 9, suas camadas são estruturadas de forma sequencial, iniciada por uma camada de entrada de dados, seguida de uma ou múltiplas camadas ocultas de neurônios e finalizada com uma camada de saída. Nessa estrutura sequencial cada neurônio presente em uma camada se conecta com todos os neurônios da camada seguinte.

Figura 9: Exemplo de rede neural *feedforward*



Os neurônios artificiais normalmente utilizados para essas redes neurais consistem em modelos matemáticos baseados no modelo *perceptron* proposto por Rosenblatt (1958) (HAYKIN, 2010). Tal como exemplificado na Figura 10, considerando um neurônio artificial k desse tipo pertencente à camada oculta l, o processamento de um dado vetor de entrada $x_i = \left(x_i^{(1)}, ..., x_i^{(n)}\right)$ nesse neurônio consiste na designação de pesos $\left\{w_{k1}^{(l)}, ..., w_{kn}^{(l)}\right\}$ para cada uma das n dimensões de x_i, no cálculo da soma ponderada dos valores de cada dimensão utilizando esses pesos, na adição de um termo *bias* $b_k^{(l)}$ a soma ponderada e na submissão do resultado dessa soma final a uma função de ativação $\sigma(\cdot)$. Essa função ativação, então, gera a ativação $a_k^{(l)}$, a qual consiste no valor de saída do neurônio.

Figura 10: Modelo de neurônio artificial



Em termos matemáticos, as operações de processamento do neurônio ilustrado na Figura 10 são descritas pelas Equações 2.1 e 2.2.

$$u_k^{(l)} = \sum_{i=1}^{n} x_i w_i^{(l)} + b_k^{(l)} \tag{2.1}$$

$$a_k^{(l)} = \sigma(u_k^{(l)}) \tag{2.2}$$

Quando acoplados a uma rede neural *feedforward*, tal como na Figura 9, os neurônios artificiais atuam de forma conjunta no processamento do vetor de entrada x_i para geração de sua respectiva resposta $\hat{y}_i$. Tal como descrito nas Equações 2.3, 2.4 e 2.5, matematicamente, em uma rede neural *feedforward* com $L-1$ camadas ocultas, a atuação conjunta dos neurônios de no processamento de um dado vetor de entrada x_i e a geração de sua respectiva resposta $\hat{y}_i$ podem ser vistas como a realização de uma sequência de operações matriciais lineares e transformações não lineares executadas pelas funções de ativação $\sigma_l(\cdot)$:

$$a_0 = x_i \qquad l = 0 \tag{2.3}$$

$$a_l = \sigma_l(a_{l-1} \cdot W_l + b_l) \qquad l = 1, \ldots, L-1 \tag{2.4}$$

$$\hat{y}_i = a_L = \sigma_L(a_{L-1} \cdot W_L + b_L) \qquad l = L \tag{2.5}$$

Onde a_l de l até $l = L-1$ representam as ativações geradas pelas camadas ocultas da rede neural e a_L representa a ativação gerada pela camada. Já W_l e b_l denotam a matriz com todos os parâmetros pesos da camada l e o vetor com todos os termos *bias* da camada l, respectivamente.

2.3.1.1 Funções de ativação

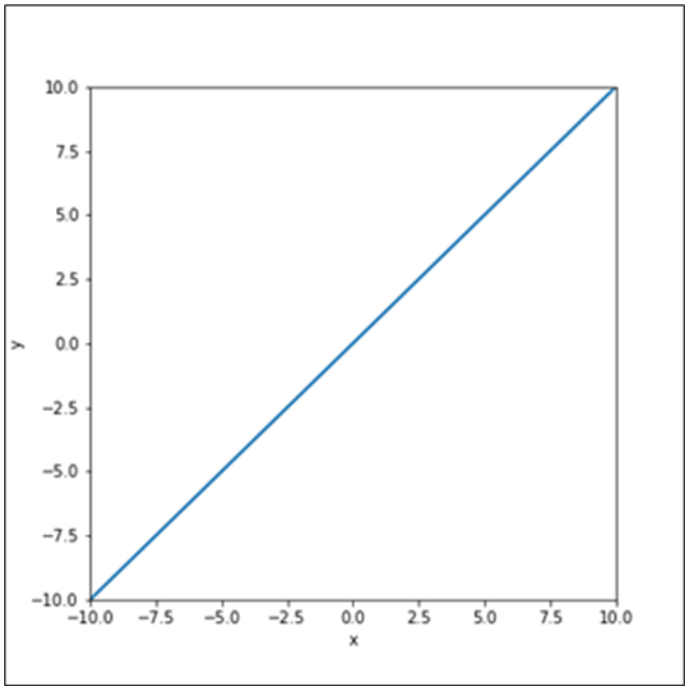
As funções de ativação têm como papel limitar os valores das respostas geradas por um neurônio dentro um intervalo de valores razoável (HAYKIN, 2010) e o tipo de função de ativação utilizada por uma rede neural podem influenciar seu desempenho. Sendo assim, a seguir são descritas as funções de ativação historicamente mais utilizadas para modelos de redes neurais *feedforward* assim como em outros tipos de redes neurais (GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Função linear**

A função de ativação linear é descrita pela Equação 2.6 e ilustrada na Figura 11.

$$\sigma(u) = au + b \tag{2.6}$$

Figura 11: Função de ativação linear



Tal como será explicado na seção 2.3.1.2, os principais métodos para treinamento de redes neurais atualmente são baseados no algoritmo gradiente descendente, o qual utiliza gradientes de uma função custo para atualizar os parâmetros da rede neural durante o processo

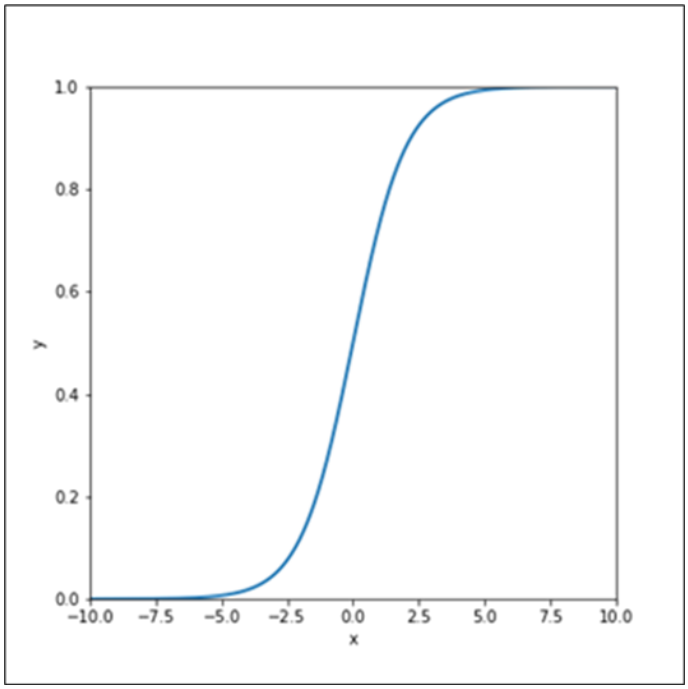
de treinamento. Pelo fato de a função linear ser de fácil diferenciação, ela é reconhecida por facilitar o treinamento de redes neurais. Entretanto, uma grande desvantagem dessa função de ativação é que ela não permite que as redes neurais aprendam a descrever comportamentos não lineares, impossibilitando que elas utilizadas para sistemas complexos não lineares. Um caso específico de sua utilização é no caso da camada de saída de uma rede neural utilizada para realizar regressão.

- **Função sigmoide**

A função de ativação sigmoide é descrita pela Equação 2.7 e ilustrada na Figura 12.

$$\sigma(u) = \frac{1}{1 + e^{-u}} \tag{2.7}$$

Figura 12: Função de ativação sigmoide



Esse tipo função tem formato de S e gera respostas no intervalo entre respostas entre 0 e 1. Pelo fato de ser não-linear, ela permite que as redes neurais aprendam a descrever comportamentos não lineares presentes nos dados. Porém, um relevante problema dessa função de ativação é que ela é sofre saturação em maior parte de seu domínio. A função tende a 0 quando os valores de u são muito negativos e tende para 1 quando os valores de u são muito negativos, sendo sensível apenas para valores de u muito próximos de 0. Esse problema pode dificultar o processo de treinamento de redes neurais, pois sua derivada tende a 0 em maior parte do seu domínio, o que pode fazer com que os gradientes utilizados no processo de treinamento também tendam a 0, impossibilitando que a atualização dos

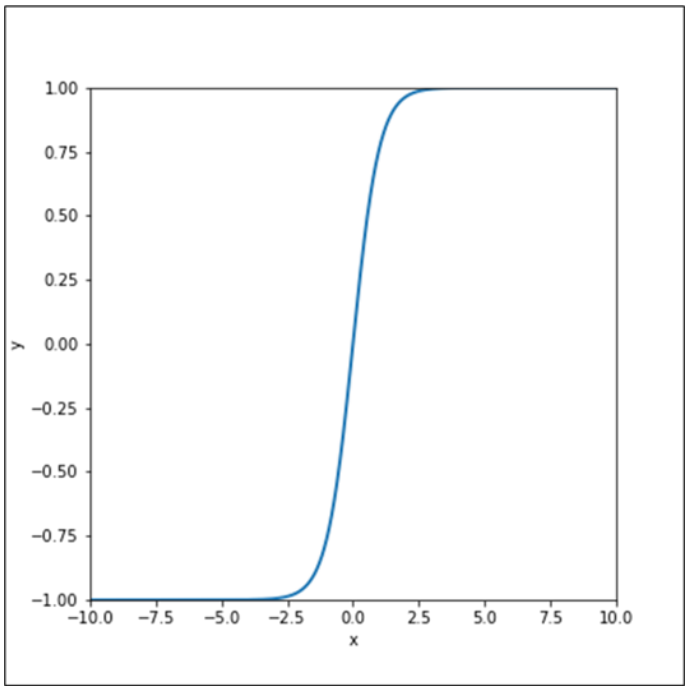
parâmetros dos modelos. Por essa razão, essa função de ativação não é atualmente a preferida para uso em redes neurais. Normalmente apenas em alguns casos de redes neurais recorrentes ela é utilizada (GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Função tangente hiperbólica**

A função de ativação tangente hiperbólica é descrita pela Equação 2.8 e ilustrada na Figura 13.

$$\sigma(u) = \tanh(u) = \frac{1}{1 + e^{-2u}} - 1 \tag{2.8}$$

Figura 13: Função de ativação tangente hiperbólica



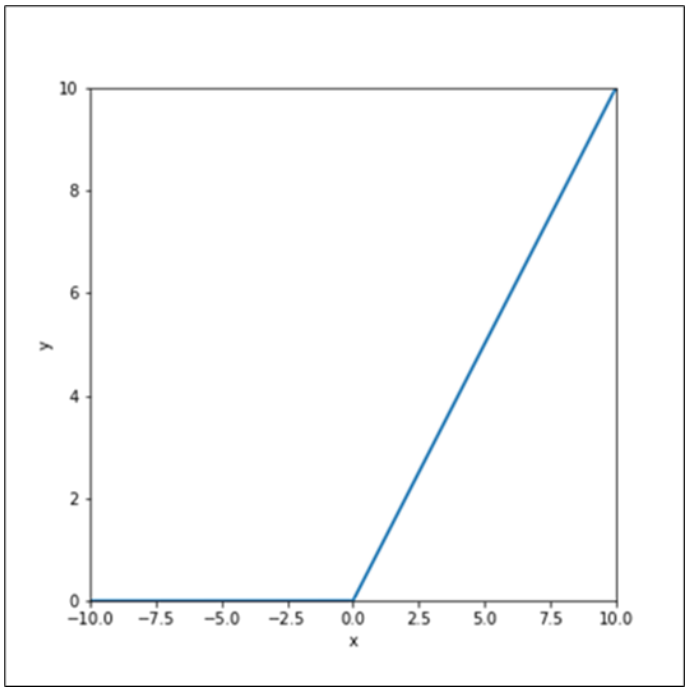
Esse tipo função também tem formato de S, porém gera respostas no intervalo entre respostas entre -1 e 1. Pelo fato de ser não-linear, ela também permite que as redes neurais aprendam a descrever comportamentos não lineares. Ela apresenta um problema semelhante ao da função de ativação sigmoide com relação à saturação em seu domínio, entretanto ela é reconhecida por performar melhor no processo de treinamento das redes neurais, principalmente pelo fato de possuir um comportamento semelhante à função linear $f(u) = u$ para valores de u próximos de 0 e fornecer valores de derivada próximos de 1 nessa região do domínio. Por essa razão, normalmente a utilização da função tangente hiperbólica é preferível em comparação à função sigmoide (GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Função *Rectified Linear Unit***

A função de ativação *Rectified Linear Unit* (ReLU) é descrita pela Equação 2.9 e ilustrada na Figura 14.

$$\sigma(u) = \max(0, u) \tag{2.9}$$

Figura 14: Função de ativação ReLU



Essa função de ativação é atualmente a mais utilizada para os modelos de redes neurais em geral. Ela é considerada uma função não linear, porém que permanece muito próxima de ser linear no sentido de que ela é uma função particionada em duas partições lineares. Por essa razão, essa função é de fácil diferenciação e é reconhecida por facilitar o treinamento de redes neurais, mantendo a capacidade das redes neurais descreverem comportamentos não lineares. Ou seja, ela é considerada uma função de ativação eficiente que pode ser considerada como uma sugestão padrão para os modelos de redes neurais em geral (GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Função *softmax***

No caso de problemas de classificação, a função *softmax* é utilizada para gerar como respostas dos neurônios da camada de saída probabilidades com relação às j possíveis classes de classificação. Para isso ela realiza a normalização de todas as j respostas fornecidas pela camada de saída da rede neural:

$$\sigma(u)_j = \frac{\exp(u_j}{\sum_j \exp(u_j)} \tag{2.10}$$

Um valor próximo de 1 para $\sigma(u)_j$ demonstra uma grande probabilidade para a classe j, e um valor próximo de 0 para $\sigma(u)_j$ demonstra uma baixa probabilidade para a classe j.

2.3.1.2 Treinamento de redes neurais *feedforward*

Para que a rede neural *feedforward* aprenda a processar um dado vetor de entrada x_i e gerar sua respectiva resposta $\hat{y}_i$ correta, ela deve ser submetida a um processo de treinamento. Denotando uma rede neural *feedforward* como uma função $\hat{y} = f^\theta(x)$, onde o termo θ representa todos os parâmetros da rede, que incluem todos os parâmetros pesos e *bias* utilizados em suas camadas, o seu processo de treinamento é tradicionalmente abordado como um problema de otimização de seus parâmetros. Dado um conjunto de dados rotulados de treinamento $D = \{(x_i, y_i)\}_{i=1}^N$, tal que $X = [x_1; \ldots; x_N]$ denota o conjunto de dados de entrada e $Y = [y_1; \ldots; y_N]$ denota o conjunto de seus respectivos dados de saída alvos, o processo de treinamento da rede neural consiste na determinação dos valores ótimos $\hat{\theta}$ para os parâmetros da rede neural que minimizam a função custo $J(\theta)$ descrita pela Equação 2.12.

$$\min_\theta \quad J(\theta) \tag{2.11}$$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} ||f^\theta(x_i) - y_i||_p \tag{2.12}$$

Onde $||\cdot||_p$ denota uma norma utilizada como métrica do erro entre as respostas geradas pela rede neural $f^\theta(x_i)$ e os valores alvos y_i. Na prática, o tipo de norma utilizada para computar os erros de predição das redes neurais durante o seu treinamento depende do tipo do problema a ser resolvido. Para problemas de regressão, o erro quadrado médio (Equação 2.13) é normalmente utilizado como funções custo:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} (f^\theta(x_i) - y_i)^2 \tag{2.13}$$

Já para problemas de classificação, a entropia cruzada (*cross-entropy*) (Equação 2.14) é normalmente utilizada:

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k} y_{i,j} \log(f^\theta(x_i)_j) \tag{2.14}$$

Onde k é o número de classes possíveis para classificação e $f^\theta(x_i)_j$ é o elemento j da

resposta fornecida por uma função de ativação *softmax*.

A solução do problema de otimização para o treinamento de uma rede neural *feedforward* é tradicionalmente obtida utilizando os algoritmos *back-propagation* (RUMELHART; HINTON; WILLIAMS, 1985) e gradiente descendente. Nessa abordagem de treinamento, normalmente os parâmetros θ são inicialmente configurados com valores randômicos e utilizados em um processamento inicial pela rede neural de todos os dados de entrada de treinamento $[x_1; \ldots; x_N]$. Esse processamento inicial resulta na geração de um conjunto de respostas $[f^\theta(x_1); \ldots; f^\theta(x_N)]$, o qual é confrontado ao conjunto de dados de saída alvos $[y_1; \ldots; y_N]$ para o cálculo da função custo $J(\theta)$. Após o cálculo da função custo, os valores de seus gradientes com relação aos parâmetros θ da rede são calculados utilizando o algoritmo *back-propagation*, os quais, em seguida, são utilizados pelo algoritmo de gradiente descente para atualizar os valores dos parâmetros θ visando minimizar a função custo $J(\theta)$. Com os novos valores dos parâmetros, um processamento dos dados de entrada de treinamento é realizado novamente e o mesmo procedimento para atualização dos parâmetros θ é executado. Todo esse processo é repetido de forma iterativa até que a função custo seja minimizada.

Com relação ao cálculo dos gradientes da função $J(\theta)$ com relação aos parâmetros θ utilizando o algoritmo *back-propagation*, ele consiste basicamente na aplicação sequencial da regra da cadeia no sentido inverso do processamento dos dados. Dessa forma, o cálculo dos gradientes de $J(\theta)$ com relação os parâmetros θ_l de cada camada l da rede neural é descrito pelas Equações 2.15, 2.16 e 2.17.

$$\delta_L = \frac{\partial J(\theta)}{\partial a_L} \otimes \sigma'(\theta_L a_{L-1}) \qquad l = L \tag{2.15}$$

$$\delta_l = \theta_{l+1} \cdot \delta_{l+1} \otimes \sigma'(\theta_l a_{l-1}) \qquad l = 1, \ldots, L-1 \tag{2.16}$$

$$\frac{\partial J(\theta)}{\partial \theta_l} = \delta_l \cdot a_{l-1} \qquad l = L \tag{2.17}$$

Onde θ_l representa os parâmetros das camadas $l = 1, \ldots, L$ da rede neural e o operador $\otimes$ denota um produto de Hadamard.

Já a atualização dos parâmetros θ_l pelo algoritmo de gradiente descendente é expressa por:

$$\theta_l = \theta_l - \eta \frac{\partial J(\theta)}{\partial \theta_l} \tag{2.18}$$

Onde η é um termo de taxa de aprendizado.

Entretanto, um dos principais problemas ao se utilizar o algoritmo de gradiente

descendente no treinamento de redes neurais em geral, é que sua convergência pode ser afetada pela presença de mínimos locais no espaço da função custo (GOODFELLOW; BENGIO; COURVILLE, 2016). Sendo assim, ao longo dos anos diversas técnicas mais eficientes em termos de convergência têm sido propostas na literatura.

Um problema encontrado no algoritmo original de gradiente descendente é que, em cada iteração, a rede neural deve processar o conjunto inteiro de dados de treinamento antes de atualizar os parâmetros, o que resulta em uma convergência pouco eficiente. Para solucionar esse problema uma variação do algoritmo de gradiente descendente, chamado algoritmo de gradiente descendente estocástico foi proposto. A principal diferença desse algoritmo com relação ao original é que invés de atualizar os parâmetros após processar todo o conjunto de dados de treinamento, ele atualiza os parâmetros após o processamento de cada instância do conjunto de treinamento. Essa modificação melhora a significativamente a convergência, porém gera um aumento significativo no custo computacional do algoritmo dado que ele necessita que a função custo e seus gradientes sejam calculados para cada amostra, podendo ser impraticável para conjuntos de dados muito grandes. Para reduzir esse custo computacional, o algoritmo chamado gradiente descendente em batelada foi criado como uma espécie de meio termo entre o algoritmo de gradiente descendente original e o algoritmo de gradiente descendente estocástico. No algoritmo de gradiente descendente em batelada, o conjunto de dados de treinamento é dividido em bateladas de amostras de dados e os parâmetros são atualizados após o processamento de cada batelada. Essa modificação garante uma convergência para o algoritmo sem aumentar de forma significativa o seu custo computacional.

Um outro problema para encontrado para a convergência do algoritmo original de gradiente descendente é a necessidade de determinar um valor para a taxa de aprendizado η. A determinação de um valor da taxa de aprendizado ótimo é essencial para convergência durante o treinamento. Uma taxa muito grande pode resultar em correções muito bruscas para parâmetros ao longo da superfície da função custo, dificultando a localização do ponto ótimo, enquanto que uma taxa muito pequena pode diminuir a velocidade de convergência. Dessa forma diversos algoritmos baseados no gradiente descendente capazes de adaptar o valor da taxa de aprendizado ao longo do processo de treinamento de redes neurais foram propostos. Entre esses algoritmos, os algoritmos *AdaGrad* (DUCHI; HAZAN; SINGER, 2011), *RMSProp* (TIELEMAN; HINTON, 2012) e *Adam* (KINGMA; BA, 2014) são os mais utilizados na prática.

2.3.1.3 Problema de sobreajuste

O principal objetivo ao treinar uma rede neural é torná-la capaz de realizar generalizações, ou seja, após treinamento, é desejado que a rede neural seja capaz de receber um novo dado de entrada x_i^* e predizer corretamente o seu dado de saída correspondente $\hat{y}_i^*$. Entretanto, um problema que pode afetar a capacidade de generalização das redes neurais é a tendência desses modelos sofrerem sobreajuste. O sobreajuste, também conhecido pela nomenclatura em inglês *overfitting*, consiste em um problema que ocorre durante o treinamento da rede neural em que o modelo acaba memorizando as propriedades dos dados de treinamento, tornando-o incapaz de realizar predições precisas quando submetidos a novos dados de entrada com propriedades diferentes dos dados de treinamento (GOODFELLOW; BENGIO; COURVILLE, 2016).

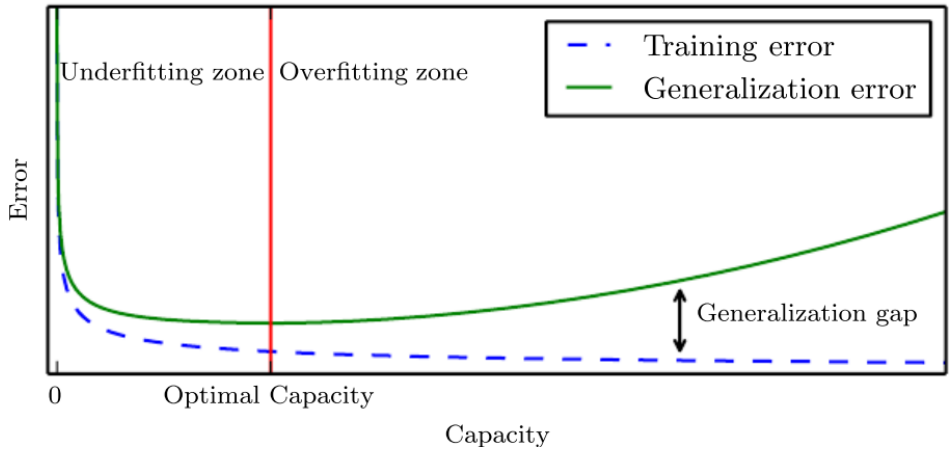
O problema de sobreajuste de uma rede neural pode ser identificado por meio da plotagem do gráfico das curvas do erro de treinamento e do erro de generalização ao longo de seu processo de treinamento, tal como exemplificado da Figura 15. O erro de treinamento consiste basicamente no valor da função custo em cada iteração do processo de treinamento da rede neural, enquanto que o erro de generalização consiste na soma dos erros de predição da rede neural com relação a um conjunto de dados de validação, após cada iteração do processo de treinamento. Tal como pode ser observado na Figura 15, a curva do erro de treinamento tende a decrescer durante o processo de treinamento, enquanto que a curva de generalização tende a um formato em U. O sobreajuste da rede neural é identificado no momento em que a curva do erro de generalização começa a descolar da curva do treinamento, gerando uma lacuna entre as curvas. Essa lacuna demonstra que, apesar do seu baixo erro de treinamento, a rede neural não é capaz de realizar generalizações de forma precisa.

Dessa forma, para evitar o problema do sobreajuste, uma estratégia muita utilizada é aplicação de técnicas de regularização. Essas técnicas consistem basicamente na adição de um termo de penalização $R(\theta)$ sobre a função custo $J(\theta)$ a ser minimizada (Equação 2.19), onde α denota um hiperparâmetro de ponderação. Esses termos de penalização dificultam que os parâmetros da rede neural memorizem as propriedades dos dados de treinamento, evitando assim o sobreajuste do modelo.

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} ||f^{\theta}(x_i) - y_i||_p + \alpha R(\theta) \tag{2.19}$$

Técnicas de regularização tais como as regularizações $L1$ e $L2$ (NG, 2004), e o *dropout* (SRIVASTAVA et al., 2014) são comumente utilizados no treinamento de redes neurais em geral.

Figura 15: Curvas do erro de treinamento e do erro de generalização ao longo do processo de treinamento

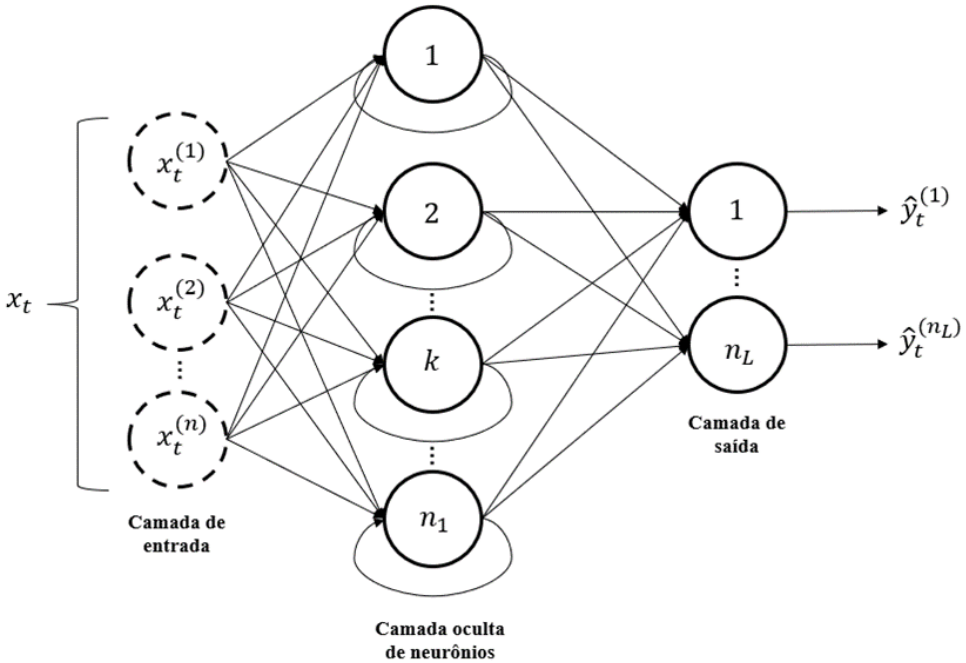


Fonte: Goodfellow, Bengio e Courville (2016)

2.3.2 Redes neurais recorrentes

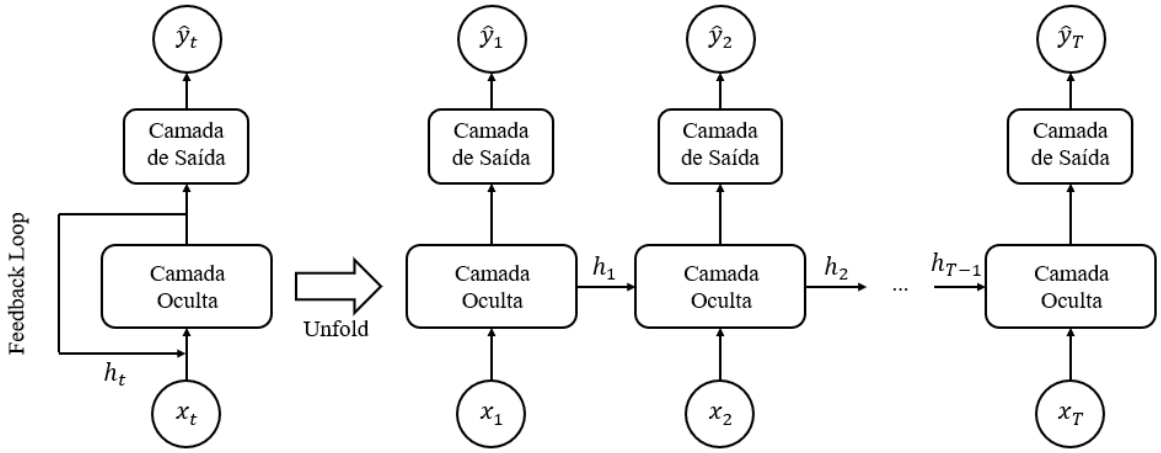
As redes neurais recorrentes consistem em um tipo específico de rede neural especializado no processamento de dados sequenciais tais como áudios, textos e séries temporais (GOODFELLOW; BENGIO; COURVILLE, 2016). Tal como exemplificado na Figura 16, esse tipo de rede neural também é estruturado como uma sequência de camadas de neurônios interconectados, porém diferentemente de uma rede neural *feedforward*, em sua camada oculta os neurônios apresentam conexões com eles próprios que permitem que eles reprocessem suas próprias respostas.

Figura 16: Exemplo de uma rede neural recorrente padrão



Tal como é ilustrado na Figura 17, essas autoconexões nos neurônios da camada oculta podem ser vistas como um *loop* de retroalimentação (um *feedback loop*) em sua estrutura que permite que informações obtidas em dados de entrada já analisados pela rede neural persistam em um estado interno h_t da rede, criando assim uma espécie de memória para a rede neural (GRAVES, 2012). Essa memória, portanto, possibilita que a rede capture a dependência de uma amostra x_t de uma sequência com relação a amostras anteriores $(x_{t-1}, x_{t-2}, \ldots)$ da mesma sequência.

Figura 17: Rede neural recorrente desdobrada



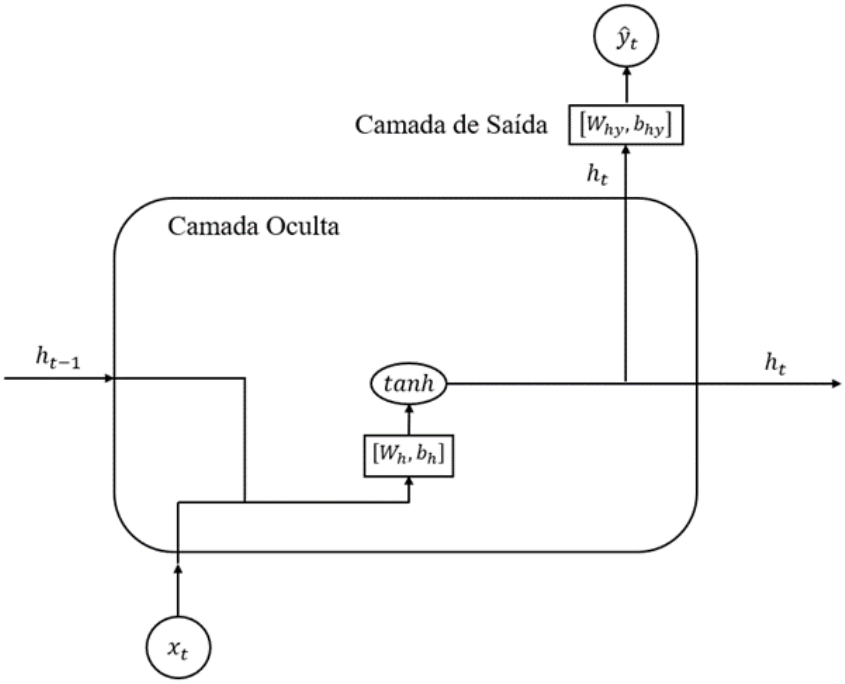
A estrutura interna da camada oculta de uma rede neural recorrente padrão (OLAH, 2015) é ilustrada na Figura 18. Utilizando essa estrutura, dado uma sequência de dados de entrada $X = [x_1; \ldots; x_T]$, o processamento desses dados sequenciais pela rede neural para produção de respectivas respostas $\hat{Y} = [\hat{y}_1; \ldots; \hat{y}_T]$ consiste basicamente na iteração das Equações 2.20 e 2.21 para $t = 1, \ldots, T$.

$$h_t = \tanh(W_h[h_{t-1}, x_t] + b_h) \tag{2.20}$$

$$\hat{y}_t = W_{hy}h_t + b_{hy} \tag{2.21}$$

Onde h_t denota o estado interno oculto da rede neural no instante t, $[h_{t-1}, x_t]$ denota a concatenação do estado interno no instante anterior $t-1$ e o dado de entrada no instante t. Já W_h e W_{hy} denotam as matrizes de pesos da camada oculta e camada de saída, respectivamente, enquanto que b_h e b_{hy}, denotam os vetores de termos *bias* da camada oculta e camada de saída, respectivamente. A função tangente hiperbólica é utilizada como função de ativação.

Figura 18: Camada oculta de uma rede neural recorrente padrão



2.3.2.1 Treinamento de redes neurais recorrentes

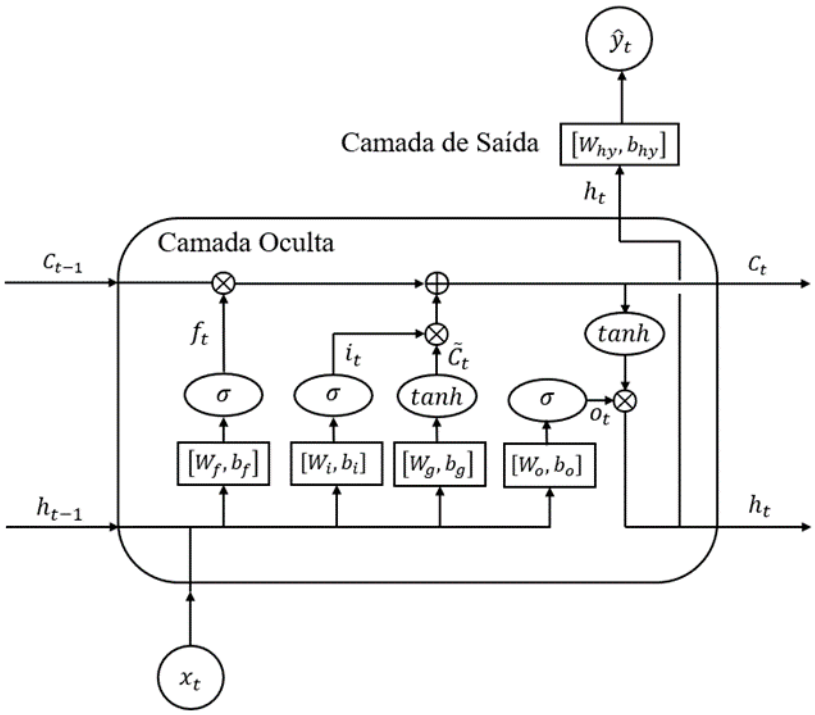
O processo de treinamento de uma rede neural recorrente é semelhante a de uma rede neural *feedforward*, porém ao invés de utilizar o algoritmo *back-propagation* para o cálculo dos gradientes, o algoritmo *back-propagation through time* (BPTT) (WILLIAMS; ZIPSER, 1995) é utilizado, o qual consiste em uma variação específica do algoritmo *back-propagation* para redes neurais recorrentes que leva em conta as conexões de retroalimentação para o cálculo dos gradientes.

Na teoria as redes neurais recorrentes padrões deveriam ser capazes de capturar dependências a longo prazo entre dados sequenciais, entretanto isso não ocorre na prática. Um problema comum encontrado durante o treinamento de redes neurais recorrentes é o problema de dissipação de gradientes (*vanishing gradient*) em que os valores de gradientes propagados ao longo das conexões de retroalimentação da rede se tornam extremamente pequenos, impossibilitando um aprendizado adequado de dependências de longo prazo pela a rede neural (GRAVES, 2012). Nesse contexto, variações das redes neurais recorrentes, tais como as redes *Long Short-Term Memory* (LSTM) (HOCHREITER; SCHMIDHUBER, 1997), têm ganhado destaque pelo fato de utilizarem variações em sua arquitetura que evitam o problema de dissipação de gradiente durante o processo de treinamento (CHUNG et al., 2014).

2.3.2.2 Redes neurais recorrentes LSTM

Uma rede neural *Long Short-Term Memory* (LSTM) consiste em um tipo específico de rede neural recorrente que utiliza portões (*gates*) de memória em sua arquitetura com o objetivo de mitigar o problema de dissipação de gradiente durante seu treinamento (HOCHREITER; SCHMIDHUBER, 1997). Diferentemente de uma rede neural recorrente padrão onde a camada oculta utiliza apenas um conjunto de neurônios (representados pelos parâmetros $[W_h, b_h]$ na Figura 18, no caso de uma rede LSTM, tal como ilustrado na Figura 19, a camada oculta utiliza quatro conjuntos de neurônios representados pelos parâmetros $[W_f, b_f]$, $[W_i, b_i]$, $[W_g, b_g]$ e $[W_o, b_o]$. Esses conjuntos de neurônios são utilizados basicamente na proteção e controle da célula de estado (C_t) da rede LSTM, representada pela linha horizontal superior na camada oculta ilustrada na Figura 19, a qual pode ser vista como uma memória de longo prazo da rede neural que tem como função transmitir informações obtidas em dados já analisados para o processamento de novos dados.

Figura 19: Camada oculta de uma rede neural LSTM



Para explicar a operação de uma rede neural recorrente LSTM no processamento de um dado de entrada x_t é necessário descrever a função de cada um dos conjuntos neurônios presentes em sua camada oculta (OLAH, 2015). O conjunto de neurônios representados pelos parâmetros $[W_f, b_f]$ na Figura 19 é chamado de portão de esquecimento (*forget gate*) e ele tem como função decidir quais informações devem ser descartadas da célula de estado C_t. Para isso, a seguinte equação matemática é utilizada:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.22}$$

Onde W_f e b_f denotam os pesos e termos *bias* dos neurônios desse portão, e $\sigma(\cdot)$ representa especificamente nesse caso a função de ativação sigmoide.

O conjunto de neurônios representados pelos parâmetros $[W_i, b_i]$, por sua vez, é denominado como portão de entrada (*input gate*) e ele tem como de função controlar quais informações devem ser armazenadas na célula de estado. Para realizar esse controle, a seguinte equação é utilizada:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.23}$$

Onde W_i e b_i denotam os pesos e termos *bias* dos neurônios desse portão.

Já o conjunto de neurônios representado pelos parâmetros $[W_g, b_g]$, não é denominado como um portão, porém ele é utilizado junto ao portão de entrada fornecendo possíveis novas informações, representadas pelo termo $\tilde{C}_t$, para atualização da célula de estado. O termo $\tilde{C}_t$ é calculado pela seguinte equação:

$$\tilde{C}_t = \tanh(W_g \cdot [h_{t-1}, x_t] + b_g) \tag{2.24}$$

Onde W_g e b_g denotam os pesos e termos *bias* dos neurônios desse conjunto, e $tanh(\cdot)$ representa a função de ativação tangente hiperbólica.

Com os valores de f_t, i_t e $\tilde{C}_t$ calculados, a célula pode então ser atualizada por meio da seguinte equação:

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t \tag{2.25}$$

Onde $\otimes$ denota um produto de Hadamard.

Por fim, o conjunto de neurônios representado pelos parâmetros $[W_o, b_o]$ é chamado de portão de saída e ele como função decidir quais informações devem ser disponibilizadas para a camada de saída e estado interno da rede neural. Para isso, a seguinte equação é utilizada:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.26}$$

Onde W_o e b_o denotam os pesos e termos *bias* dos neurônios desse portão.

O estado interno oculto h_t é calculado então utilizando o_t e informações fornecidas pela célula de estado atualizada C_t:

$$h_t = o_t \otimes \tanh(C_t) \tag{2.27}$$

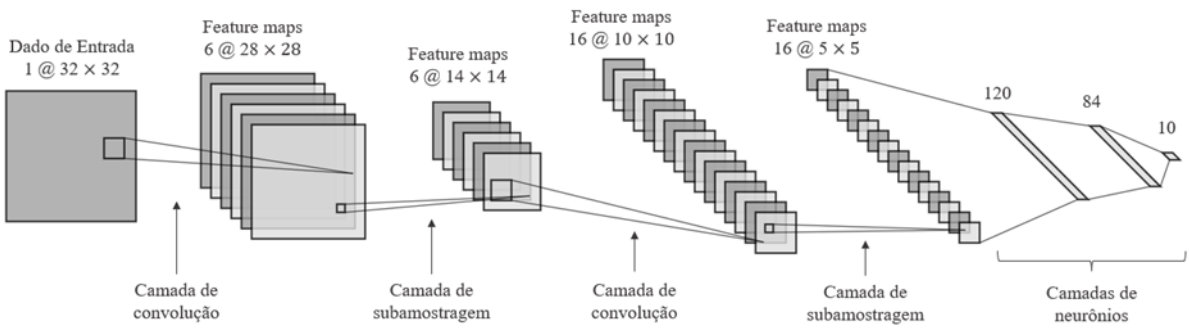
E a resposta da rede neural $\hat{y}_t$ para o dado de entrada x_t é calculado pela equação:

$$\hat{y}_t = W_{hy} h_t + b_{hy} \tag{2.28}$$

2.3.3 Redes neurais convolucionais

As redes neurais convolucionais consistem em uma variação das redes neurais *feedforward* especializada no processamento de dados com topologia em formato de *grid*, tais como as imagens, as quais podem ser vistas como um *grid* de *pixels* (GOODFELLOW; BENGIO; COURVILLE, 2016). Por essa razão, esses modelos são largamente utilizados em problemas de classificação e reconhecimento de imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Tal como ilustrado na Figura 20, uma típica rede neural convolucional é normalmente composta por camadas convolucionais, seguida de camadas de subamostragem (também chamadas de camadas de *pooling*) e terminadas com camadas de neurônios interconectados (tal como as camadas de redes neurais *feedforward*).

Figura 20: Exemplo de rede neural convolucional



Fonte: Gerado utilizando a ferramenta NN-SVG (LENAIL, 2019)

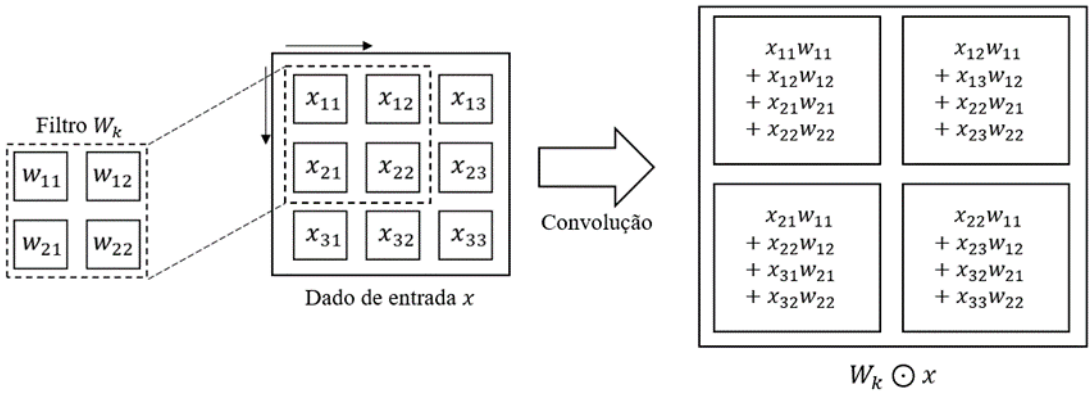
Em uma rede neural convolucional, as camadas convolucionais recebem como entrada dados x arranjados em três dimensões, $m \times m \times r$, onde m denota a altura e espessura do dado, e r se refere a sua profundidade (ou ao seu número de canais @). Cada camada convolucional é composta por múltiplos filtros (também conhecidos como *kernels*) com dimensões $n \times n \times q$, onde n deve ser sempre menor que m e q pode ser igual ou inferior a r. Esses filtros são utilizados para mapear os dados de entrada da camada utilizando operações de convolução. No caso das redes neurais convolucionais, a operação de convolução consiste basicamente na soma dos elementos da matriz resultante do produto de Hadamard entre os parâmetros pesos W_k que compõe um filtro k e partições x_i do dado de

entrada x. Tal como no exemplo bidimensional da Figura 21, na operação de convolução o filtro k percorre todo o dado de entrada da camada executando operações de convolução para diferentes partições do dado, utilizando os mesmos parâmetros pesos W_k para todas as operações. Os resultados das operações de convolução são somados a um termo *bias* b_k e submetidos a uma função de ativação $\sigma(\cdot)$, a qual gera como resposta um *feature map* (mapa de características):

$$f_k = \sigma(W_k \odot x + b_k) \tag{2.29}$$

Onde o operador $\odot$ denota a operação de convolução do filtro para o dado de entrada x. Todos os filtros k que compõe a camada de convolução atuam na mesma forma, gerando cada um deles um *feature map* f_k diferente.

Figura 21: Exemplo de rede neural convolucional



Assim que são gerados, cada *feature map* é fornecido para uma camada de subamostragem para serem submetidos a operações de *pooling*. As operações de *pooling* utilizadas nas camadas de subamostragem também utilizam filtros que percorrem os *feature maps* e realizam operações de redução do número de parâmetros para cada partição percorrida. Por exemplo, no caso de operações de *average pooling*, para cada partição percorrida, o filtro calcula média dos elementos do fragmento (Figura 23). Já no caso de operações de *max pooling*, para cada partição percorrida, o filtro retorna o elemento de maior valor na partição (Figura 22). As camadas de subamostragem tem como função reduzir o número de parâmetros treináveis da rede e consequentemente diminuir a complexidade de seu processo de treinamento.

Figura 22: Exemplo de operação *average pooling*

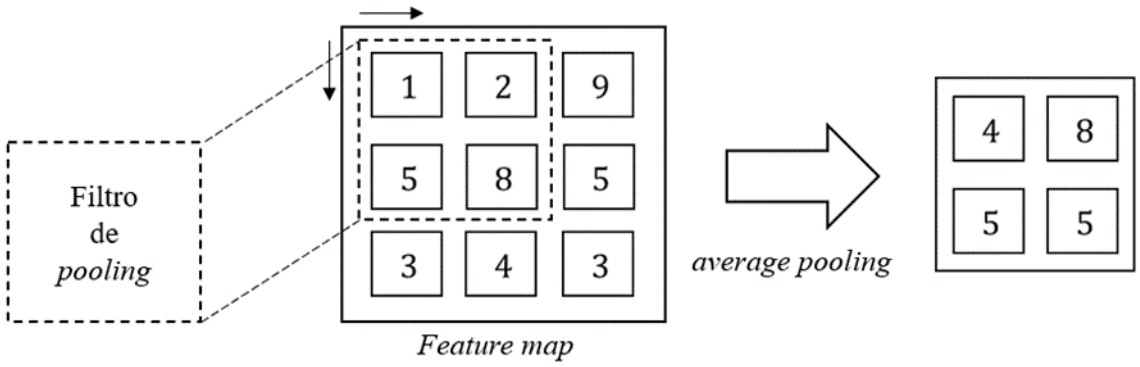
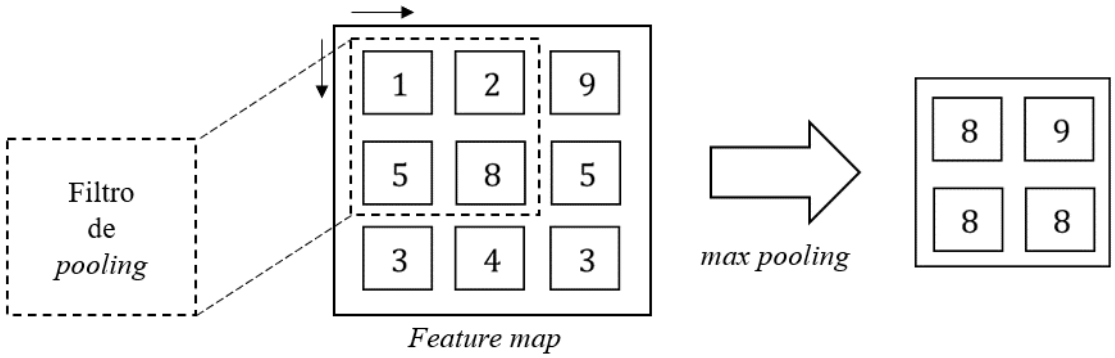


Figura 23: Exemplo de operação *max pooling*



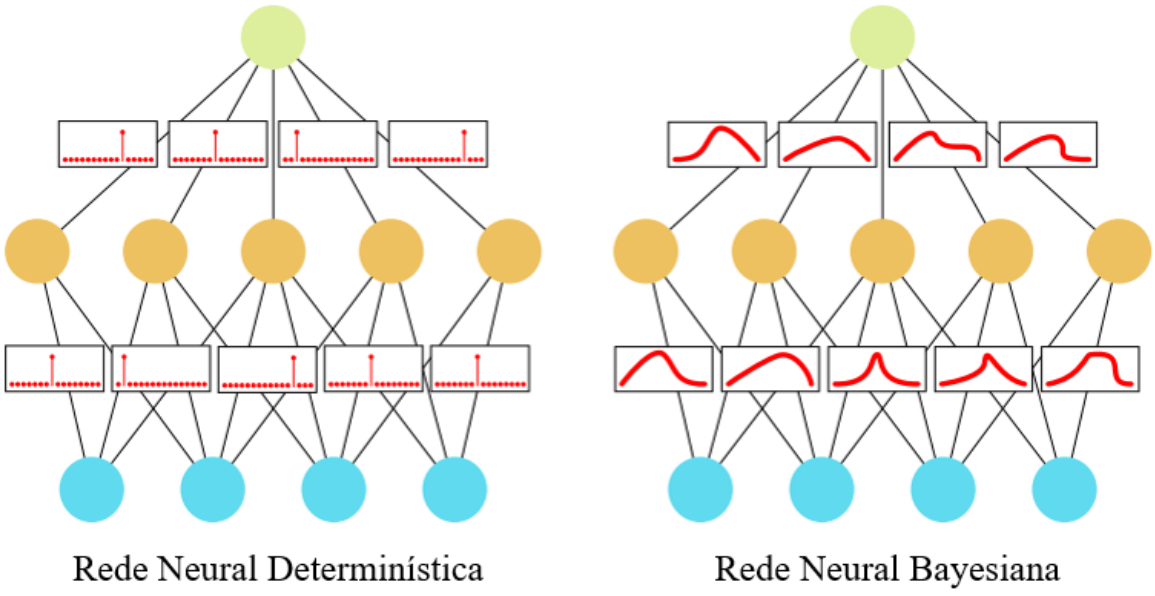
Já as camadas de neurônios que terminam as redes neurais convolucionais tem como função utilizar todas as informações capturadas nas camadas anteriores para geração da resposta da rede neural, como por exemplo *scores* de classificação ou valores de regressão. O processo de treinamento para redes neurais convolucionais é semelhante ao utilizado em redes neurais *feedforward*, sendo baseado também nos algoritmos *back-propagation* e gradiente descendente. No entanto, em comparação com as redes neurais *feedforward*, cada camada de uma rede neural convolucional possui um número muito menor de parâmetros, o que na prática é vantajoso para seu processo de treinamento.

3 REDES NEURAIS BAYESIANAS

3.1 Definição

As redes neurais bayesianas consistem em redes neurais artificiais estocásticas que são treinadas utilizando inferência bayesiana. Redes neurais artificiais estocásticas consistem em modelos de rede neurais que possuem componentes estocásticos em sua arquitetura. Como pode ser observado na Figura 24, no caso das redes neurais bayesianas seus parâmetros são considerados estocásticos e são descritos por distribuições probabilísticas. Ou seja, diferentemente das tradicionais redes neurais determinísticas, nas quais cada parâmetro assume apenas um valor numérico pontual, nas redes neurais bayesianas cada parâmetro é descrito por uma distribuição probabilística de valores prováveis para ele. Dessa forma, ao treinar uma rede neural bayesiana, ao invés de treinar um único modelo, um conjunto de modelos é treinado e suas previsões são agregadas em distribuições probabilísticas preditivas. Essas características das redes neurais bayesianas as tornam mais robustas contra o problema de sobreajuste e as permitem fornecer estimativas de suas incertezas de forma direta, o que são vantagens comparado às tradicionais redes neurais artificiais (GAL, 2016; JOSPIN et al., 2022).

Figura 24: Comparação entre uma rede neural determinística e uma rede neural bayesiana



Fonte: Adaptado de Jospin et al. (2022)

3.2 Formulação

O primeiro passo para formulação de uma rede neural bayesiana, é a escolha do tipo de arquitetura da rede neural. Os diferentes tipos de redes neurais artificiais, tais como rede neurais *feedforward*, convolucionais e recorrentes, podem ser utilizados para formulação de uma rede neural bayesiana. Sendo assim, supondo por exemplo a escolha da formulação de uma rede neural *feedforward* bayesiana, nesse primeiro passo seria definido o número de camadas que a rede neural terá, o número de neurônios em cada camada, assim como as funções de ativação que serão utilizadas. Após esse primeiro passo, supondo que o modelo de rede neural escolhido seja representado pela função $\hat{y} = f^{\theta}(x)$, onde θ denota o conjunto de todos os seus parâmetros, o passo seguinte é escolha de uma distribuição *a priori* $p(\theta)$ sobre o espaço dos parâmetros θ e uma distribuição de verossimilhança $p(y|x, \theta)$.

A distribuição *a priori* $p(\theta)$ representa uma convicção inicial sobre quais são os valores mais prováveis dos parâmetros θ que geraram os dados de treinamento que será utilizado, antes que qualquer amostra de dado do conjunto seja analisado. A definição de uma distribuição *a priori* $p(\theta)$ para um modelo tal como uma rede neural bayesiana, no entanto, normalmente não é uma tarefa intuitiva. Isso se deve ao fato de não ser exatamente explícito como modelos com um número grande de parâmetros e arquiteturas complexas, tais como as redes neurais, generalizarão para uma determinada parametrização. Sendo assim, na prática, distribuições normais gaussianas são comumente utilizadas como distribuições *a priori* para redes neurais bayesianas (GAL, 2016; JOSPIN et al., 2022). Elas funcionam relativamente na prática, mesmo não existindo argumento teórico que o torne melhor uso dessas distribuições em relação ao uso de qualquer outro tipo de distribuição. Por essa razão, distribuições normais gaussianas são utilizadas como como distribuições *a priori* das redes neurais bayesianas implementadas para esse trabalho. Métodos mais robustos para definição de distribuições *a priori* para redes neurais bayesianas tem sido desenvolvidos (JOSPIN et al., 2022), entretanto, o uso desses métodos não será abordado nesse trabalho.

Já a distribuição de verossimilhança $p(y|x, \theta)$ consiste no modelo probabilístico por qual as entradas x geram as saídas y, dado uma configuração dos parâmetros θ. No caso da utilização da rede neural para problemas de classificação, por exemplo, o modelo *softmax* é normalmente utilizado (GAL, 2016):

$$p(y = k|x, \theta) = \frac{\exp(f_k^{\theta}(x)}{\sum_{k'} \exp(f_{k'}^{\theta}(x))} \tag{3.1}$$

Enquanto que no caso da utilização da rede neural para problemas de regressão, o modelo gaussiano é normalmente utilizado (GAL, 2016):

$$p(y|x,\theta) = \mathcal{N}(y; f^{\theta}(x), \sigma) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(f^{\theta}(x) - y)^2}{2\sigma^2}\right) \tag{3.2}$$

3.3 Treinamento

Dado um conjunto de dados de treinamento rotulado $D = \{(x_i, y_i)\}_{i=1}^{N}$, com dados de entradas representados por $X = \{x_1, \ldots, x_N\}$ e seus respectivos valores de saída alvos representados $Y = \{y_1, \ldots, y_N\}$, uma rede neural bayesiana $\hat{y} = f^{\theta}(x)$ é treinada com o objetivo de determinar os valores mais prováveis dos parâmetros θ de terem gerado os dados D. Ou seja, busca-se determinar as distribuições probabilísticas dos valores mais prováveis dos parâmetros θ que possibilitam que a rede neural, ao processar os dados de entrada X, gere os dados de saída alvos Y. Para isso, um método de treinamento baseado em um processo de inferência bayesiana é utilizado. Nesse processo, dado os conjuntos de dados X e Y, é realizado o cálculo da distribuição *a posteriori* $p(\theta|X,Y)$ sobre o espaço do parâmetros θ, a qual descreve quais são os valores mais prováveis dos parâmetros que geraram os dados de treinamento, de acordo com a combinação de informações prévias sobre os parâmetros fornecidos pela distribuição *a priori* $p(\theta)$ e de informações sobre os parâmetros fornecidas pelos dados. Ou seja, a distribuição *a posteriori* consiste basicamente em uma atualização da distribuição *a priori* $p(\theta)$ após análise dos dados.

O cálculo da distribuição *a posteriori* $p(\theta|X,Y)$, é realizada basicamente por meio da aplicação do teorema de Bayes utilizando a distribuição *a priori* $p(\theta)$, a distribuição de verossimilhança $p(Y|X,\theta)$ e um termo de normalização $p(Y|X)$:

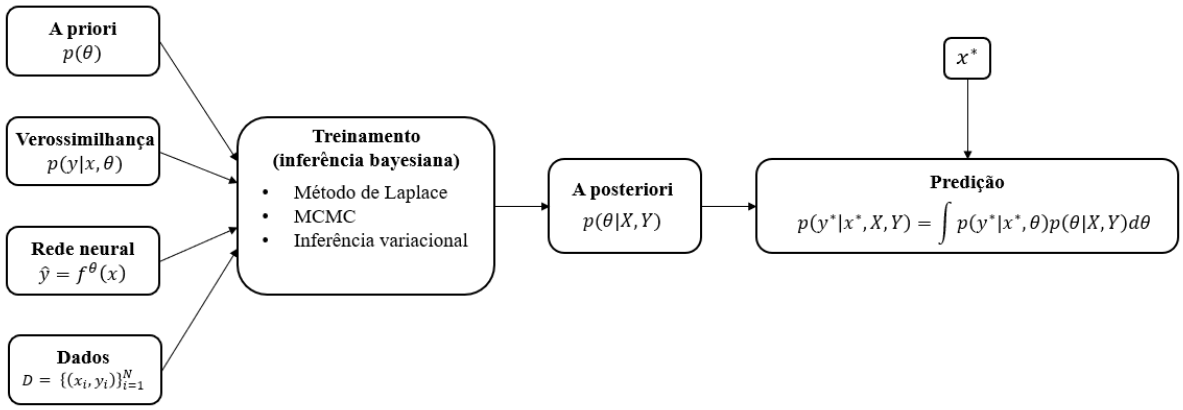
$$p(\theta|X,Y) = \frac{p(Y|X,\theta)p(\theta)}{p(Y|X)} \tag{3.3}$$

Com o cálculo da distribuição *a posteriori* $p(\theta|X,Y)$, o processo de treinamento da rede neural bayesiana é finalizado, habilitando-a então para realizar predições. No caso das redes neurais bayesianas, dado um novo dado de entrada x^*, o cálculo de seu respectivo dado de saída y^* é realizado por meio da seguinte integral:

$$p(y^*|x^*,X,Y) = \int p(y^*|x^*,\theta)p(\theta|X,Y)d\theta \tag{3.4}$$

Todo o processo de formulação, treinamento e utilização de uma rede neural bayesiana para predição é esquematizado na Figura 25.

Figura 25: Processo de formulação, treinamento e utilização de uma rede neural bayesiana



Diferentemente dos modelos de redes neurais determinística, em que as predições assumem valores numéricos pontuais, nas redes neurais bayesianas suas predições são descritas por distribuições probabilísticas preditivas $p(y^*|x^*, X, Y)$ que descrevem quais são os valores mais prováveis de y^*, dado x^* e os conjuntos de dados X e Y, e que consequentemente quantificam as incertezas das predições. Além disso, de acordo com Jospin et al. (2022), a maioria das técnicas de regularização utilizadas para o treinamento redes neurais tradicionais podem ser entendidos, sob uma perspectiva bayesiana, como a definição de uma distribuição *a priori*, e por essa razão o processo de treinamento das redes neurais bayesianas é reconhecido por originalmente já possuir mecanismos contra o problema de sobreajuste.

Apesar do processo de treinamento das redes neurais bayesianas ser teoricamente factível, na prática as redes neurais bayesianas são bem difíceis de serem treinadas (GAL, 2016). Um grande problema no processo de inferência bayesiana é o fato do cálculo da distribuição *a posteriori* $p(\theta|X, Y)$ exata não ser analiticamente possível para modelos que contenham muitos parâmetros, tal como as redes neurais. Um componente chave no cálculo da distribuição *a posteriori* é o termo de normalização $p(Y|X)$, o qual é calculado pela integral:

$$p(Y|X) = \int p(Y|X, \theta)p(\theta)d\theta \tag{3.5}$$

O cálculo desse termo exige a marginalização da distribuição de verossimilhança sobre todos os valores possíveis para θ, o que só é possível de ser realizado analiticamente no caso de modelos bem simples com um número limitado de parâmetros.

Sendo assim, dado essa dificuldade do cálculo da distribuição $p(\theta|X, Y)$ exata, na prática, uma estratégia muito utilizada é o uso de métodos de Monte Carlo para realizar a

amostragem da distribuição $p(\theta|X,Y)$ e utilizar as amostras geradas para pelo menos obter as principais informações da distribuição $p(y^*|x^*,X,Y)$ quando a rede neural bayesiana for realizar uma predição.

Os métodos de Monte Carlo consistem em técnicas computacionais que permitem a aproximação de integrais que não podem ser calculadas de forma exata por meio do uso amostras geradas a partir de uma distribuição probabilística (GOODFELLOW; BENGIO; COURVILLE, 2016). Supondo, por exemplo, a integral:

$$s = \int f(u)p(u)du \tag{3.6}$$

Onde $f(u)$ é uma função e $p(u)$ é uma distribuição probabilística da variável randômica u. Utilizando métodos de Monte Carlo, ao realizar a amostragem de T amostras $u_1, \ldots, u_T$ a partir da distribuição $p(u)$, é provado que essa integral pode ser aproximada pelo cálculo da média:

$$s = \int f(u)p(u)du \approx \frac{1}{T}\sum_{t=1}^{T} f(u_t) \tag{3.7}$$

No caso das redes neurais bayesianas, os métodos de Monte Carlo podem ser utilizados para obter uma aproximação da média $\hat{y}^*$ da distribuição preditiva $p(y^*|x^*,X,Y)$, a qual é calculada pela integral:

$$\tilde{y}^* = \int f^{\theta}(x^*)p(\theta|X,Y)d\theta \tag{3.8}$$

Supondo, por exemplo, a geração de T amostras $\theta_1, \ldots, \theta_T$ a partir da distribuição *a posteriori* $p(\theta|X,Y)$, a média $\tilde{y}^*$ de $p(y^*|x^*,X,Y)$ pode ser aproximada pela média das predições $f^{\theta_1}(x^*), \ldots, f^{\theta_T}(x^*)$:

$$\tilde{y}^* = \int f^{\theta}(x^*)p(\theta|X,Y)d\theta \approx \frac{1}{T}\sum_{t=1}^{T} f^{\theta_t}(x^*) \tag{3.9}$$

Além disso, utilizando as amostras geradas, uma estimativa da incerteza relativa à distribuição $p(y^*|x^*,X,Y)$ pode ser obtida pela expressão:

$$\Sigma^* \approx \frac{1}{T}\sum_{t=1}^{T}(f^{\theta_t}(x^*) - \tilde{y}^*)(f^{\theta_t}(x^*) - \tilde{y}^*)^{tr} \tag{3.10}$$

No entanto, um problema dessa estratégia é que a realização amostragem da distribuição *a posteriori* $p(\theta|X,Y)$ é bem difícil. Por essa razão, para utilizar essa estratégia é necessário o uso de métodos de aproximação de inferência bayesiana que facilitam realizar amostragem da distribuição *a posteriori*.

3.4 Métodos de inferência bayesiana aproximada

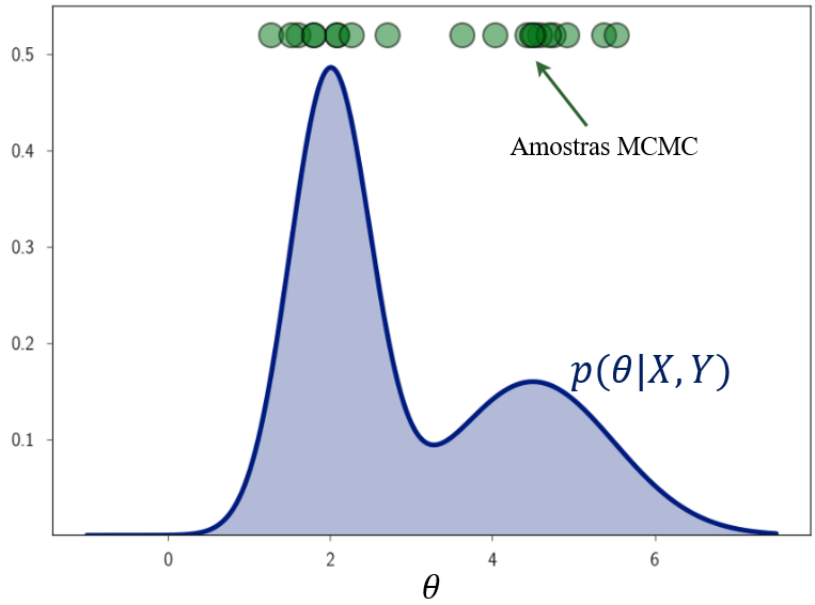
Desde a introdução do conceito de rede neural bayesiana nos anos 90 por MacKay (1992) e Neal (1995), diversos métodos de inferência bayesiana aproximada que facilitam realizar amostragem da distribuição *a posteriori* de redes neurais bayesianas foram propostos. Em sua maioria, esses métodos são baseados em três dos principais métodos de inferência bayesiana aproximada: os métodos de Monte Carlo via cadeias de Markov, o método de aproximação de Laplace e o método de inferência variacional (MADDOX et al., 2019).

3.4.1 Métodos de Monte Carlo via Cadeias de Markov

Os algoritmos de Monte Carlo via Cadeias de Markov (*Markov Chain Monte Carlo* - MCMC) consistem em uma família de algoritmos que usam cadeias de Markov para realizar estimativas de Monte Carlo (GOODFELLOW; BENGIO; COURVILLE, 2016). No contexto das redes neurais bayesianas, esses algoritmos são considerados uma solução para realizar a amostragem da distribuição textita posteriori $p(\theta|X, Y)$ exata (JOSPIN et al., 2022).

Supondo uma distribuição *a posteriori* $p(\theta|X, Y)$ da qual se deseja realizar a amostragem, os algoritmos de MCMC atuam basicamente na construção de uma cadeia de Markov, a qual consiste em uma sequência de amostras randômicas θ_t em que cada amostra θ_i probabilisticamente só depende da amostra anterior θ_{t-1} para ser gerada. O intuito dos algoritmos é que o processo de construção da cadeia de Markov seja desenvolvido até que as amostras da sequência eventualmente se distribuam de acordo com a distribuição *a posteriori* $p(\theta|X, Y)$ da qual se deseja amostrar. Ou seja, o objetivo final dos algoritmos é que a cadeia de Markov convirja para a distribuição *a posteriori* $p(\theta|X, Y)$ e que as amostras geradas pela cadeia possam ser utilizadas como amostras dessa distribuição. Na maioria dos algoritmos MCMC, demora-se um tempo para que a cadeia de Markov convirja para distribuição desejada, e por essa razão algumas amostras iniciais da cadeia devem ser descartadas. Além disso, pelo de amostras sucessivas da sequência serem altamente correlacionadas, é recomendado evitar um número grande de amostras sucessivas sejam utilizadas para garantir um certo nível de independência entre as amostras. Um exemplo ilustrativo de como seriam distribuídas as amostras de uma distribuição *a posteriori* $p(\theta|X, Y)$ geradas utilizando um algoritmo MCMC é apresentada na Figura 26.

Figura 26: Exemplo ilustrativo de amostras geradas via MCMC



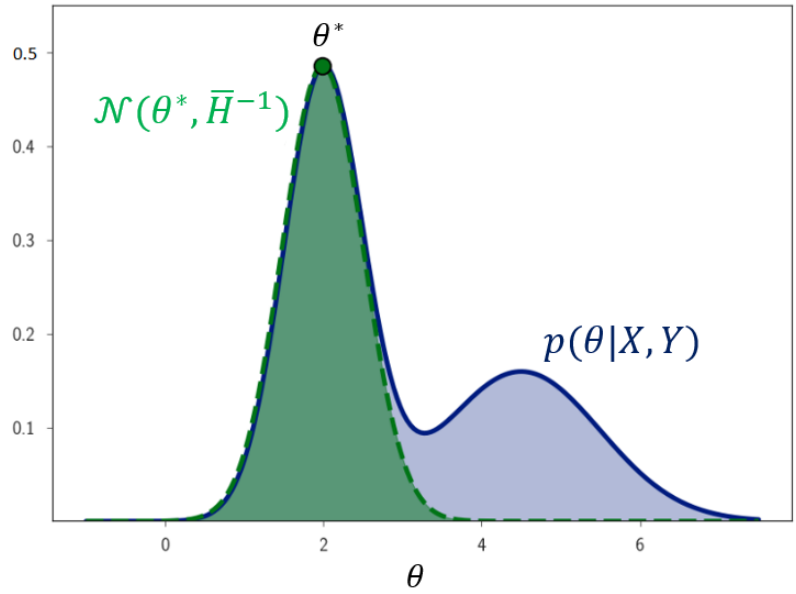
Fonte: Adaptado de Maddox (2020)

Existem diversos tipos de algoritmos de MCMC, entre os quais deve-se destacar o algoritmo de Metropolis-Hastings e o algoritmo de amostragem de Gibbs. Entretanto, no caso das redes neurais bayesianas, o algoritmo de Metropolis-Hastings é o mais relevante, pois ele permite realizar amostragem da distribuição textita posteriori $p(\theta|X, Y)$ sem precisar calculá-la de forma exata (JOSPIN et al., 2022). O uso de algoritmos de MCMC para métodos de aproximação de inferência bayesiana em redes neurais bayesianas foi primeiramente introduzido no trabalho de Neal (1995), o qual propõe o uso do método *Hamiltonian Monte Carlo* (HMC) para amostragem da distribuição *a posteriori*. O método HMC consiste em uma variação do algoritmo de Metropolis-Hastings que utiliza conceitos de dinâmica hamiltoniana para obtenção de amostras de distribuições *a posteriori* de forma mais eficiente. Entretanto, ele é um método difícil de ser aplicado, pois é difícil ser ajustado e não se adequa bem para conjuntos de dados grandes pelo fato de exigir o cálculo de gradientes do modelo de verossimilhança avaliada em todo o conjunto de dados (GAL, 2016). Sendo assim, como uma extensão do método HMC, o método *Stochastic Gradient Hamiltonian Monte Carlo* (SGHMC) foi proposto por Chen, Fox e Guestrin (2014) com intuito de solucionar o problema de escalabilidade do método HMC. O método SGHMC permite que gradientes estocásticos sejam usados no processo de inferência bayesiana, possibilitando que conjuntos de dados grandes sejam processados. No entanto, apesar de resolver o problema de escalabilidade do método HMC, o método SGHMC ainda é difícil de ser ajustado, o que ainda dificulta sua implementação na prática (MADDOX et al., 2019).

3.4.2 Método de aproximação de Laplace

O método de aproximação de Laplace é um método que permite aproximar uma distribuição probabilística complexa a uma distribuição gaussiana utilizando uma expansão de Taylor de segunda ordem (MACKAY, 2003). Tal como ilustrado na Figura 27, no caso das redes neurais bayesianas, esse método é utilizado para aproximar a distribuição *a posteriori* a uma distribuição gaussiana, permitindo assim que amostras aproximadas da distribuição *a posteriori* exata sejam facilmente geradas, uma vez que o processo de amostragem de uma distribuição gaussiana é mais simples.

Figura 27: Exemplo ilustrativo do método de aproximação de Laplace



Fonte: Adaptado de Maddox (2020)

Supondo que se deseje obter uma aproximação a distribuição *a posteriori* $p(\theta|X,Y)$, a aplicação do método de Laplace consiste basicamente no ajuste de uma distribuição gaussiana centrada na moda da distribuição $p(\theta|X,Y)$ por meio da obtenção da expansão de Taylor de segunda ordem em torno da moda. Para isso, primeiramente deve-se obter a moda da distribuição *a posteriori* $p(\theta|X,Y)$. A moda da distribuição $p(\theta|X,Y)$, denotada por θ^*, pode ser obtida pelo cálculo da estimativa da máxima *a posteriori*, o qual é realizado pela solução do seguinte problema de otimização:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \frac{p(Y|X,\theta)p(\theta)}{p(Y|X)} \tag{3.11}$$

Pelo fato do termo $p(Y|X)$ não ser função de θ, esse problema pode ser simplificado por:

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmax}}\ p(Y|X,\theta)p(\theta) \\ &= \underset{\theta}{\operatorname{argmax}}\ \log p(Y|X,\theta) + \log p(\theta)\end{aligned} \tag{3.12}$$

Com a moda θ^* calculada, ela então é utilizada para aproximar o logaritmo da distribuição *posteriori* por meio do uso da expansão de Taylor de segunda ordem em torno da moda θ^*:

$$\log p(\theta|X,Y) \approx \log p(\theta^*|X,Y) - \frac{1}{2}(\theta-\theta^*)^T\bar{H}(\theta-\theta^*) \tag{3.13}$$

Onde $\bar{H} = \mathbb{E}[H]$ representa o valor esperado da matriz Hessiana calculado em θ^*. Deve-se destacar que o termo de primeira ordem é omitido na Equação 3.13 pelo fato de a expansão ser realizada em torno de θ^*, onde o gradiente é nulo.

Calculando o exponencial da equação 3.13 nota-se que o lado direito da equação adquire a forma de uma função de densidade gaussiana com média θ^* e covariância $\bar{H}^{-1}$:

$$p(\theta|X,Y) \approx p(\theta^*|X,Y)\exp\left(-\frac{1}{2}(\theta-\theta^*)^T\bar{H}(\theta-\theta^*)\right) \approx \mathcal{N}(\theta;\theta^*,\bar{H}^{-1}) \tag{3.14}$$

Portanto, a aplicação do método de aproximação de Laplace no caso de redes neurais bayesianas, permite que a distribuição *a posteriori* seja aproximada por uma distribuição gaussiana $\mathcal{N}(\theta^*,\bar{H}^{-1})$ e que amostras geradas por essa distribuição sejam consideradas amostras aproximadas da distribuição *a posteriori* $p(\theta|X,Y)$ exata:

$$\theta \sim \mathcal{N}(\theta;\theta^*,\bar{H}^{-1}) \tag{3.15}$$

Entretanto, pelo fato de os modelos de redes neurais apresentarem um elevado número de parâmetros, o cálculo da inversa da matriz hessiana $\bar{H}$ se torna uma tarefa extremamente custosa para esses modelos, sendo este um fator limitante para aplicação do método de Laplace para aproximação de inferência em redes neurais bayesianas (RITTER; BOTEV; BARBER, 2018).

Um dos primeiros trabalhos a sugerir o uso do método de Laplace para aproximação de inferência em redes neurais foi MacKay (1992), o qual propõe o uso de uma aproximação diagonal da inversa da matriz Hessiana no lugar da inversa da matriz Hessiana exata com intuito de evitar os problemas de custo computacional. Apesar de sua praticabilidade em casos simples redes neurais, o método apresentado por MacKay (1992) sofre com o aumento da complexidade das redes neurais e do volume de dados analisados, não

podendo ser aplicado, portanto, para casos mais complexos de redes neurais (GAL, 2016). De forma a solucionar esse problema de escalabilidade, Ritter, Botev e Barber (2018) propõe o uso do método de Laplace junto de uma aproximação fatorada de Kronecker para matriz hessiana de forma a solucionar o problema de escalabilidade apresentado MacKay (1992). Já Maddox et al. (2019) propõe um método chamado *Stochastic Weight Averaging - Gaussian* (SWAG), o qual não é baseado no método de aproximação de Laplace, mas que igualmente aproxima a distribuição *a posteriori* para uma distribuição gaussiana. Ele basicamente aproxima a distribuição *a posteriori* para uma distribuição gaussiana com a solução do método *Stochastic Weight Averaging* (SWA) como média e uma matriz de covariância diagonal de posto reduzido derivada a partir de múltiplas iterações do gradiente descendente estocástico. Uma crítica distribuição *a posteriori*, no entanto, realizada a esses métodos é que eles são capazes apenas de capturar precisamente uma moda de uma distribuição *a posteriori* e consequentemente não performam bem o caso de distribuições multimodais (JOSPIN et al., 2022).

3.4.3 Método de inferência variacional

O método de inferência variacional consiste em um método de inferência aproximada baseado na obtenção de uma distribuição *a posteriori* aproximada por meio do uso de distribuições variacionais (GAL, 2016). Uma distribuição variacional $q_\varphi(\theta)$ consiste em uma distribuição parametrizada por φ (parâmetros variacionais) e cuja estrutura é fácil de ser avaliada, como por exemplo tal como uma distribuição gaussiana, exponencial, gama, de Bernoulli, entre outras. Desta forma, ao se definir uma distribuição variacional, o método de inferência variacional tem como objetivo solucionar um problema de minimização de divergência de Kullback-Leibler (KL) com respeito à φ, de forma obter a distribuição $q_\varphi(\theta)$ mais próxima possível da verdadeira distribuição *a posteriori*:

$$\min_{\varphi} \mathrm{KL}(q_\varphi(\theta)||p(\theta|X,Y)) = \min_{\varphi} \int q_\varphi(\theta) \log \frac{q_\varphi(\theta)}{p(\theta|X,Y)} d\theta \tag{3.16}$$

Dado que a distribuição *a posteriori* $p(\theta|X,Y)$ exata não pode ser calculada analiticamente, esse problema otimização é normalmente reescrito de forma a eliminar esse termo da equação:

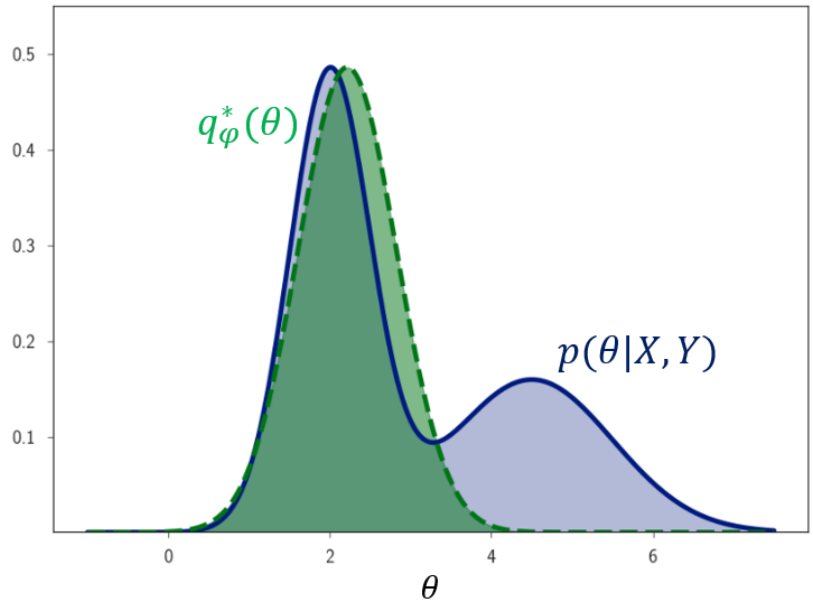
$$\min_{\varphi} \int q_\varphi(\theta) \log \frac{q_\varphi(\theta)}{p(\theta|X,Y)} d\theta = \min_{\varphi} \int q_\varphi(\theta) \log \frac{q_\varphi(\theta) p(Y|X)}{p(Y|X,\theta) p(\theta)} d\theta =$$
$$= \min_{\varphi} \left(- \int q_\varphi(\theta) \log p(Y|X,\theta) d\theta + \int q_\varphi(\theta) \frac{q_\varphi(\theta)}{p(\theta)} d\theta + \int q_\varphi(\theta) \log p(Y|X) d\theta \right) =$$
$$= \min_{\varphi} \left(\mathbb{E}_{q_\varphi(\theta)}(\log p(Y|X,\theta)) + \mathrm{KL}(q_\varphi(\theta)||p(\theta)) + \log p(Y|X) \right) \quad (3.17)$$

Considerando que $\log p(Y|X)$ é um termo que não depende de φ, este termo é excluído do problema de otimização. Sendo assim, o problema de minimização a ser resolvido no método de inferência variacional passa ser descrito por:

$$\min_{\varphi} \mathrm{KL}(q_\varphi(\theta)||p(\theta|X,Y)) =$$
$$\min_{\varphi} \left(-\mathbb{E}_{q_\varphi(\theta)}(\log p(Y|X,\theta)) + \mathrm{KL}(q_\varphi(\theta)||p(\theta)) \right) \quad (3.18)$$

Ao solucionar esse problema de minimização, tal como exemplificado na Figura 28, uma distribuição variacional ótima $q^*_\varphi(\theta)$ é obtida tal que esta pode ser vista como uma aproximação da verdadeira distribuição *a posteriori* ($p(\theta|X,Y) \approx q^*_\varphi(\theta)$).

Figura 28: Exemplo ilustrativo do método de inferência variacional



Fonte: Adaptado de Maddox (2020)

Dado que a distribuição variacional ótima $q^*_\varphi(\theta)$ obtida normalmente possue um processo de amostragem mais simples comparado à distribuição *a posteriori* $p(\theta|X,Y)$ exata, amostras da distribuição $q^*_\varphi(\theta)$ podem ser facilmente geradas e utilizadas como amostras aproximadas da distribuição $p(\theta|X,Y)$:

$$\theta \sim q^*_\varphi(\theta) \tag{3.19}$$

Os trabalhos de Hinton e Camp (1993) e Barber e Bishop (1998) foram um dos primeiros a apresentar o uso de inferência variacional no contexto de redes neurais bayesianas. O trabalho de Hinton e Camp (1993) demonstrou a utilização de uma distribuição gaussiana variacional com matriz covariância diagonal para aproximar a distribuição *a posteriori* $p(\theta|X, Y)$ de uma simples rede neural bayesiana com uma única camada oculta de neurônios, enquanto que o trabalho de Barber e Bishop (1998) propôs um método semelhante, porém que já utilizava uma distribuição gaussiana variacional com matriz covariância completa para aproximar a distribuição *a posteriori*. Entretanto, um problema de ambos os métodos não são praticáveis para modelos de redes neurais complexas que contém um grande número de parâmetros e que são treinadas utilizando grandes conjunto de dados (GAL, 2016). Sendo assim, ao longo dos anos, diversos métodos baseados em inferência variacional que visam solucionar esse problema de escalabilidade foram propostos.

Baseado em Hinton e Camp (1993), Graves (2011) foi o primeiro a propor um método baseado inferência variacional com capacidade de ser aplicado em casos mais elaborados de redes neurais bayesianas. Graves (2011) propõe o ajuste de distribuições gaussianas variacionais sobre parâmetros da rede neural tal como uso de métodos de Monte Carlo durante a solução do problema de minimização de divergência KL. Apesar do método de Graves (2011) ser considerado o primeiro praticável em modelos mais complexos, de acordo com Hernández-Lobato e Adams (2015), sua performance não é totalmente satisfatória na prática. Já Blundell et al. (2015) propõe um método semelhante ao de Graves (2011), porém ele sugere o uso do truque de reparametrização apresentado por Kingma e Welling (2013) para definição dos valores dos parâmetros peso da rede neural. Este método de Blundell et al. (2015), nomeado como *Bayes by Backprop*, melhora a performance comparado à técnica de Graves (2011), porém ao mesmo tempo gera um aumento do número de parâmetros a serem ajustados, o que torna o método custoso computacionalmente. Apesar dessa desvantagem, várias aplicações de redes neurais bayesianas utilizam o *Bayes by Backprop* para aproximação do processo de inferência bayesiana (BRUSAFERRI et al., 2019; DEODATO; BALL; ZHANG, 2020; FORTUNATO; BLUNDELL; VINYALS, 2017). Dentre os métodos de aproximação de inferência bayesiana para redes neurais baseados em inferência variacional, o método de *MC Dropout* (*Monte Carlo Dropout*) de Gal e Ghahramani (2016a) é um dos mais utilizados em aplicações reais de redes neurais bayesianas (GAL; GHAHRAMANI, 2016b; WESTHUIZEN; LASENBY,

2017; GAL; ISLAM; GHAHRAMANI, 2017; SUN et al., 2020), pois ele é um método de inferência bayesiana aproximada de fácil implementação e que não exige alterações na estrutura dos modelos de redes neurais ou nos seus métodos de treinamento (GAL, 2016). O método se baseia basicamente na interpretação da técnica de regularização *dropout* (SRIVASTAVA et al., 2014) como um método de inferência variacional que utiliza uma distribuição de Bernoulli como distribuição *a posteriori* variacional.

3.4.4 Estudo prático dos métodos de inferência bayesiana aproximada

Dado que esse trabalho tem como objetivo apresentar aplicação de modelos de redes neurais bayesianas para solução de problemas reais, nessa seção é apresentado um estudo prático dos dois dos principais métodos de inferência bayesiana aproximada utilizados em aplicações reais de redes neurais bayesianas: o *Bayes by Backprop* e o *MC Dropout*. Para esse estudo prático, uma descrição detalhada sobre esses métodos é apresentada e em seguida são demonstradas as aplicações de ambos os métodos para a solução de um simples problema de regressão utilizando uma rede neural bayesiana. Por fim os resultados das aplicações são apresentados e uma comparação dos métodos é realizada.

3.4.4.1 *Bayes by Backprop*

O *Bayes by Backprop* consiste em um método de inferência bayesiana aproximada que permite a utilização de inferência variacional em rede neurais bayesianas complexas e que é compatível ao algoritmo *backpropagation*. Para isso, ele propõe o uso do truque reparametrização proposto por Kingma e Welling (2013) para definição dos parâmetros peso de uma rede neural como função de parâmetros variacionais relativos a uma distribuição variacional. Assumindo, por exemplo, uma distribuição variacional gaussiana $q_\varphi(\theta)$ utilizada para aproximar a distribuição *a posteriori* $p(\theta|X, Y)$ de uma rede neural bayesiana $f^\theta(x)$, a amostras θ da distribuição $q_\varphi(\theta)$ são geradas por meio das seguintes expressões:

$$\theta = \mu + \sigma \cdot \epsilon \tag{3.20}$$

$$\sigma = \log(1 + \exp(\rho)) \tag{3.21}$$

$$\epsilon \sim \mathcal{N}(0, I) \tag{3.22}$$

Onde $\varphi = (\mu, \rho)$ são parâmetros variacionais da distribuição variacional gaussiana $q_\varphi(\theta)$ e ϵ é uma variável randômica. O termo σ é definido por $\log(1+\exp(\rho))$ para garantir esse parâmetro nunca seja negativo.

Ao configurar a amostragem dos parâmetros θ desta maneira, permite-se que o modelo seja treinado por meio da solução um problema de otimização dos parâmetros $\varphi = (\mu, \rho)$ utilizando o algoritmo *backpropagation*, tornando assim possível que distribuições gaussianas sejam aproximadamente inferidas sobre os parâmetros θ da rede neural bayesiana (BLUNDELL et al., 2015). Para que a otimização dos parâmetros $\varphi = (\mu, \rho)$, utilizando a técnica de amostragem de Monte Carlo, amostras θ são geradas a partir da distribuição variacional $q_\varphi(\theta)$ em cada iteração do processo de treinamento e os gradientes da função custo em relação esses parâmetros são utilizados para o ajuste desses parâmetros. A sequência de passos para o treinamento da rede neural bayesiana por meio do *Bayes by Backprop* é descrita a seguir:

1. Dado uma base de dados $D = (X, Y)$
2. Definir a taxa de aprendizado η
3. Inicializar randomicamente os parâmetros variacionais $\varphi = (\mu, \rho)$
4. Inicializar a distribuição *a priori* $p(\theta)$
5. Repetir:
   (a) Amostrar $\epsilon \sim \mathcal{N}(0, I)$
   (b) Calcular $\theta = \mu + \log(1 + \exp(\rho)) \cdot \epsilon$
   (c) Calcular função custo $J(\theta, \varphi)$:

$$J(\theta, \varphi) = -\log p(Y|X, \theta) + \log \frac{q_\varphi(\theta)}{p(\theta)} \tag{3.23}$$

   (d) Utilizando *backpropagation*, calcular gradiente de $J(\theta, \varphi)$ com respeito a θ:

$$\Delta_\theta = \frac{\partial J(\theta, \varphi)}{\partial \theta} \tag{3.24}$$

   (e) Utilizando *backpropagation*, calcular gradiente de $J(\theta, \varphi)$ com respeito a μ:

$$\Delta_\mu = \frac{\partial J(\theta, \varphi)}{\partial \theta} + \frac{\partial J(\theta, \varphi)}{\partial \mu} \tag{3.25}$$

   (f) Utilizando *backpropagation*, calcular gradiente de $J(\theta, \varphi)$ com respeito a ρ:

$$\Delta_\rho = \frac{\partial J(\theta, \varphi)}{\partial \theta} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial J(\theta, \varphi)}{\partial \rho} \tag{3.26}$$

   (g) Atualizar os parâmetros variacionais μ e ρ

$$\mu \leftarrow \mu - \eta \Delta_\mu \tag{3.27}$$

$$\rho \leftarrow \rho - \eta \Delta_\rho \tag{3.28}$$

6. Até $\varphi = (\mu, \rho)$ convergir

Com relação a distribuição a priori $p(\theta)$, uma distribuição gaussiana pode ser utilizada, porém o método propõe o uso de uma mistura em escala de duas distribuições gaussianas com médias em zero e variâncias diferentes.

Após o treinamento, a distribuição $q_\varphi(\theta)$ é então obtida como uma aproximação para a distribuição *a posteriori* $p(\theta|X, Y)$, e suas amostras podem ser utilizadas, por meio de um método de aproximação de Monte Carlo, para obter estimativas da média e da incerteza de uma distribuição preditiva $p(y^*|x^*, X, Y)$ para um novo dado de entrada x^*, tal como descrito a seguir:

1. Dado um novo dado de entrada x^*
2. Repetir para $t = 1, \ldots, T$:
   (a) Amostrar $\epsilon_t \sim \mathcal{N}(0, I)$
   (b) Calcular $\theta_t = \mu + \log(1 + \exp(\rho)) \cdot \epsilon_t$
   (c) Calcular $f^{\theta_t}(x^*)$
   (d) Armazenar $f^{\theta_t}(x^*)$
3. Utilizando as amostras $f^{\theta_1}(x^*), \ldots, f^{\theta_T}(x^*)$ calcular:

$$\tilde{y}^* = \int f^\theta(x^*) p(\theta|X, Y) d\theta \approx \frac{1}{T} \sum_{t=1}^{T} f^{\theta_t}(x^*) \tag{3.29}$$

$$\Sigma^* \approx \frac{1}{T} \sum_{t=1}^{T} (f^{\theta_t}(x^*) - \tilde{y}^*)(f^{\theta_t}(x^*) - \tilde{y}^*)^{tr} \tag{3.30}$$

O método *Bayes by Backprop* é aplicado para solução de problemas reais, já tendo sido utilizado por exemplo para predição de preço de eletricidade (BRUSAFERRI et al., 2019) e na classificação de imagem de células (DEODATO; BALL; ZHANG, 2020).

3.4.4.2 *MC Dropout*

O *MC Dropout* consiste em um método para de inferência bayesiana aproximada utilizado para redes neurais bayesianas complexas. Esse método utiliza uma nova interpretação da técnica de regularização *dropout* para aproximar o processo de inferência bayesiana em uma rede neural bayesiana. O *dropout* (SRIVASTAVA et al., 2014) consiste

em uma técnica de regularização muito utilizada especialmente em modelos de redes neurais para evitar problemas de sobreajuste. A técnica consiste basicamente na desativação randômica de parte das unidades de processamento de uma rede neural em cada iteração do processo de treinamento.

Gal e Ghahramani (2016b) prova que o treinamento de uma rede neural que utiliza *dropout* é matematicamente equivalente a uma aproximação ao processo gaussiano probabilístico profundo que permite modelar distribuições sobre funções. O trabalho demonstra que a minimização da função custo do treinamento de uma rede neural $f^{\theta}(x)$ regularizada utilizando *dropout* é matematicamente equivalente à minimização da divergência de KL entre distribuição *a posterior* de um processo gaussiano profundo e uma distribuição variacional $q_{\varphi}(\theta)$ cujas amostras são geradas da seguinte forma:

$$z_i \sim Bernoulli(p_i) \tag{3.31}$$

$$\theta_i = \varphi_i \cdot diag(z_i) \tag{3.32}$$

Onde φ_i é um parâmetro variacional que consiste na matriz de parâmetros da rede neural com todas as unidades de processamento ativas, z_i denota uma variável binária gerada a partir de amostragens da distribuição de Bernoulli com probabilidade p_i utilizada para ativar ou desativar unidades da rede neural de forma randômica, e θ_i representa a matriz de parâmetros da rede neural após aplicação do *dropout*.

Dessa forma, de acordo com Gal e Ghahramani (2016b), após o treinamento de uma rede neural regularizada utilizando *dropout*, a distribuição variacional $q_{\varphi}(\theta)$ é considerada uma aproximação da distribuição *a posterior* $p(\theta|X, Y)$ da versão bayesiana da rede neural $f^{\theta}(x)$. A sequência de passos para o treinamento da rede neural regularizada utilizando *dropout* é descrita a seguir:

1. Dado uma base de dados $D = (X, Y)$
2. Definir da taxa de aprendizado η
3. Definir a probabilidade de p_i da distribuição de Bernoulli para cada camada $i = 1, \ldots, L$
4. Inicializar randomicamente os parâmetros variacionais φ
5. Repetir:
   (a) Amostrar $z_i \sim \text{Bernoulli}(p_i)$
   (b) Calcular $\theta_i = \varphi_i \cdot diag(z_i)$

(c) Calcular função custo $J(\theta)$:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} ||f^{\theta}(x_i) - y_i||_p + \alpha||\theta||_2^2 \tag{3.33}$$

(d) Calcular gradiente de $J(\theta)$ com respeito a θ:

$$\Delta_\theta = -\frac{\partial J(\theta)}{\partial \theta} \tag{3.34}$$

(e) Atualizar os parâmetros θ

$$\theta \leftarrow \theta - \eta\Delta_\theta \tag{3.35}$$

6. Até θ convergir

Além disso, de acordo com Gal e Ghahramani (2016b), a aplicação do *dropout* durante a fase de predição dessa rede neural é equivalente ao processo de amostragem da distribuição $q_\varphi(\theta)$ descrito pelas Equações 3.31 e 3.32. Supondo, por exemplo, o processamento de um novo dado de entrada x^* pela rede neural treinada com *dropout*, T repetições do processamento desse dado geram T amostras $\theta_1, \ldots, \theta_T$ da distribuição $q_\varphi(\theta)$, as quais podem então ser utilizadas para obter estimativas da média e da incerteza de uma distribuição de predição $p(y^*|x^*, X, Y)$ para um novo dado de entrada x^*, tal como descrito a seguir:

1. Dado um novo dado de entrada x^*
2. Repetir para $t = 1, \ldots, T$:
   (a) Amostrar $z_t \sim \text{Bernoulli}(p_t)$
   (b) Calcular $\theta_t = \varphi_t \cdot diag(z_t)$
   (c) Calcular $f^{\theta_t}(x^*)$
   (d) Armazenar $f^{\theta_t}(x^*)$
3. Utilizando as amostras $f^{\theta_1}(x^*), \ldots, f^{\theta_T}(x^*)$ calcular:

$$\tilde{y}^* = \int f^{\theta}(x^*)p(\theta|X, Y)d\theta \approx \frac{1}{T}\sum_{t=1}^{T} f^{\theta_t}(x^*) \tag{3.36}$$

$$\Sigma^* \approx \frac{1}{T}\sum_{t=1}^{T}(f^{\theta_t}(x^*) - \tilde{y}^*)(f^{\theta_t}(x^*) - \tilde{y}^*)^{tr} \tag{3.37}$$

Como pode ser notado, o método de *MC Dropout* não exige alterações na estrutura dos modelos de redes neurais ou nos seus métodos tradicionais de treinamento (GAL,

2016) e desse modo ele é considerado um método de fácil implementação. Devido a essas características, o método de *MC Dropout* ele é uma alternativa popular para a inferência bayesiana aproximada de modelos de redes neurais bayesianas complexas aplicadas para solução de problemas reais, tendo já demonstrado bons resultados em problemas tais como de análise de sentimentos (GAL; GHAHRAMANI, 2016b), diagnósticos médicos (WESTHUIZEN; LASENBY, 2017), reconhecimento de imagens (GAL; ISLAM; GHAHRAMANI, 2017) e detecção de falhas em processos industriais (SUN et al., 2020).

3.4.4.3 Experimento

Com objetivo de explorar de forma prática os métodos *Bayes by Backprop* e *MC Dropout*, ambos os métodos são aplicados o treinamento de uma rede neural utilizada para a solução de um de um simples problema de regressão. A solução do problema proposto consiste no treinamento de um modelo por meio de um conjunto de dados de treinamento contendo 50 amostras de entrada x geradas randomicamente a partir do intervalo $[-4, 4]$, e seus correspondentes valores de saída y calculados pela expressão $y = x^3 + \epsilon_n$, onde $\epsilon_n \sim \mathcal{N}(0, 9)$. Os dados do problema são ilustrados na Figura 29. Após treinamento, o objetivo é que o modelo seja capaz de predizer y a partir do valor de x. Para solução do problema, propõe-se o treinamento de duas redes neurais bayesianas simples, cada uma com uma única camada oculta contendo 100 unidades de processamento de regressão, tal como ilustrado na Figura 30. Uma das redes neurais é treinada utilizando o método *Bayes by Backprop* e a outra é treinada utilizando *MC Dropout*. Após o treinamento dessas redes neurais, elas são testadas para a predição dos dados de x no intervalo $[-5, 5]$. Dado que não existem nos principais *frameworks* comerciais da linguagem Python para implementação de redes neurais (tal como o PyTorch e o Tensorflow) módulos já implementados para o treinamento de redes neurais bayesianas utilizando os métodos *Bayes by Backprop* e *MC Dropout*, ambos os métodos que tiveram que ser implementados do zero utilizando ferramentas desses *frameworks* comerciais. Especificamente para esse estudo prático, o *framework* PyTorch foi utilizado. Os códigos de programa utilizados para o treinamento das redes neurais bayesianas utilizando os métodos *Bayes by Backprop* e *MC Dropout* são encontrados no repositório do Github ⟨https://github.com/RyujiGT96/masters-project/tree/master/0-bayesian-neural-network⟩.

Figura 29: Dados de treinamento para solução do problema de regressão

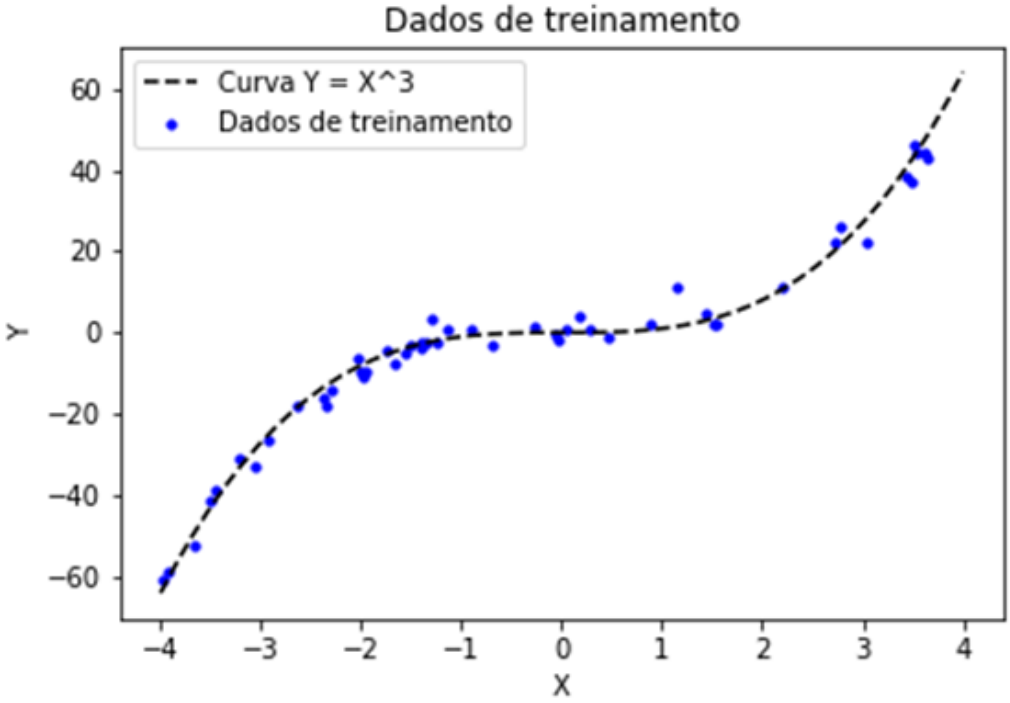
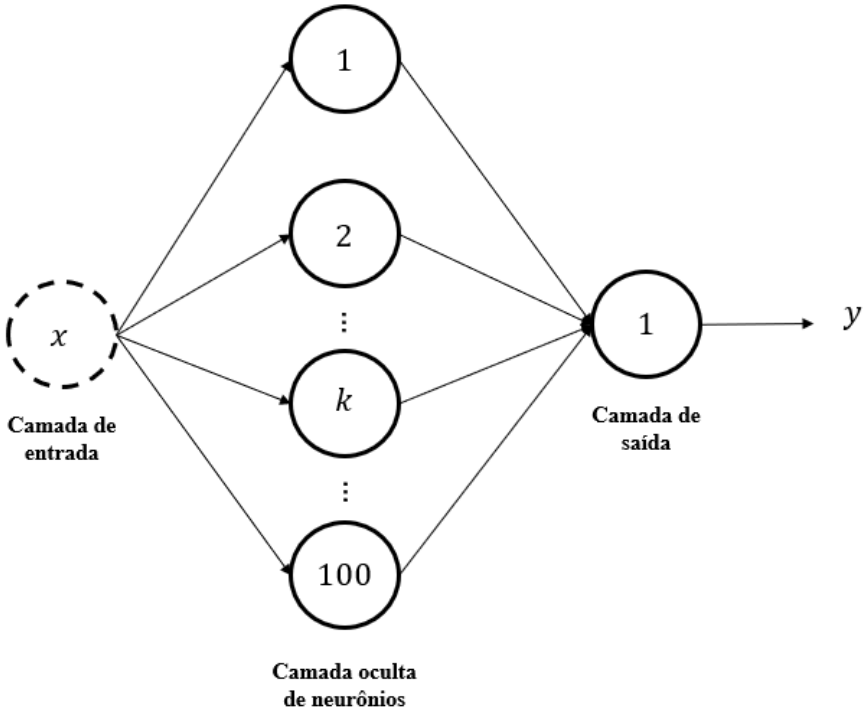


Figura 30: Modelo de rede neural bayesiana treinado



3.4.4.4 Resultados

As soluções obtidas pelas redes neurais bayesianas treinadas pelos métodos *Bayes by Backprop* e *MC Dropout* para o problema de regressão são apresentadas nas Figuras 31 e 32, respectivamente.

Figura 31: Resultado da rede neural treinada via *Bayes by Backprop*

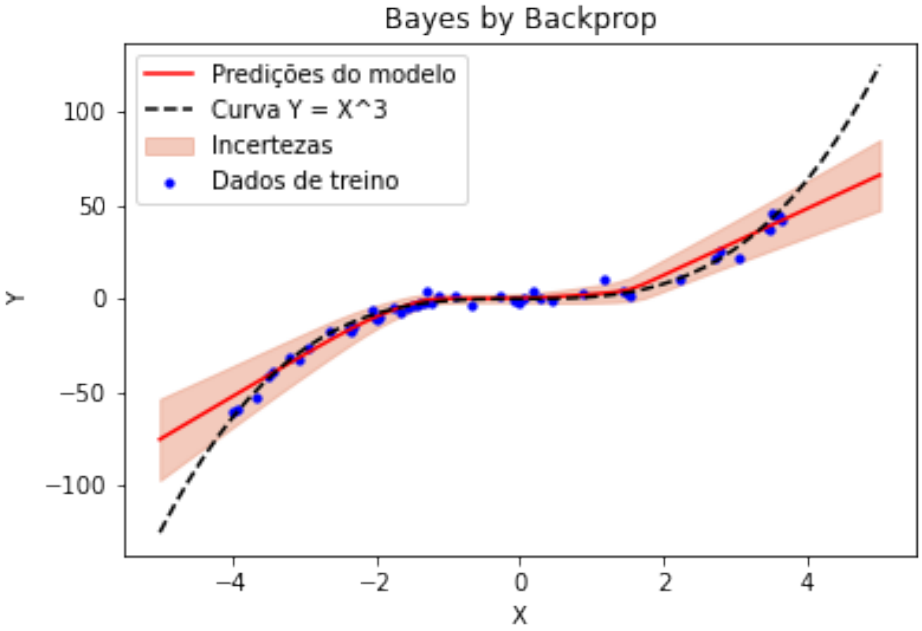
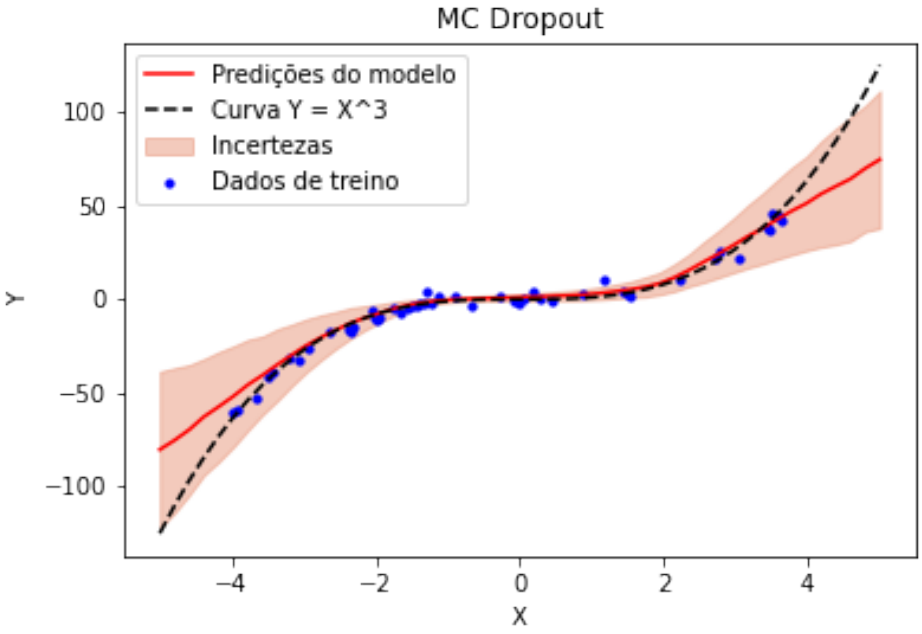


Figura 32: Resultado da rede neural treinada via *MC Dropout*



Tal como é ilustrado nas Figuras 31 e 32, ambos os métodos foram capazes de ajustar modelos de regressão não-linear para os dados e de representar os valores de incerteza de suas predições. Nota-se que em ambos os casos os valores de incerteza são menores nas regiões com maior concentração de dados de treinamento, o que é um fato esperado dado que estas são regiões nas quais os modelos treinados possuem mais informações para realização de predições e consequente são mais seguros em suas predições. Nas regiões com menor concentração de dados, os valores de incerteza são mais elevados e as predições obtidas pelos modelos se distanciam dos valores desejados. Comparando a performance das predições em cada um dos casos, o modelo treinado via *MC Dropout* obteve uma curva de predições mais próxima da curva $y = x^3$, com um valor de coeficiente de determinação R^2 igual a 0.91, enquanto que o modelo treinado via o método *Bayes by Backprop* apresentou um valor de coeficiente de determinação R^2 igual a 0.88. Desse modo, o modelo treinado via *MC Dropout* demonstrou-se ser uma melhor opção para a solução do problema proposto.

3.4.4.5 Análise comparativa

Pelo fato de não exigir alterações na estrutura dos modelos de redes neurais ou nos seus métodos tradicionais de treinamento, a implementação do método *MC Dropout* demonstrou ser muito mais simples e direta comparado à implementação do método *Bayes by Backprop*, o qual exige que cada parâmetro peso da rede neural seja descrito por dois parâmetros variacionais. Além disso, o método *MC Dropout* demonstra ser de mais fácil adaptação para diferentes arquiteturas de redes neurais tais como redes neurais recorrentes e convolucionais (GAL; GHAHRAMANI, 2016b; GAL; ISLAM; GHAHRAMANI, 2017), diferentemente do método *Bayes by Backprop* que é mais difícil de ser ajustado, por exemplo, para redes neurais recorrentes (FORTUNATO; BLUNDELL; VINYALS, 2017). Com relação ao custo computacional dos métodos, o *Bayes by Backprop* é notoriamente mais custoso que o *MC Dropout* pelo fato de duplicar o número de parâmetros que devem ajustados durante o treinamento da rede neural. Por essas razões, o método *MC Dropout* será utilizado para o treinamento dos modelos de redes neurais bayesianas nos estudos de casos de estudo propostos para esse trabalho.

4 ESTUDO DE CASO 1

4.1 Introdução

A poluição do ar em áreas urbanas é considerada um dos principais problemas ambientais da atualidade. Estima-se que 96% da população urbana esteja exposta a níveis de poluição atmosférica acima do recomendado (LEWIS; EDWARDS, 2016) e que cerca de 4.2 milhões de mortes por ano no mundo possuam relação à exposição a esse tipo de poluição (WHO, 2016, 2021). Nesse contexto, a implantação de sistemas de monitoramento da qualidade do ar tem se estabelecido como um componente chave para a remediação da poluição do ar em grandes cidades. Ao realizar a coleta de dados sobre concentrações de poluentes atmosféricos em diferentes regiões das cidades, esses sistemas possibilitam a construção de mapas de concentração de poluentes que fornecem informações às autoridades públicas para o entendimento do comportamento dos poluentes atmosféricos dentro das áreas urbanas e que auxiliam na execução de ações de remediação contra esses poluentes.

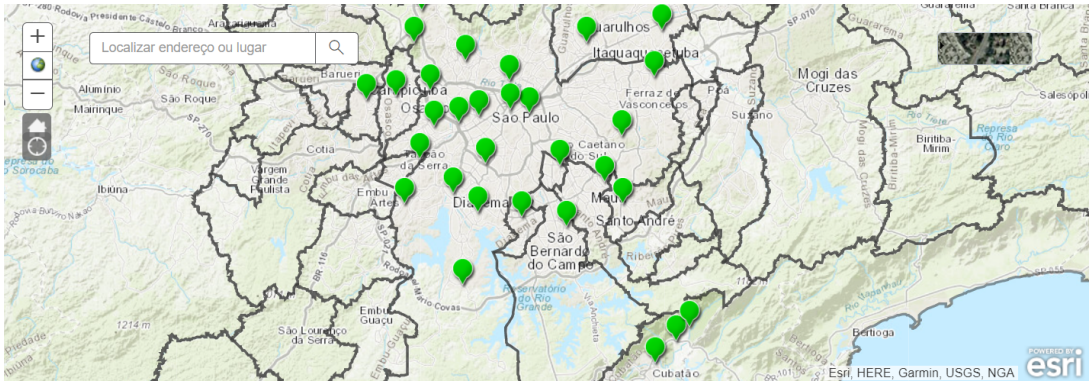
Figura 33: Exemplo de estação de monitoramento da qualidade do ar

Fonte: CETESB (2019)

Tradicionalmente os sistemas de monitoramento da qualidade do ar utilizados em grandes cidades são baseados em estações de monitoramento, tal como a ilustrada na Figura 33. Essas estações realizam a coleta de dados de concentração de poluentes atmosféricos em seus locais de instalação dentro das cidades e são caracterizadas por serem equipadas por robustos equipamentos de medição de concentração de poluentes at-

mosféricos, tais como espectrômetros e cromatógrafos, os quais são capazes de fornecer medições de concentração com um elevado nível de precisão. Essas estações, no entanto, possuem um elevado custo de aquisição e operação, o que limita o número de unidades de monitoramento desse tipo que as cidades podem conter em seus sistemas de monitoramento da qualidade do ar. Dessa forma, tal como exemplificado na Figura 34, as cidades geralmente possuem um número pequeno de estações de monitoramento instaladas de forma esparsa em sua área urbana, o que faz com que seus sistemas de monitoramento forneçam mapas de concentração de poluentes atmosféricos com baixa resolução espacial (HASENFRATZ; SAUKH; THIELE, 2012).

Figura 34: Estações de monitoramento na região metropolitana de São Paulo



Fonte: CETESB (2022)

Mapas de concentração de poluentes atmosféricos de baixa resolução espacial não são considerados ideias para monitoramento da qualidade do ar em grandes cidades, pois eles não fornecem informações precisas sobre o comportamento da poluição atmosférica de toda área urbana monitorada. Normalmente mapas de baixa resolução só fornecem informações precisas sobre o comportamento da poluição atmosférica nos locais onde as estações de monitoramento estão instaladas ou em regiões próximas a essas estações, não sendo capazes de fornecer informações sobre o comportamento da poluição atmosférica em locais distantes das estações de monitoramento. A disponibilização de mapas concentração de poluentes atmosféricos de alta resolução espacial é importante especialmente em grandes cidades, dado que os poluentes atmosféricos normalmente se comportam de forma bem específica dependendo da localização em uma área urbana. Mesmo em uma região pequena de uma cidade, tal como um bairro, variações significativas na concentração de poluentes gasosos podem existir devido a elementos de sua ocupação urbana, como grandes avenidas, cruzamentos movimentados, e *canyons* urbanos (HASENFRATZ; SAUKH; THIELE, 2012; HASENFRATZ et al., 2015). Sendo assim, o ideal é que as grandes cidades utilizem sistemas de monitoramento com um número elevado de estações de mo-

nitoramento instaladas em toda sua área urbana de forma a tornar possível a construção de mapas de concentração de poluentes de alta resolução espacial. No entanto, devido ao elevado custo das estações de monitoramento, sistemas de monitoramento nesses moldes não são viáveis economicamente para a maioria das cidades.

Nesse contexto, na última década, sensores de poluição do ar de baixo custo têm emergido como opções promissoras para o desenvolvimento de sistemas de monitoramento da qualidade do ar economicamente viáveis capazes de gerar mapas de concentração de poluentes com alta resolução espacial (MORAWSKA et al., 2018). Sensores desse tipo, ilustrados na Figura 35, consistem em dispositivos para medição de concentração de poluentes gasosos de bem menor custo e dimensão comparados aos robustos equipamentos das tradicionais estações de monitoramento e que podem utilizados para a construção de pequenas estações de monitoramento da qualidade do ar de baixo custo, tal como a ilustrada na Figura 36.

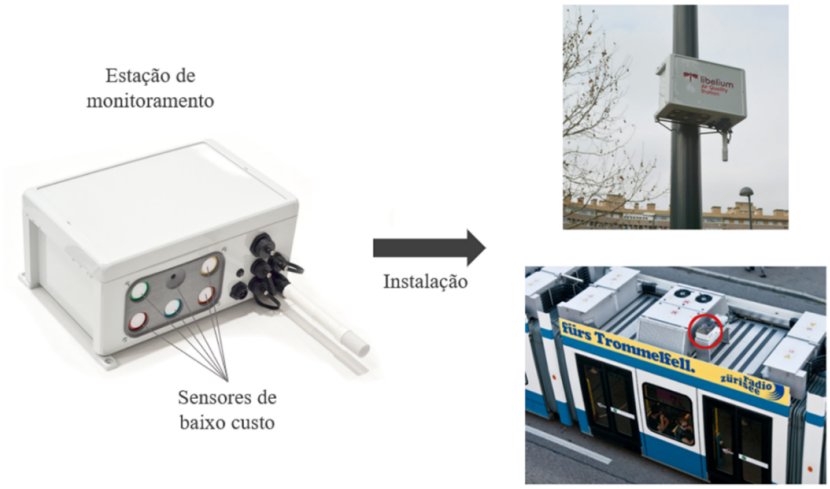
Figura 35: Exemplos de sensores de poluição do ar de baixo custo

Fonte: Libelium (2015)

Essas pequenas estações de monitoramento, além de abrigarem os sensores de poluição, elas são equipadas com sensores para medição de condições climáticas (temperatura, umidade relativa, direção do vento, etc) e com os *hardwares* necessários para a operação da estação e gerenciamento dos dados coletados, tais como placas para integração dos sensores, placas para armazenamento, processamento e controle dos dados, antenas GPS (*Global Positioning System*), entradas de energia, além de portas de comunicação à Internet que permitem que os dados sejam enviados para servidores centrais onde os dados coletados por todas estações de monitoramento da cidade monitorada são integrados (CASTELL et al., 2017; MORAWSKA et al., 2018).

Devido ao seu baixo custo, essas pequenas estações de monitoramento podem ser amplamente instaladas nas cidades aproveitando por exemplo elementos da infraestrutura urbana, tais como postes de iluminação e veículos de transporte público, e utilizadas de forma a auxiliar as estações tradicionais de monitoramento realizando a coleta de dados

Figura 36: Exemplo de estação de sensores de poluição do ar de baixo custo



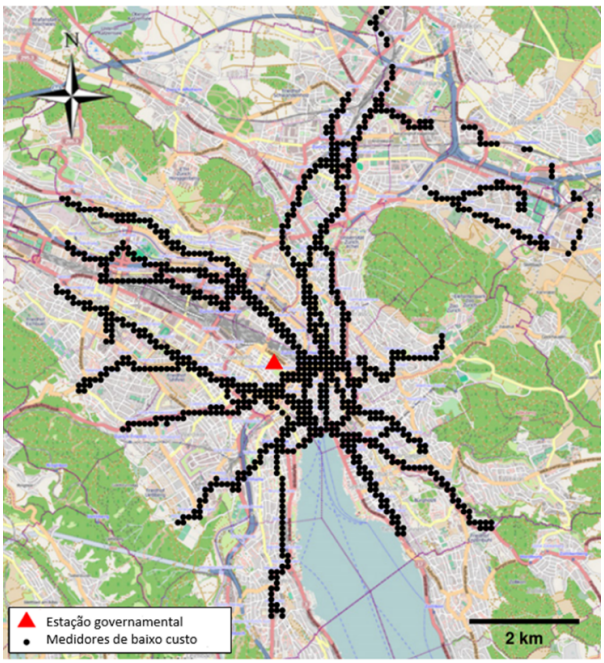
Fonte: Hasenfratz et al. (2015), Libelium (2022a, 2022b)

de concentração em regiões onde não seria possível apenas com as estações tradicionais (MORAWSKA et al., 2018; CONCAS et al., 2021). Desse modo, tal como exemplificado na Figura 37, a ampla instalação dessas estações de baixo custo possibilita o aumento substancial do número de pontos de coleta de dados de concentração de poluentes atmosféricos dos sistemas de monitoramento de qualidade do ar já existentes nas cidades, permitindo assim que eles gerem mapas de concentração de poluentes atmosféricos com alta da resolução espacial de forma economicamente viável.

Os sensores de poluição do ar de baixo custo podem ser categorizados em duas classes: sensores de poluentes particulados e sensores de poluentes gasosos. Os sensores de poluentes particulados de baixo custo normalmente são baseados em princípios de detecção óptica. Nesses sensores o ar é bombeado para uma pequena câmara, onde uma fonte de luz, tal como um LED (*Light-Emitting Diode*) ou um laser de baixa potência, é utilizado para iluminar o ar e assim avaliar a dispersão da luz. Dependendo do número de partículas na mistura de ar, a luz é dispersa com uma intensidade diferente, a qual é medida por um fotodiodo. Ao medir a intensidade de dispersão de luz, o fotodiodo gera um sinal de corrente elétrica, o qual é então associado ao valor de concentração de poluente particulado presente no ar (MAAG; ZHOU; THIELE, 2018).

Já os sensores de poluentes gasosos de baixo custo normalmente consistem em sensores semicondutores óxido metálicos ou sensores eletroquímicos. Um sensor semicondutor óxido metálico é composto basicamente por um componente semicondutor óxido metálico sensorial que ao ser aquecido e entrar em contato com ar é capaz de detectar a presença de um certo poluente gasoso alvo por meio de uma reação química entre esse

Figura 37: Monitoramento da qualidade do ar da cidade de Zurique

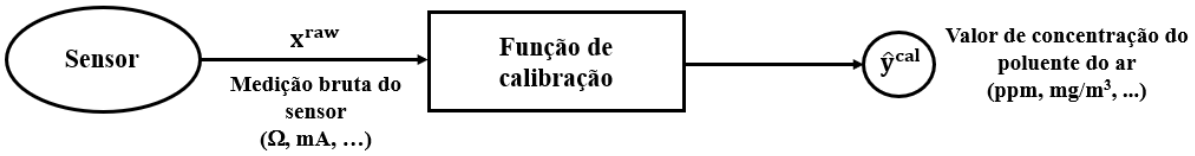


Fonte: Hasenfratz et al. (2015)

poluente e sua superfície. Essa reação gera diferentes alterações na condutividade elétrica do componente sensorial dependendo da concentração do poluente, possibilitando assim que o valor de condutividade elétrica resultante da reação seja medido pelo sensor e associado ao valor da concentração do poluente gasoso alvo presente no ar. Já um sensor eletroquímico é composto por dois eletrodos separados por um meio eletrolítico e detecta a presença de um certo poluente gasoso no ar por meio de uma reação eletroquímica desse poluente em um dos eletrodos. Essa reação eletroquímica gera uma corrente elétrica entre os eletrodos, a qual apresenta diferentes intensidades dependendo da concentração do poluente, permitindo assim que o valor de intensidade da corrente elétrica gerada entre eletrodos seja medido pelo sensor e associado ao valor da concentração do poluente gasoso alvo presente no ar (MAAG; ZHOU; THIELE, 2018; CONCAS et al., 2021).

Uma característica desses tipos de sensores de poluição do ar é que eles fornecem medições brutas em unidades de resistência ou corrente elétrica, o que torna necessário que eles utilizem funções de calibração para conversão dessas medições brutas para valores de concentração do poluente do alvo, tal como esquematizado na Figura 38. Essas funções são normalmente modelos matemáticos parametrizados cujos parâmetros são calibrados de forma a associar corretamente o valor de resistência ou corrente elétrica medido pelo sensor ao valor da concentração do seu poluente alvo presente no ar.

Figura 38: Função de calibração de um sensor de poluente do ar



O principal problema desses sensores de baixo custo, no entanto, é que, comparado aos robustos equipamentos das estações de monitoramento, suas medições não são tão estáveis e precisas. Esses sensores tendem a sofrer com o problema de *drift* relacionado à degradação de seus componentes sensoriais ao longo do tempo que faz com que suas medições percam precisão com decorrer do tempo. Além disso, um grande problema desses sensores é que suas medições são fortemente afetadas por fatores externos, como a umidade e temperatura do local de medição, assim como pela influência de outros poluentes que não sejam seu poluente alvo de detecção, problema que é chamado de sensibilidade cruzada. Sendo assim, para que esses sensores possam ser utilizados na prática, uma solução muito utilizada é a calibração periódica desses dispositivos durante seu período de operação. A calibração desses sensores consiste basicamente no ajuste dos parâmetros de suas funções de calibração de modo a corrigir os erros que afetam suas medições e assim permitir que os valores de concentração fornecidos pelos sensores se aproximem ao máximo dos valores de concentração esperados. A calibração periódica dos sensores de baixo custo permite por exemplo que os erros gerados pelos problemas de *drift* e sensibilidade cruzada sejam periodicamente corrigidos, possibilitando assim que os sensores mantenham um nível de precisão aceitável durante todo seu o período de operação (CONCAS et al., 2021; MAAG; ZHOU; THIELE, 2018).

Tradicionalmente as calibrações de sensores de poluição do ar de baixo custo são realizadas em laboratórios. No entanto, uma desvantagem dessa abordagem de calibração é que em laboratório os sensores dificilmente são calibrados nas mesmas condições externas que serão encontradas em seu local operação, podendo fazer que eles performem mal em campo mesmo após a calibração. Além disso, com a ampla instalação desses sensores nas cidades, essa abordagem de calibração tem se tornado cada vez mais impraticável, uma vez que ao utilizá-la muitos sensores precisariam ser movidos para laboratórios, submetidos a processos demorados de calibração e posteriormente reinstalados em seus locais de operação (VITO et al., 2018; CONCAS et al., 2021).

Nesse contexto, métodos de calibração baseados em modelos de aprendizado de máquina têm emergido como excelentes soluções para a calibração de sensores de poluição

do ar de baixo custo, pois permitem que os sensores sejam calibrados em campo, ou seja, os dispositivos são calibrados no seu local de operação, sem a necessidade de serem deslocados para laboratórios, e consequentemente nas mesmas condições externas nas quais serão submetidos durante sua operação, tornando o processo de calibração desses sensores mais prática e eficiente (CONCAS et al., 2021).

Sendo assim, esse estudo de caso tem como proposta apresentar a aplicação de um novo método baseado em rede neural bayesiana para a calibração de um sensor de poluição do ar de baixo custo e provar sua aplicabilidade. Diversos estudos sobre aplicação de métodos baseados em redes neurais tradicionais para calibração de sensores de poluição do ar de baixo custo são encontrados na literatura, porém não há estudos sobre aplicação de métodos baseados em redes neurais bayesianas para solução do mesmo tipo de problema. Dessa forma, esse estudo de caso tem como objetivos específicos apresentar a aplicação de um método baseado em rede neural bayesiana para calibração de sensores de poluição do ar de baixo custo utilizando um conjunto de dados *benchmark*, avaliar desempenho do método e analisar suas vantagens e desvantagens.

4.2 Análise bibliográfica

A calibração de sensores de poluição do ar de baixo custo é tradicionalmente realizada em laboratórios. Nessa abordagem o sensor a ser calibrado é colocado em uma câmara à vácuo onde ele é exposto a diferentes misturas de gases poluentes com concentrações conhecidas. Por meio desse procedimento, os parâmetros da função de calibração do sensor são então ajustados com o objetivo de minimizar a diferença entre os valores de concentração do poluente alvo retornados pelo sensor e os valores reais de concentração desse poluente para cada uma das misturas (VITO et al., 2018; CONCAS et al., 2021).

Um problema dessa abordagem de calibração é que os sensores em laboratório não são calibrados nas mesmas condições externas que eles encontrão em seu local operação (VITO et al., 2018). Pelo fato das medições de sensores de poluição do ar de baixo custo serem fortemente afetadas por fatores externos (tais como a temperatura, a umidade e a influência de outros poluentes), para que eles sejam calibrados de forma adequada é necessário que eles sejam expostos a condições externas semelhantes a que eles encontrarão em seu local de operação, no entanto isso é bem difícil de ser realizado em laboratório. Dificilmente as misturas de gases às quais os sensores são expostos em laboratório serão idênticas em termos de composição às misturas de gases que eles encontrarão em campo, e as condições climáticas simuladas em laboratório normalmente também não cobrem todas variações de condições possíveis de serem encontradas pelos sensores durante sua operação. Dessa forma, uma crítica feita a essa abordagem de calibração é que os sensores

mesmo após calibrados em laboratório não performam bem em campo (CASTELL et al., 2017).

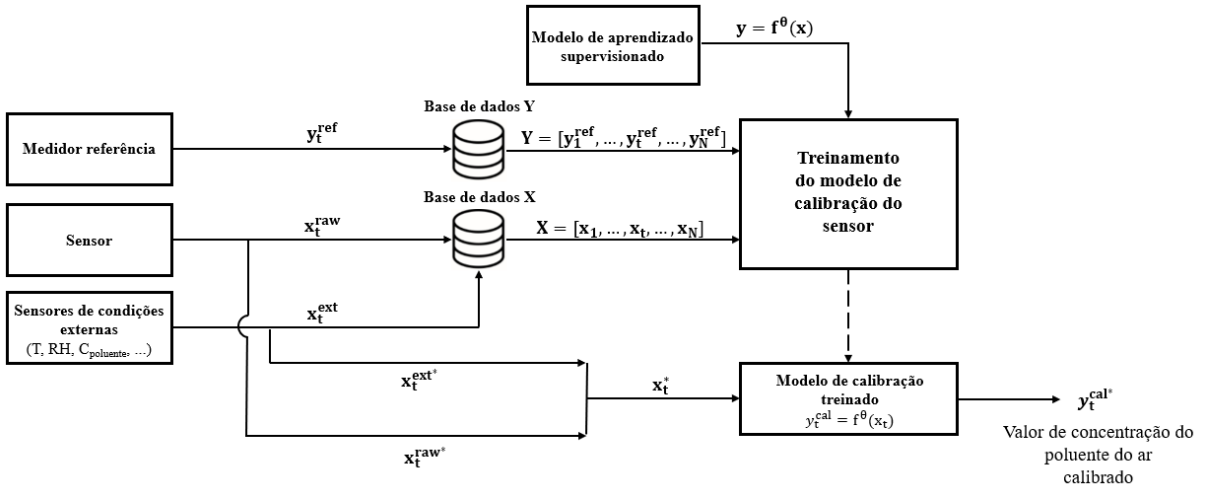
Além disso, um outro problema dessa abordagem é que o processo de calibração de um único sensor em laboratório pode consumir muito tempo (VITO et al., 2018). Nessa abordagem, o processo de calibração normalmente envolve a exposição do sensor a uma grande variedade de misturas de gases de modo que ele possa explorar a maior quantidade possível de combinações de diferentes concentrações de gases, temperaturas e umidades. Isso torna o processo de calibração bem demorado, pois a câmara à vácuo precisa ser ajustada para cada mistura de gases a qual o sensor é exposto. Devido a esse problema, com a ampla instalação desses sensores nas cidades, a abordagem de calibração em laboratório tem se tornado cada vez mais impraticável, uma vez que ao utilizá-la um enorme número de sensores precisariam ser movidos para laboratórios, submetidos a longos processos de calibração e posteriormente reinstalados em seus locais de operação (CONCAS et al., 2021).

Nesse contexto, métodos de calibração baseados em modelos de aprendizado de máquina têm emergido como excelentes soluções para a calibração de sensores de poluição do ar de baixo custo pelo fato de permitirem que os sensores sejam calibrados em campo (CONCAS et al., 2021). O processo de calibração do sensor nessa abordagem consiste no treinamento de um modelo de aprendizado de máquina multivariável capaz de analisar um dado bruto x^{raw} coletado pelo sensor e dados de condições externas x^{ext} que afetam as medições do sensor (como a concentração de outros poluentes, a temperatura ambiente e a umidade relativa do ar) coletados por outros sensores da estação de monitoramento, e retornar como resposta um valor de concentração do poluente alvo do sensor y^{cal} que se aproxime ao máximo do valor de concentração do mesmo poluente y^{ref} medido por algum medidor referência instalado próximo ao sensor, tal como uma tradicional estação de monitoramento ou outro sensor previamente calibrado, os quais normalmente possuem um melhor precisão comparado ao sensor sendo calibrado.

Tal como pode ser observado na Figura 39, o processo de calibração do sensor utilizando métodos de aprendizado de máquina pode ser visto como a solução de um problema de aprendizado supervisionado em que um modelo de calibração $y_t^{cal} = f^{\theta}(x_t)$ é treinado utilizando um conjunto de dados rotulados $D = (x_t, y_t^{ref})_{t=1}^{N}$, onde x_t denota um dado de entrada do modelo que consiste na concatenação do dado bruto x_t^{raw} e dados de condições externas x_t^{ext} coletados no instante t ($x_t = [x_t^{raw}, x_t^{ext}]$), e y_t^{ref} denota seu respectivo dado de saída alvo composto pelo dado de concentração do poluente alvo coletado pelo medidor referência no mesmo instante t. Os dados de entrada $X = [x_1, \ldots, x_N]$ são normalmente obtidos a partir de dados que armazenam os dados coletados pela estação

de monitoramento na qual o sensor está instalado, enquanto que os dados de saída alvo $Y = [y_1^{ref}, \ldots, y_N^{ref}]$ são obtidos a partir de base de dados que armazenam os dados coletados pelo medidor referência do sensor.

Figura 39: Abordagem dos métodos de calibração de sensores baseados em modelos de aprendizado de máquina



O treinamento do modelo de calibração $y_t^{cal} = f^{\theta}(x_t)$ de um sensor consiste geralmente na solução de um problema de otimização em que os parâmetros θ do modelo são ajustados visando minimizar a diferença dos valores de concentração do poluente alvo retornados pelo modelo ($[y_1^{cal}, ..., y_N^{cal}]$) e os valores de concentração do mesmo poluente fornecidos pelo medidor referência do sensor ($[y_1^{ref}, ..., y_N^{ref}]$), tal como descrito na Equação 4.1. Após o treinamento, a ideia é que o modelo de calibração $y_t^{cal} = f^{\theta}(x_t)$ seja capaz de receber um novo dado bruto x_t^{raw*} junto de seus respectivos dados de condições externas x_t^{ext*}, e retorne o seu correspondente valor de concentração calibrado y^{cal*}.

$$\min_{\theta} \frac{1}{N} \sum_{t=1}^{N} ||f^{\theta}(x_t) - y_t^{ref}|| \tag{4.1}$$

Pelo fato dos conjuntos de dados $X = [x_1, \ldots, x_N]$ e $Y = [y_1^{ref}, \ldots, y_N^{ref}]$ utilizados no treinamento o modelo poderem ser obtidos a partir de banco de dados, os métodos de calibração baseados em modelos de aprendizado de máquina permitem que os sensores sejam calibrados em campo por meio apenas da análise desses dados, utilizando os recursos de processamento de dados das próprias estações monitoramento ou mesmo servidores na nuvem. Ou seja, os sensores são calibrados sem precisar ser removidos dos seus locais de operação e sob as mesmas condições externas às quais eles serão submetidos durante sua operação, o que torna o processo de calibração desses sensores mais prática e eficiente (CONCAS et al., 2021).

Ao longo dos anos diferentes métodos baseados em modelos aprendizado de máquina para calibração de sensores de poluição do ar foram propostos na literatura. Entre esses diversos métodos, destacam-se métodos tais como os baseados em modelos de regressão linear (HASENFRATZ; SAUKH; THIELE, 2012; SAUKH; HASENFRATZ; THIELE, 2015; CORDERO; BORGE; NARROS, 2018; VITO et al., 2018; ZIMMERMAN et al., 2018), *support vector machines* (SVM) (CORDERO; BORGE; NARROS, 2018; VITO et al., 2018), e *random forest* (CORDERO; BORGE; NARROS, 2018; ZIMMERMAN et al., 2018). Em especial, métodos de calibração baseados em modelos redes neurais têm sido muito utilizados ao longo dos anos, dado a capacidade dos modelos de redes neurais processarem conjuntos de dados multivariáveis que incluem dados de condições externas e lidarem bem com o problema de sensibilidade cruzada, e devido ao fato desses modelo possuírem uma excelente capacidade de modelarem comportamentos não lineares entre dados de entrada do modelo de calibração e seus dados de saída alvo (VITO et al., 2018). Métodos baseados , por exemplo, em redes neurais *feed-forward* (VITO et al., 2008, 2009; SPINELLE et al., 2015; ESPOSITO et al., 2016; SPINELLE et al., 2017; CORDERO; BORGE; NARROS, 2018; VITO et al., 2018) e redes neurais recorrentes (ESPOSITO et al., 2016) são encontrados na literatura. Entretanto, uma característica comum dos métodos de calibração baseados em redes neurais é que eles utilizam esses modelos em uma abordagem determinística e consequentemente possuem as desvantagens de sofrerem com o problema de sobreajuste e de não serem capazes de fornecer valores de incerteza do dado de concentração final fornecido pelo modelo de calibração.

Valores de incerteza são de grande valia no contexto da validação do uso de sensores de poluição de baixo custo, pois permitem avaliar a confiabilidade dos dados fornecidos pelo modelo de calibração. Esses valores são tão relevantes tal que um dos critérios estabelecidos pelo conselho da União Europeia para autorizar a utilização desses novos dispositivos como medidores oficiais da qualidade do ar é que os dados calibrados fornecidos pelos sensores não ultrapassem um certo valor permitido de incerteza (UNION, 2008). Entre os trabalhos sobre métodos de calibração baseados em redes neurais apenas alguns deles abordam a obtenção de estimativas de valores incerteza do dado de concentração final fornecido pelo modelo de calibração (SPINELLE et al., 2015; ESPOSITO et al., 2016; SPINELLE et al., 2017) e em suas abordagens essas estimativas não são fornecidas diretamente pelo modelo, exigindo a utilização um mecanismo extras para o cálculo dos valores de incerteza.

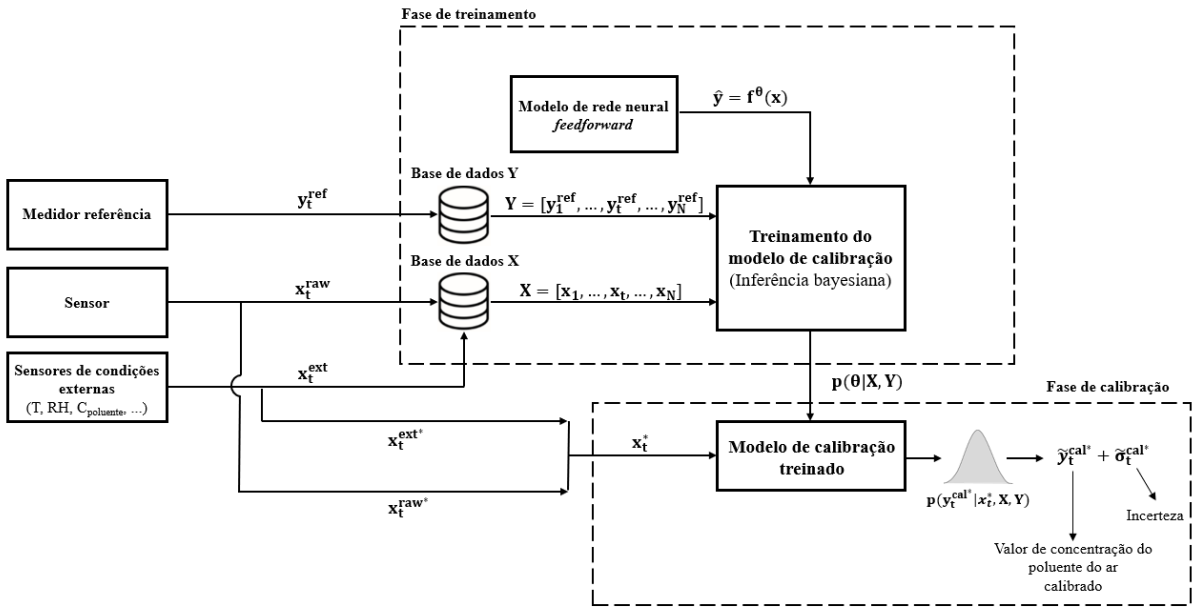
Nesse contexto, o desenvolvimento de métodos de calibração baseados em redes neurais bayesianas se estabelece como uma alternativa potencial para solução dos problemas dos métodos de calibração baseados em redes neurais determinísticas, dado ao

fato das redes neurais bayesianas serem mais robustas contra o problema de sobreajuste e serem capazes de fornecer estimativas dos valores de incerteza de suas saídas de forma direta. Ao analisar a literatura, nota-se que existem diversos trabalhos sobre métodos de calibração baseados em redes neurais para calibração de sensores de poluição do ar, porém nenhum deles aborda o uso de redes neurais bayesianas. Sendo assim, esse trabalho demonstra ser pioneiro nessa questão.

4.3 Metodologia

Para esse estudo de caso propõe-se a aplicação de um novo método baseado em rede neural bayesiana para a calibração de um sensor de poluição do ar de baixo custo. Conforme ilustrado na Figura 40, esse método baseia-se no treinamento de um modelo de rede neural *feedforward* bayesiana $y_t^{cal} = f^{\theta}(x_t)$ por meio de um processo de inferência de bayesiana de modo a capacitá-lo a receber como entrada um dado x_t, composto por um dado bruto coletado pelo sensor (x_t^{raw}) e dados de condições externas (x_t^{ext}) coletados no instante t, e retornar como respostas o valor de concentração do poluente alvo do sensor de entrada que se aproxime ao máximo do valor de concentração medido pelo medidor referência do sensor, assim como uma estimativa do valor de incerteza desse valor de concentração retornado pelo modelo.

Figura 40: Esquema do método de calibração de sensores de poluição do ar baseado em rede neural bayesiana



Pelo fato de o modelo de rede neural treinado ser bayesiano, sua resposta consiste em uma distribuição probabilística preditiva aproximada $p(y_t^{cal}|x_t, X, Y)$ dos valores mais prováveis para o dado concentração do poluente alvo, cuja média $\tilde{y}_t^{cal}$ é utilizada como o valor de concentração do poluente alvo do sensor retornado pelo modelo de calibração e variância $\tilde{\sigma}_t^{cal}$ é utilizada como do seu respectivo valor de incerteza. Para o treinamento do modelo, dados de entrada $X = [x_1, \ldots, x_N]$ e dados de saída alvo $Y = [y_1^{ref}, \ldots, y_N^{ref}]$ previamente coletados e armazenados em bancos de dados são utilizados e, com relação ao processo de inferência bayesiana necessário para o treinamento da rede neural bayesiana, o método de aproximação de inferência bayesiana *MC Dropout* é utilizado. Esse método foi escolhido pelo fato de ser simples de implementar, não exigir modificação na estrutura da rede neural *feedforward* e garantir um boa capacidade de generalização ao modelo (GAL; GHAHRAMANI, 2016a, 2016b). Uma vez treinado, a ideia é que o modelo de rede neural bayesiana seja então aplicado para receber novos dados brutos x_t^{raw*} coletados pelo sensor junto de seus correspondentes dados de condições externas x_t^{ext*}, e retorne seus respectivos dados de concentração calibrados $\tilde{y}_t^{cal*}$ e estimativas de suas incertezas $\tilde{\sigma}_t^{cal*}$.

4.4 Experimento

Com objetivo de avaliar desempenho do método baseado em rede neural bayesiana para a calibração de um sensor de poluição do ar de baixo custo proposto nesse estudo, um experimento da aplicação desse método é realizado utilizando um conjunto de dados *benchmark* para avaliação de métodos de calibração de sensores de poluição do ar de baixo custo.

4.4.1 Conjunto de dados

Para esse experimento é utilizado o conjunto de dados do ENEA (*National Agency for New Technologies, Energy and Sustainable Economic Development*) (VITO et al., 2008), o qual consiste em um conjunto com 9258 instâncias de médias horárias de medições realizadas por cinco sensores químicos óxido metálicos instalados em uma estação de monitoramento multissensorial de qualidade do ar e por equipamentos de uma estação de monitoramento da qualidade do ar referência. A coleta dos dados foi realizada em uma área de tráfego de carros significativamente poluída em uma cidade italiana no período entre março de 2004 e fevereiro de 2005. A vantagem da utilização desse conjunto de dados se deve ao fato dele já ser utilizado como *benchmark* para avaliação de diferentes métodos de calibração baseados em modelos de aprendizado de máquina (VITO et al., 2018), possibilitando assim a comparação da performance do modelo a ser desenvolvido nesse caso de estudo a outros modelos já testados para esse conjunto de dados. As variáveis

disponíveis no conjunto de dados são listadas na Tabela 1.

Tabela 1: Variáveis do conjunto de dados do ENEA

Variável	Nome	Descrição	Unidade
$x^{(1)}$	Date	Data	DD/MM/YYYY
$x^{(2)}$	Time	Horário	HH.MM.SS
$x^{(3)}$	CO	Concentração média horária de CO – medição referência	mg/m^3
$x^{(4)}$	PT08.S1	Medição bruta média horária do sensor de CO	Ω
$x^{(5)}$	NMHC	Concentração média horária de hidrocarbonetos não metânicos NMHC - medição referência	mg/m^3
$x^{(6)}$	PT08.S2	Medição bruta média horária do sensor de hidrocarbonetos não metânicos NMHC	Ω
$x^{(7)}$	C6H6	Concentração média horária de benzeno - medição referência	mg/m^3
$x^{(8)}$	NO_x	Concentração média horária de NO_x – medição referência	ppb
$x^{(9)}$	PT08.S3	Medição bruta média horária do sensor de NO_x	Ω
$x^{(10)}$	NO_2	Concentração média horária de NO_2 – medição referência	mg/m^3
$x^{(11)}$	PT08.S4	Medição bruta média horária do sensor de NO_2	Ω
$x^{(12)}$	PT08.S5	Medição bruta média horária do sensor de O_3	Ω
$x^{(13)}$	T	Temperatura	°C
$x^{(14)}$	RH	Umidade relativa	%
$x^{(15)}$	AH	Umidade absoluta	-

Para permitir uma comparação justa entre o método proposto nesse estudo e os métodos analisados em Vito et al. (2018), uma configuração de divisão do conjunto de dados semelhante à proposta por Vito et al. (2018) foi utilizada: as primeiras 504 instâncias de dados foram aplicadas para treinamento do modelo e as instâncias restantes foram aplicadas para teste do modelo treinado.

4.4.2 Pré-processamento dos dados

Com objetivo de preparar os dados para treinamento do modelo de calibração proposto, alguns procedimentos de limpeza de dados foram realizados. Uma vez que todas variáveis do conjunto de dados (com exceção das variáveis “Date” e “Time”) contém dados faltantes, ações para correção desses dados foram tomadas. Pelo fato de mais de 90% dos dados do atributo “NMHC” são dados faltantes, optou-se pelo descarte dos dados dessa variável. Já com relação aos dados faltantes das demais variáveis, optou-se por preenchê-los utilizando o método *k-nearest neighbors* (TROYANSKAYA et al., 2001). Por fim, com intuito de evitar problemas de discrepâncias dimensionais, um processo de normalização dos dados foi realizado.

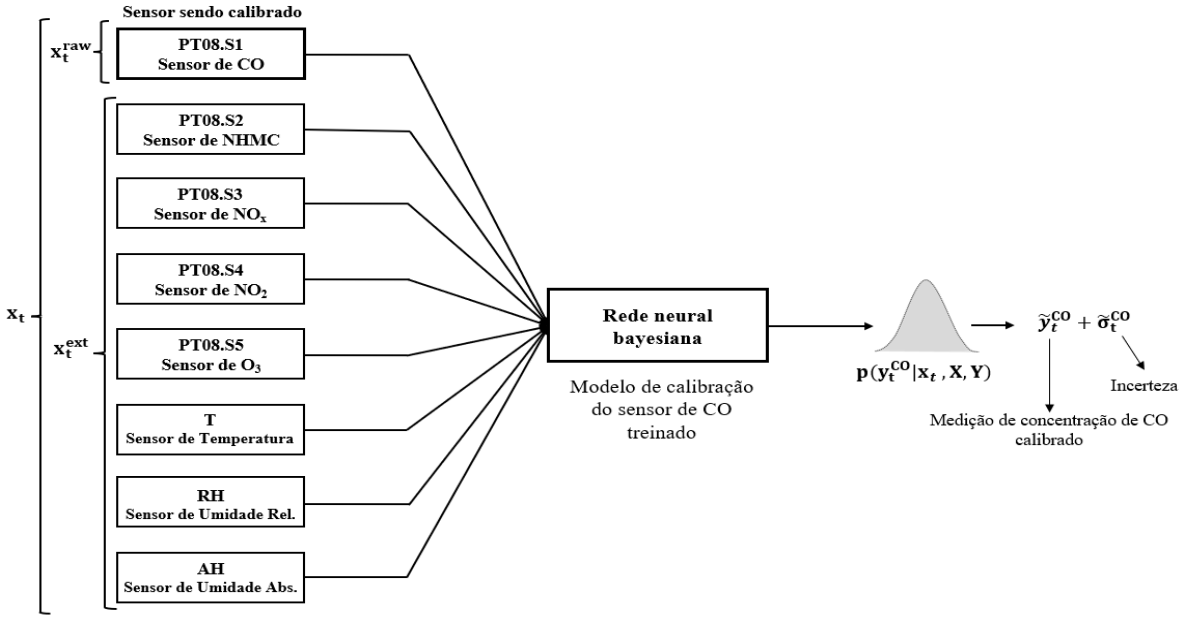
4.4.3 Modelo de calibração

Para avaliar o desempenho do método proposto, especificamente para este experimento, optou-se pelo caso da calibração do sensor PT08.S1 de monóxido de carbono

(CO) dado que a calibração desse sensor é abordada em outros estudos sobre métodos de calibração (VITO et al., 2018), possibilitando assim a comparação de desempenho entre diferentes métodos.

Para o desenvolvimento desse experimento, então, um modelo de rede neural *feedforward* bayesiana foi treinado e testado para calibração do sensor PT08.S1 de CO. Tal como é ilustrado na Figura 41, esse modelo foi estruturado de forma a receber como entrada x_t os dados das variáveis "PT08.S1", "PT08.S2", "PT08.S3", "PT08.S4", "PT08.S5", "T", "RH" e "AH", e ao processar esses dados de entrada, retornar como resposta uma distribuição probabilística preditiva aproximada $p(y_t^{\mathrm{CO}}|x_t, X, Y)$ dos valores mais prováveis para o dado concentração de CO de acordo com os dados de entrada. A média $\tilde{y}_t^{\mathrm{CO}}$ da distribuição $p(y_t^{\mathrm{CO}}|x_t, X, Y)$ é utilizada como o valor de concentração de CO calibrado e a variância $\tilde{\sigma}_t^{\mathrm{CO}}$ da distribuição é utilizada como respectivo valor de incerteza desse valor de concentração de CO calibrado. Os dados da variável de entrada "PT08.S1" são considerados como os dados brutos x_t^{raw} obtidos pelo sensor de CO a serem calibrados, enquanto que os dados das demais variáveis de entrada são considerados como informações das condições externas x_t^{ext} que podem afetar a medição da concentração de CO.

Figura 41: Modelo de rede neural bayesiana para calibração do sensor de CO



Para o treinamento do modelo, dados da variável "CO", os quais são medições referência para medições do sensor PT08.S1 de CO, foram utilizados como dados de saída alvo $y_t^{\mathrm{CO_{ref}}}$. Já com relação ao método para aproximação de inferência bayesiana, tal como explicado na seção 4.3, o método *MC-Dropout* foi utilizado. O modelo foi implementado

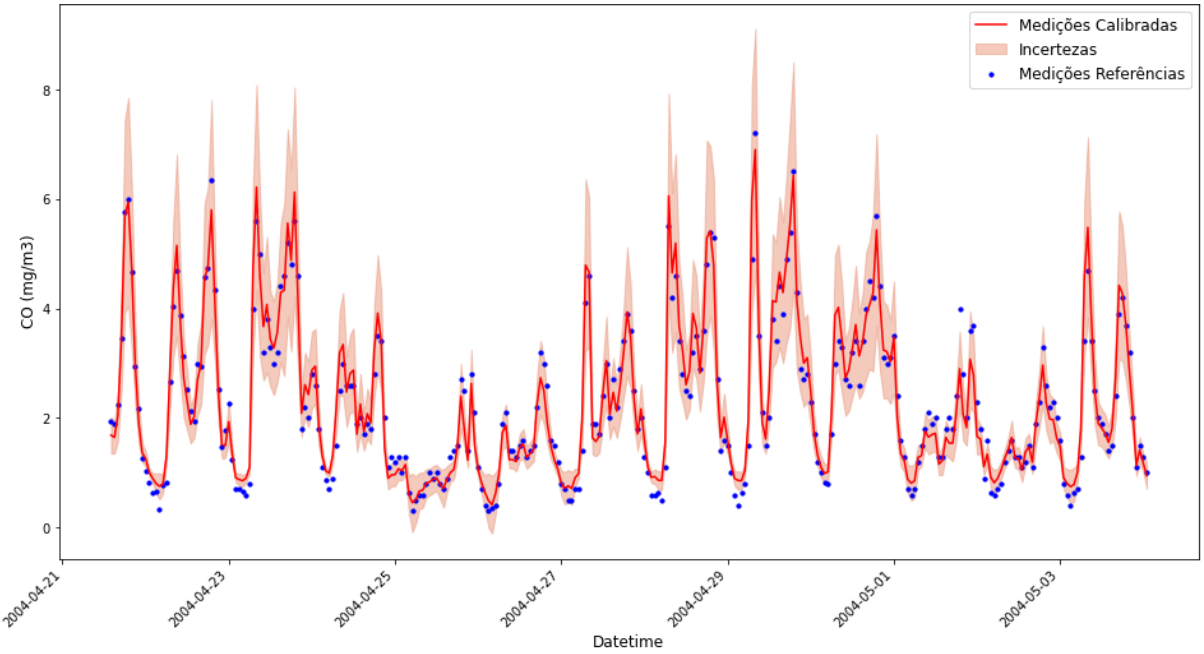
utilizando a linguagem *Python* e a biblioteca *Pytorch* (PASZKE et al., 2019) para construção de modelos de redes neurais. Já seus hiperparâmetros são determinados utilizando o framework *Optuna* (AKIBA et al., 2019).

O modelo final para calibração do sensor de CO foi fixado como uma rede neural bayesiana com uma única camada oculta contendo 24 unidades de processamento que utilizam a função de ativação ReLU e uma camada de saída contendo apenas uma unidade de processamento. Esse modelo foi treinado utilizando o método *MC-Dropout* para aproximação de inferência bayesiana com taxa de *dropout* igual a 0.20, taxa de aprendizado igual a 0.014, 400 épocas (iterações) de treinamento e otimizador *RMSprop* (TIELEMAN; HINTON, 2012). Os códigos de programa utilizados nesse estudo de caso podem ser encontrados no repositório do Github ⟨https://github.com/RyujiGT96/masters-project/tree/master/1-sensor-calibration⟩.

4.5 Resultados e discussão

4.5.1 Resultados de teste

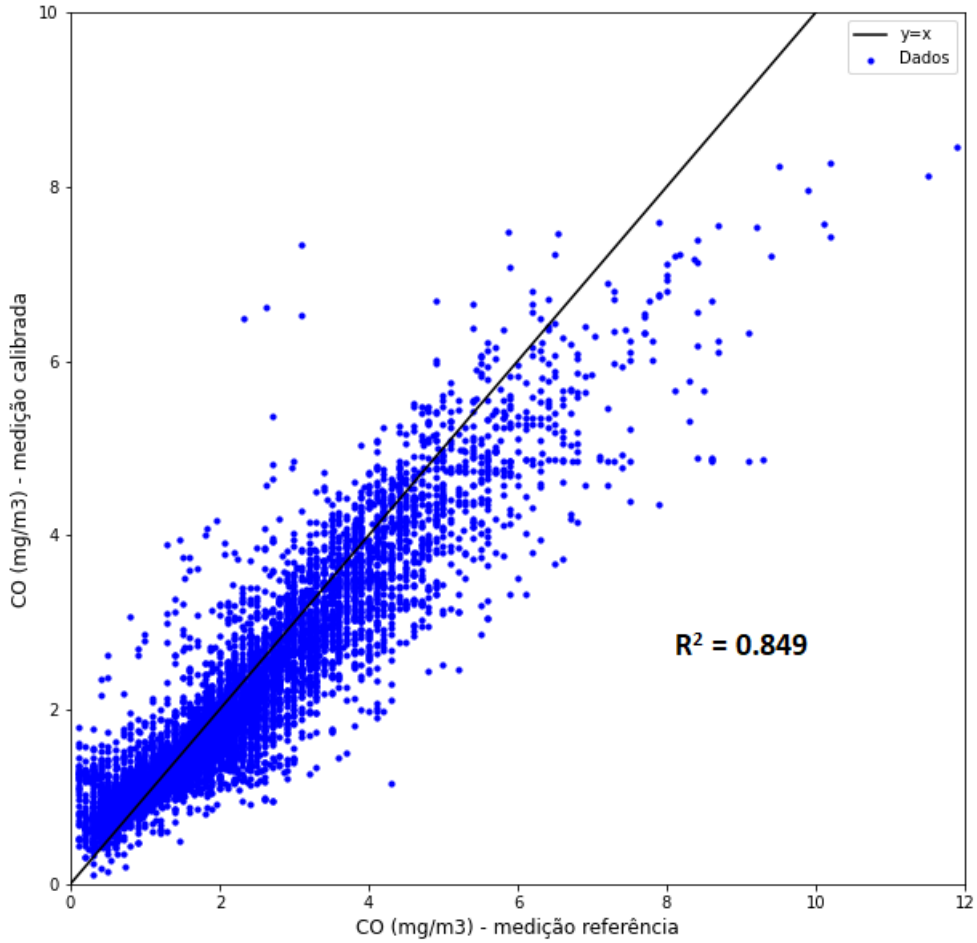
Figura 42: Resultados da calibração do sensor PT08.S1 de CO utilizando o modelo treinado



O resultado da aplicação do modelo treinado para a calibração das medições do sensor PT08.S1 de CO disponibilizadas pelo conjunto de dados de teste é ilustrado na Figura 42. De forma geral, analisando visualmente a Figura 42, nota-se que o modelo apresentou um bom nível de precisão, sendo capaz de gerar medições calibradas da concentração de CO bem próximas de suas respectivas medições de referência. Além disso,

pode ser visto na Figura 42 que as medições calibradas são fornecidas junto com suas respectivas estimativas de incerteza, tornando assim possível avaliar a confiabilidade dessas medições calibradas de acordo com seu nível de incerteza.

Figura 43: Distribuição das medições calibradas pelo modelo em relação às medições referências



Analisando a Figura 43, a qual ilustra a distribuição das medições calibradas pelo modelo em relação às medições referências, nota-se que a distribuição das medições se concentra em torno da reta $y = x$ com um coeficiente de determinação R^2 igual a 0.849, o que comprova numericamente que houve uma boa correlação entre os valores das medições calibradas e os valores de medições referências.

Analisando especificamente as vantagens do uso de uma rede neural em uma abordagem bayesiana para calibração de sensores, notou-se que modelo treinado não sofreu com o problema de sobreajuste (*overfitting*), demonstrando uma boa capacidade de generalização quando testado frente ao conjunto de dados de teste. Esse fato pode ser explicado pela característica das redes neurais bayesianas usufruírem de mecanismos naturais de regularização que evitam que elas sofram sobreajuste. Tal como destacado no trabalho de Concas et al. (2021), a tendência das redes neurais sofrerem sobreajuste é um

dos fatores negativos que desestimulam o uso de redes neurais para calibração de sensores de poluição do ar. Sendo assim, o método de calibração proposto demonstra ser uma possível solução para essa questão.

Além disso, o uso de uma rede neural bayesiana também mostrou ser vantajoso com a relação ao acesso a valores de incerteza das medições calibradas. Em comparação a outros métodos de calibração baseados em redes neurais, tais como os propostos por Spinelle et al. (2015), Esposito et al. (2016) e Spinelle et al. (2017), os quais propõe o acesso das incertezas de suas calibrações por meio uso de mecanismos extras para o cálculo de valores de incerteza de predições de modelos, o método de calibração apresentado nesse estudo demonstrou ser vantajoso ao possibilitar que as estimativas de incerteza das medições calibradas sejam fornecidas diretamente pelo modelo de calibração, facilitando assim a análise desses valores e a avaliação do nível de confiança das medições calibradas. Essa vantagem é explicada pelo fato dos modelos de calibração baseados em redes neurais bayesiana fornecerem como respostas distribuições probabilísticas preditivas das quais pode se obter diretamente estimativas dos valores de incerteza das medições calibradas.

Com relação às limitações do método de calibração proposto, notou-se que o custo computacional para uso do método pode ser um fator limitante para sua aplicação. O treinamento de modelos de calibração baseados em redes neurais é conhecido por ser mais custoso computacionalmente comparado ao treinamento de modelos baseados em outros tipos de técnicas de aprendizado de máquina (CONCAS et al., 2021), e especificamente falando de redes neurais bayesianas, o treinamento pode ser ainda mais custoso, pelo fato de necessitar utilizar métodos de inferência aproximada. Além disso, indo além ao custo de treinamento, a operação dos modelos de redes neurais bayesianas treinados para calibrar as medições dos sensores também é custosa computacionalmente, pois esses modelos geram como resposta distribuições probabilísticas preditivas, as quais, para serem calculadas, exigem um processamento de dados de entrada mais robusto comparado a cálculo de valores pontuais de predição. Sendo assim, pode-se afirmar que o método não é ideal para o caso em que a própria estação de sensores de poluição, utilizando seus limitados recursos computacionais, necessita realizar o treinamento dos modelos de calibração de seus sensores e aplicá-los para calibração de suas medições, tal como é o caso de algumas estações móveis de sensores de poluição do ar. O método demonstra ser mais adequado para o caso de estações sensores fixos que possuem recursos computacionais mais robustos disponíveis ou, por exemplo, para estações móveis de sensores com conexão à internet e suporte para computação em nuvem para processamento de suas medições.

4.5.2 Análise comparativa

Para o desenvolvimento de uma análise comparativa do desempenho do modelo de calibração baseado em rede neural bayesiana utilizado nesse experimento em relação ao desempenho de modelos de calibração baseados também baseados em aprendizado de máquina, foram utilizados dados do desempenho dos modelos de aprendizado de máquina testados no trabalho de Vito et al. (2008), no qual foi realizado um estudo comparativo do desempenho de diferentes modelos de aprendizado de máquina na calibração de sensores utilizando o mesmo conjunto de dados utilizado nesse experimento. No trabalho Vito et al. (2008) foram comparados o desempenho de quatro tipos de modelos de aprendizado de máquina tradicionais: rede neural *feedforward* (não bayesiana), modelo de *Suport Vector Machine* SVM, modelo de *Gaussian Process Regression* (GPR) e modelo de *Multiple Linear Regressor* (MLR). Para realizar um compração justa, foram selecionados dados do desempenho desses modelos quando treinados com 504 amostras de dados e utilizando *tapped delay length* igual a 1 hora, as quais são as mesmas condições utilizadas para treinamento do modelo de calibração testado nesse experimento.

Para comparar o desempenho dos modelos, optou-se por utilizar como métrica o valor de erro absoluto médio (*Mean Absolute Error* – MAE) entre os valores das medições calibradas pelo modelo e os valores de medições referências, uma vez que essa é a métrica disponível com relação ao desempenho dos modelos de aprendizado de máquina testados por Vito et al. (2008) para calibração do mesmo sensor PT08.S1 de CO . Sendo assim, para esse experimento o cálculo do MAE foi formulado como:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |\tilde{y}_i^{\mathrm{CO}} - y_i^{\mathrm{CO_{ref}}}| \tag{4.2}$$

Sendo assim, conforme destacado na Tabela 2, com base no resultado da aplicação do modelo de rede neural bayesiana otimizado para a calibração das medições do sensor PT08.S1 de CO disponibilizados pelo conjunto de dados de teste, o modelo um valor MAE igual a 0.381. Comparando com valores de MAE obtidos pelos modelos testados por Vito et al. (2018) para a calibração do mesmo sensor de CO, o modelo rede neural bayesiana apresentou um valor de MAE inferior em comparação aos valores de MAE obtidos pelos demais modelos, comprovando, portanto, que o método de calibração baseado no uso de modelo de rede neural bayesiana proposto neste estudo pode apresentar um desempenho tão bom quanto ao apresentado por métodos de calibração baseados em modelos de aprendizado de máquina mais tradicionais.

Tabela 2: Comparação de desempenho dos modelos

Método	Mean Absolute Error (MAE)
Multiple Linear Regressors	1.09
Support Vector Machine	0.64
Gaussian Process Regression	0.55
Neural Network	0.61
Bayesian Neural Network	**0.38**

4.6 Conclusões

Visando analisar a aplicabilidade do método baseado em rede neural bayesiana proposto nesse estudo de caso para calibração de sensores de poluição do ar de baixo-custo, foi realizado um experimento de aplicação desse método utilizando um conjunto de dados *benchmark* para avaliação de métodos de calibração baseados em modelos de aprendizado de máquina. O experimento de aplicação mostrou que um modelo de calibração baseado em rede neural bayesiana é capaz de fornecer medições calibrada com bom nível de precisão, com as vantagens de não sofrer com o problema de sobreajuste e de fornecer estimativas das incertezas das medições calibradas de forma direta, sem a necessidade de mecanismo adicionais para o cálculo desses valores. Já com relação a limitações, o custo computacional para seu uso pode ser considerado um fator limitador, tal que o método demonstra ser mais adequado para o caso de estações sensores fixos que possuem recursos computacionais mais robustos disponíveis. Apesar disso, de forma geral, o método pode ser considerado uma opção promissora para calibração de sensores de poluição do ar de baixo-custo, com um desempenho que bate de frente com a de outros métodos de calibração baseados em técnicas de aprendizado de máquina tradicionais. Conclui-se, portanto, que o método de calibração baseado em rede neural bayesiana é aplicável para calibração de sensores de poluição do ar de baixo-custo.

5 ESTUDO DE CASO 2

5.1 Introdução

No atual processo de evolução das indústrias químicas modernas, há uma grande demanda no desenvolvimento de tecnologias que aprimorem a qualidade, eficiência e segurança de seus meios de produção. Especificamente falando sobre a segurança das indústrias químicas, um problema que ainda necessita ser solucionado é a dependência de operadores humanos nas indústrias químicas para o gerenciamento de falhas em seus processos químicos (SHU et al., 2016).

Falhas em processos químicos são definidas como desvios não permitidos em relação às condições normais de operação, os quais podem ser causados, por exemplo, pelo mau funcionamento de equipamentos e instrumentos, assim como por erros cometidos por operadores humanos (CHIANG; RUSSELL; BRAATZ, 2000a). A rápida detecção, diagnóstico e correção de falhas em processos químicos é essencial para a manutenção da segurança das indústrias químicas dado que distúrbios causados por uma única falha podem desencadear outras falhas, podendo assim tornar a operação dos processos incontrolável, causar acidentes industriais, e consequentemente gerar graves danos materiais, ambientais ou até mesmo humanos.

Por essa razão, atualmente as indústrias químicas são equipadas com sistemas de monitoramento que são utilizados para a detecção e diagnóstico de falhas em seus processos químicos. Esses sistemas auxiliam basicamente os operadores no monitoramento de variáveis de processo, permitindo que eles acompanhem o comportamento das variáveis durante a operação do processo e realizem a detecção e diagnóstico de falhas por meio da análise de comportamentos anormais das variáveis. Ou seja, eles fornecem meios para que os operadores identifiquem as causas das falhas, executem ações de correção e assim evitem maiores danos à operação dos processos químicos. No entanto, apesar da grande maioria das indústrias químicas possuírem sistemas de monitoramento implementados, ocorrências de acidentes em indústrias químicas ainda são comuns (SHU et al., 2016).

Uma das explicações para a ainda frequente ocorrência desses acidentes se deve ao fato de os sistemas de monitoramento de processos químicos ainda serem fortemente dependentes de operadores humanos para realizar a detecção e diagnóstico de falhas (SHU et al., 2016; WU; ZHAO, 2018). Essa forte dependência de operadores torna a tarefa de detecção e diagnóstico de falhas vulnerável a erros humanos e consequentemente acaba gerando a possibilidade de acidentes ocorrerem em decorrência desses erros. Nota-se,

portanto, que existe uma demanda para o desenvolvimento de sistemas de monitoramento inteligentes capazes de detectar e diagnosticar falhas em processos químicos de forma autônoma e precisa.

No sentido do desenvolvimento desses sistemas, ao longo dos últimos anos, diversos estudos sobre aplicações de métodos de aprendizado de máquina para sistemas de monitoramento de processos químicos têm ganhado notoriedade. Esses métodos têm se destacado pelo fato de permitirem que computadores analisem dados históricos de operação do processo e aprendam a realizar a detecção ou diagnóstico da falha de forma autônoma (WU; ZHAO, 2018).

Sendo assim, seguindo essa tendência, este estudo de caso tem como proposta estudar a aplicabilidade de um método baseado em rede neural bayesiana para detecção automatizada de falhas em processos químicos reais. Especificamente propõe-se estudar a aplicabilidade do método proposto por Sun et al. (2020), o qual consiste um método promissor de detecção de falhas baseado em um modelo de rede neural recorrente bayesiana. Esse método foi avaliado principalmente utilizando simulações de processos químicos e há uma escassez de estudos sobre sua aplicabilidade no caso de processos químicos reais. Dessa forma, esse caso de estudo tem como objetivo específico analisar o desempenho de um método baseado em rede neural bayesiana semelhante ao proposto por Sun et al. (2020) para a detecção de falhas em um processo químico real. Um caso específico de detecção de falha em uma unidade de craqueamento catalítico em leito fluidizado é utilizado para o desenvolvimento do estudo.

5.2 Análise bibliográfica

Ao longo das últimas décadas, diversos métodos de detecção e diagnóstico de falhas foram propostos na literatura de forma a contribuir para o desenvolvimento de sistemas de monitoramento de processos químicos. Em geral esses métodos podem ser divididos em três classes: métodos analíticos, métodos baseados em conhecimentos e métodos orientados a dados (CHIANG; RUSSELL; BRAATZ, 2000a).

Os métodos analíticos são baseados na utilização de modelos matemáticos analíticos que são construídos a partir de equações de princípios primários, tais como equações de conservação de massa e energia, e que descrevem o processo químico monitorado sob condições normais de operação. Esses modelos são geralmente utilizados para a detecção de falhas por meio do monitoramento de valores resíduos, os quais consistem em valores de discrepância entre medições reais de variáveis de processo coletadas por sensores e valores dessas mesmas variáveis estimados pelos modelos. Dessa forma, um valor resíduo elevado é considerado um sinal de ocorrência falha, enquanto que um valor resíduo baixo

é considerado um sinal de operação normal (CHIANG; RUSSELL; BRAATZ, 2000a). A grande vantagem desses métodos é o fato deles utilizarem modelos analíticos detalhados do processo que permitem obter resultados de detecção mais precisos comparado aos métodos das outras duas classes. Por outro lado, um grande empecilho para a utilização desses métodos atualmente é que, com a evolução das indústrias químicas modernas, os processos químicos estão cada vez mais complexos, o que vem tornando a modelagem matemática desses processos uma tarefa cada vez mais difícil e custosa (GE; SONG; GAO, 2013).

Os métodos baseados em conhecimento, por sua vez, baseiam-se na utilização de conhecimentos disponíveis sobre o comportamento do processo químico monitorado e da experiência de seus operadores para construção de modelos qualitativos para detecção e diagnósticos de suas falhas. Entre esses métodos, destacam-se métodos de análise causal que realizam a modelagem das relações entre sintomas identificados no processo e as falhas, e métodos baseados em sistemas especialistas (*expert systems*) capazes de diagnosticar falhas por meio de regras que simulam conhecimentos e experiências de operadores do processo (CHIANG; RUSSELL; BRAATZ, 2000a). Esses métodos tendem a ser mais intuitivos comparados aos métodos das outras classes, porém eles exigem a utilização de uma base de conhecimentos e experiências sobre o comportamento do processo monitorado, cuja criação consome muito tempo e pode ser bem difícil dependendo da complexidade do processo (GE; SONG; GAO, 2013).

Já os métodos orientados a dados baseiam-se na análise de dados históricos de operação do processo químico para desenvolver modelos de detecção e diagnóstico de falhas. Pelo fato desses métodos não exigirem a utilização de modelos analíticos do processo nem de bases de conhecimentos sobre o processo, aplicações desses métodos têm ganhado popularidade, principalmente em processos químicos complexos em que modelos matemáticos analíticos e bases de conhecimentos são extremamente difíceis de serem obtidos. Além disso, com a crescente implantação de sensores, sistemas computacionais e tecnologias da informação nas indústrias químicas, uma quantidade de dados históricos de processos químicos tem se tornado disponível para os sistemas de monitoramento, o que tem facilitado a utilização desses métodos na prática. Sendo assim, atualmente o estudo de métodos de detecção e diagnóstico de falhas orientados a dados tem se estabelecido como o foco da área de desenvolvimento de sistemas de monitoramento inteligentes para processos químicos (YIN et al., 2012; GE; SONG; GAO, 2013).

Entre os métodos orientados a dados para detecção e diagnóstico de falhas em processos químicos, os métodos baseados em técnicas de aprendizado de máquina tais como *principal component analysis* (PCA) (WISE et al., 1990), *partial least squares* (PLS)

(MACGREGOR et al., 1994), *independent component analysis* (ICA) (KANO et al., 2003), *Fisher discriminant analysis* (FDA) (CHIANG; RUSSELL; BRAATZ, 2000b), *support vector machines* (SVM) (CHIANG; KOTANCHEK; KORDON, 2004) e redes neurais artificiais (VENKATASUBRAMANIAN; CHAN, 1989) são os mais conhecidos. Esses métodos têm se destacado pelo fato de permitirem que sistemas computacionais, por meio da análise de dados históricos de operação dos processos, aprendam os padrões de operação normal e de operação em falha dos processos, e assim se tornem capazes de realizar detecções e diagnósticos de falhas de forma autônoma.

Especificamente para a detecção de falhas, os métodos baseados na técnica PCA são amplamente utilizados na indústria dado a sua simplicidade (CHENG; HE; ZHAO, 2019; SUN et al., 2020). Entretanto, apesar de sua popularidade, esses métodos apresentam limitações de só poderem ser eficientemente aplicados quando os dados do processo são distribuídos de forma gaussiana, as variáveis do processo são linearmente correlacionadas e o processo é operado sob uma única condição estacionária, condições as quais são dificilmente encontradas em processos químicos. Os dados de processos químicos são normalmente mais complexos, caracterizados por distribuições não gaussianas, relações não lineares entre variáveis de processo, variações ao longo do tempo e comportamentos dinâmicos (GE; SONG; GAO, 2013). Por essa razão, ao longo dos últimos anos, diversos esforços têm sido realizados para o desenvolvimento de métodos capazes de lidar com dados de processos químicos com essas características. Por exemplo, métodos baseados em *kernel*-PCA (KPCA) (LEE et al., 2004) foram desenvolvidos para lidar com a não linearidade das dados de processo, enquanto que métodos baseados em *dynamic*-PCA (DPCA) (KU; STORER; GEORGAKIS, 1995; DONG; QIN, 2018) foram desenvolvidos com objetivo de lidar com a dinâmica temporal do processo. Entretanto, apesar desses avanços, ainda busca-se desenvolver métodos capazes de lidar com todas as características complexas dos dados de processos químicos.

Com o recente sucesso do *deep learning* na solução de problemas complexos de diversas áreas científicas, métodos baseados em redes neurais para detecção de falhas em processos químicos também têm sido cada vez mais propostos. Aplicações de métodos baseados em modelos de redes neurais tais como *deep belief networks* (DBN) (ZHANG; ZHAO, 2017), autoencoders (LV et al., 2016; CHENG; HE; ZHAO, 2019), redes neurais convolucionais (WU; ZHAO, 2018) e redes neurais recorrentes (ZHAO; SUN; JIN, 2018; CHENG; HE; ZHAO, 2019; SUN et al., 2020) têm sido estudadas. No sentido do desenvolvimento de métodos de detecção de falhas capazes de lidar com as características complexas dos dados de processos químicos, esses métodos têm gerado grandes contribuições, apresentando principalmente uma boa capacidade na modelagem da não

linearidade e da dinâmica temporal dos processos químicos (SUN et al., 2020).

Para realizar a detecção de falhas em processos químicos, as redes neurais podem ser utilizadas em duas abordagens: como modelos de classificação ou como modelos de regressão (SUN et al., 2020). Na abordagem como modelos de classificação, as redes neurais são treinadas a receber como entrada dados de operação do processo e, de acordo com as informações desses dados, a classificar a operação do processo como normal ou em falha. Essa abordagem pode ser bem eficiente principalmente quando há disponível dados históricos de falhas que já ocorreram no processo, podendo também ser utilizada para realizar o diagnóstico dessas falhas. Por outro lado, uma grande desvantagem dessa abordagem é que para treinar as redes neurais é necessária uma quantidade considerável de dados de operação em condição de falha, os quais são normalmente escassos em comparação à quantidade de dados de operação normal. Além disso, nessa abordagem as redes neurais podem ter dificuldade de detectar falhas que nunca ocorreram no processo ou falhas não relatadas nos dados históricos utilizados para treinamento do modelo. Métodos tais como os propostos por Lv et al. (2016), Zhang e Zhao (2017), Wu e Zhao (2018) e Zhao, Sun e Jin (2018) utilizam redes neurais com essa abordagem.

Já na abordagem como modelos de regressão, as redes neurais são utilizadas como modelos de predição de variáveis de monitoramento. Nessa abordagem, os modelos de rede neurais são treinados com dados de operação normal com objetivo de torná-los capazes de prever os valores das variáveis de monitoramento do sistema em condições normais de operação a partir de valores passados das mesmas variáveis de monitoramento. Dessa forma, esses modelos após treinamento são utilizados para detecção de falhas ao comparar suas previsões com as medições reais das variáveis de saída do sistema. Uma falha no processo é detectada caso a diferença entre a previsão e a medição real for significativa, caso contrário a operação do processo é considerada normal. Essa abordagem tem a vantagem de que as redes neurais são treinadas utilizando apenas dados de operação normal, os quais são abundantes em dados históricos de processos químicos. Além disso, nessa abordagem as falhas detectáveis não se restringem apenas a falhas conhecidas. Qualquer falha que gere um desvio significativo da condição normal de operação é detectada. Essa abordagem, portanto, demonstra ser vantajosa em termos práticos. Apesar disso, especificamente para detecção de falhas em processos químicos, a utilização de redes neurais nessa abordagem é ainda menos comum na literatura, podendo ser encontrada apenas em métodos tais como os propostos por Cheng, He e Zhao (2019) e Sun et al. (2020). Por outro lado, na detecção de falhas em outras áreas, a utilização de redes neurais como modelos de regressão já é mais comum (MALHOTRA et al., 2015, 2016; WANG et al., 2017; HUNDMAN et al., 2018).

Entre métodos de detecção de falhas que utilizam redes neurais como modelos de regressão, o método proposto por Sun et al. (2020) demonstra ser um dos mais promissores. O método de Sun et al. (2020) baseia-se na utilização de um modelo rede neural recorrente bayesiana (BRNN - *Bayesian Recurrent Neural Network*), o qual é um tipo de modelo que, além de apresentar a excelente capacidade das redes neurais recorrentes de analisar dados sequenciais não lineares (tais como as séries temporais de variáveis de processos químicos), usufrui das vantajosas características das redes neurais bayesianas de apresentar mecanismos naturais de regularização que evitam problemas de sobreajuste e por fornecer diretamente os valores de incerteza de suas predições. O sobreajuste é um problema que pode afetar a aplicabilidade das redes neurais para a detecção de falhas, pois faz com que os modelos treinados percam capacidade de generalização e não performem tão bem quando realmente utilizados na indústria. Por essa razão é importante que mecanismos de regularização sejam utilizados durante o treinamento de redes neurais aplicadas para detecção de falhas. Já os valores de incerteza podem ser de grande utilidade no contexto da detecção de falhas realizada por modelos computacionais tais como as redes neurais, pois são informações que descrevem o nível de confiança de predições realizadas por esses modelos e que podem ser utilizados como informações adicionais que auxiliem na detecção das falhas.

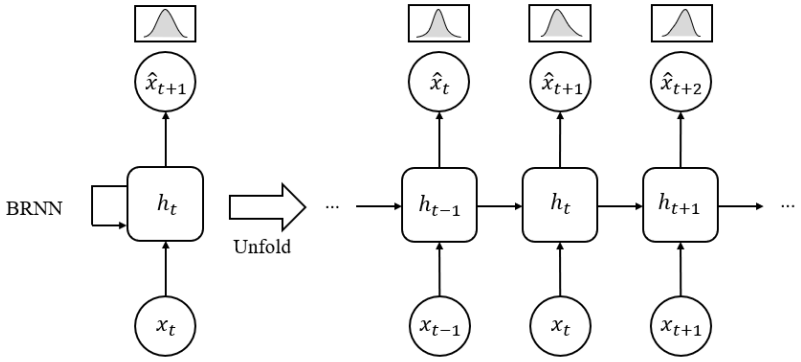
No trabalho de Sun et al. (2020), seu método de detecção de falhas baseado em BRNN foi testado em dois casos de estudo: em um extenso caso que utiliza dados gerados por uma simulação do processo químico e um outro caso mais simples que utiliza dados obtidos a partir de um processo químico real. No caso simulado, são utilizados dados gerados por uma simulação do processo químico Tennessee-Eastman (DOWNS; VOGEL, 1993), o qual é considerado um caso *benchmark* para avaliação de métodos de detecção de falhas para processos químicos. Já no caso real, são utilizados dados reais da operação de uma torre de amina. Em ambos os casos o método apresentou um bom desempenho, demonstrando ser uma opção com potencial para detecção de falhas em processos químicos reais. Apesar disso, estudos de aplicação do método para casos reais ainda são escassos na literatura. O único estudo de aplicação do método para um processo químico real encontrado na literatura é o apresentado por Sun et al. (2020). Sendo assim, este estudo de caso visa contribuir na análise da aplicabilidade do método de Sun et al. (2020) em mais um caso real.

5.3 Metodologia

Tal como foi descrito na sessão anterior, o método proposto por Sun et al. (2020) se baseia na utilização de uma BRNN como um modelo de regressão para detecção de

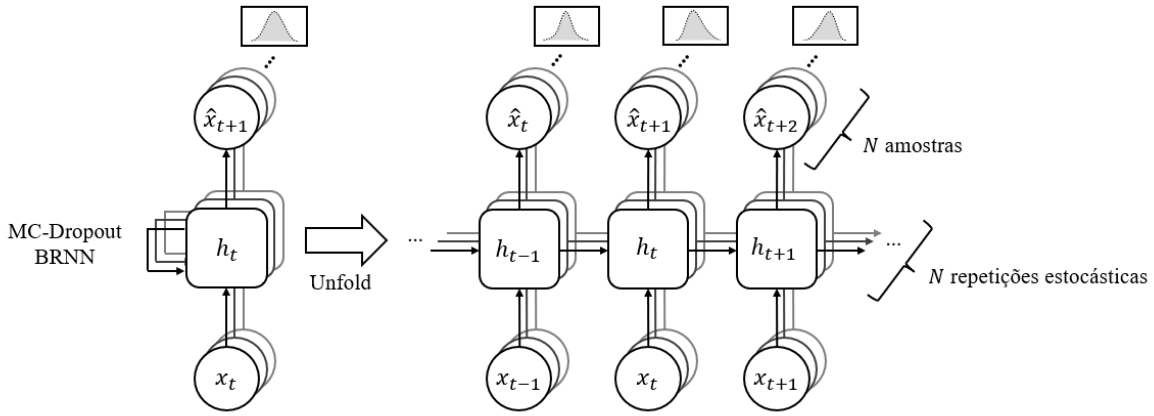
falhas em processos químicos. Tal como ilustrado na Figura 44, nesse método a BRNN é treinada a receber como entrada um dado x_t e fornecer como resposta uma predição $\hat{x}_{t+1}$, onde x_t denota um vetor contendo medições de variáveis do processo em um instante t, e $\hat{x}_{t+1}$ denota um vetor contendo valores de predição para as mesmas variáveis no instante seguinte $t+1$. Para calcular $\hat{x}_{t+1}$, a rede neural utiliza as informações contidas em x_t e informações obtidas em dados previamente analisados $[x_{t-1}, x_{t-2}, x_{t-3}, \dots]$ e que são descritas pela variável estado h_{t-1}. No entanto, pelo fato da rede neural recorrente utilizada ser bayesiana, $\hat{x}_{t+1}$ calculado pela BRNN não contém apenas valores pontuais, mas sim uma distribuição probabilística preditiva que fornece informações dos valores mais prováveis para as variáveis do processo no instante $t+1$ assim como descreve a incerteza da predição do modelo.

Figura 44: Modelo de BRNN para detecção de falhas



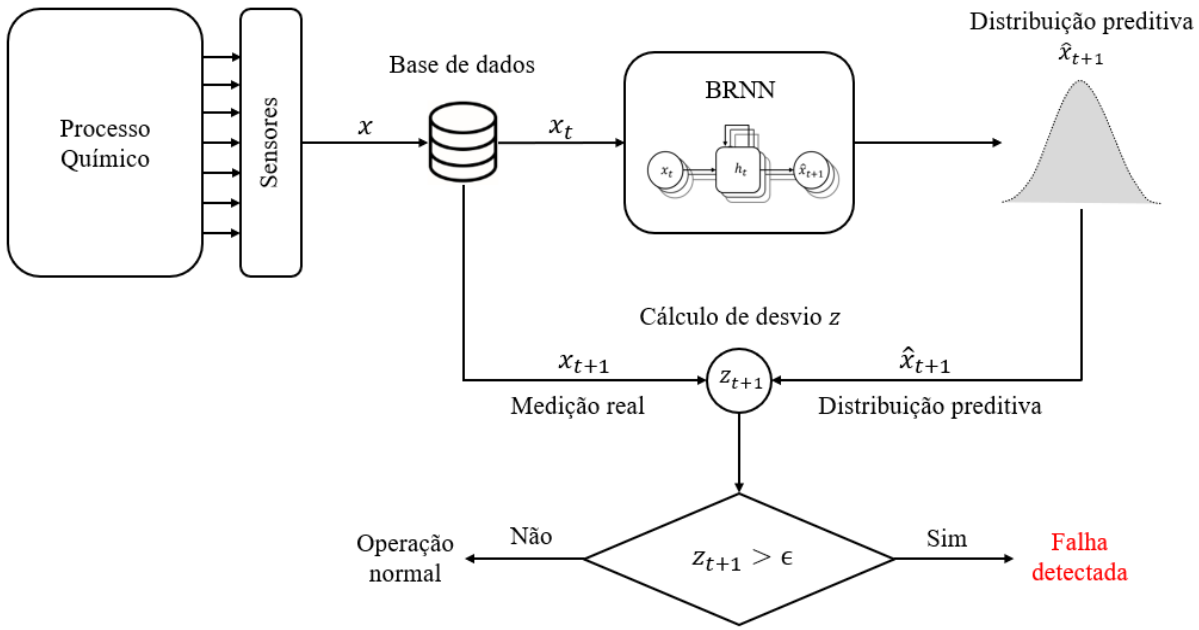
A rede neural é treinada de modo *offline* utilizando apenas dados históricos de operação normal do processo de modo que ela aprenda a predizer os valores mais prováveis de $\hat{x}_{t+1}$ em condições normais de operação do processo. Os dados de treinamento são organizados como dados rotulados, tal que para cada dado x_t, o dado x_{t+1} é designado como seu respectivo rótulo. Para o treinamento da BRNN, a técnica de aproximação de inferência bayesiana *MC-Dropout* é utilizada pelo fato de ser simples de implementar, não exigir modificação na estrutura da rede neural e garantir um boa capacidade de generalização ao modelo (GAL; GHAHRAMANI, 2016a, 2016b). Tal como ilustrado na Figura 45, ao treinar a BRNN utilizando *MC-Dropout*, para o cálculo da aproximação de $\hat{x}_{t+1}$, N repetições da propagação da entrada x_t no modelo treinado são realizadas, com uma máscara de *dropout* diferente em cada repetição. Desse modo, N amostras de Monte Carlo $\{\hat{x}_{t+1}(i)\}_{i=1,\dots,N}$ são geradas e utilizadas para aproximar a distribuição preditiva $\hat{x}_{t+1}$.

Figura 45: *MC-Dropout* BRNN para detecção de falhas



Após treinada e validada, a BRNN é então aplicada para detecção *online* de falhas seguindo o procedimento ilustrado na Figura 46. Ao receber um novo dado x_t contendo medições reais de variáveis do processo monitorado, a rede neural calcula a distribuição preditiva aproximada $\hat{x}_{t+1}$ que descreve os valores mais prováveis para as mesmas variáveis no instante seguinte $t+1$ em condição normal de operação do processo. No momento que o dado real x_{t+1} é coletado, ele é então comparado à distribuição preditiva aproximada $\hat{x}_{t+1}$ gerada pelo modelo com objetivo de analisar se há alguma discrepância. Caso o desvio z_{t+1} entre medições reais x_{t+1} e a distribuição preditiva $\hat{x}_{t+1}$ ultrapassar um certo valor limite ϵ, a ocorrência de falha no processo é detectada. Caso contrário, a operação do processo é considerada normal.

Figura 46: Procedimento para detecção de falhas utilizando BRNN

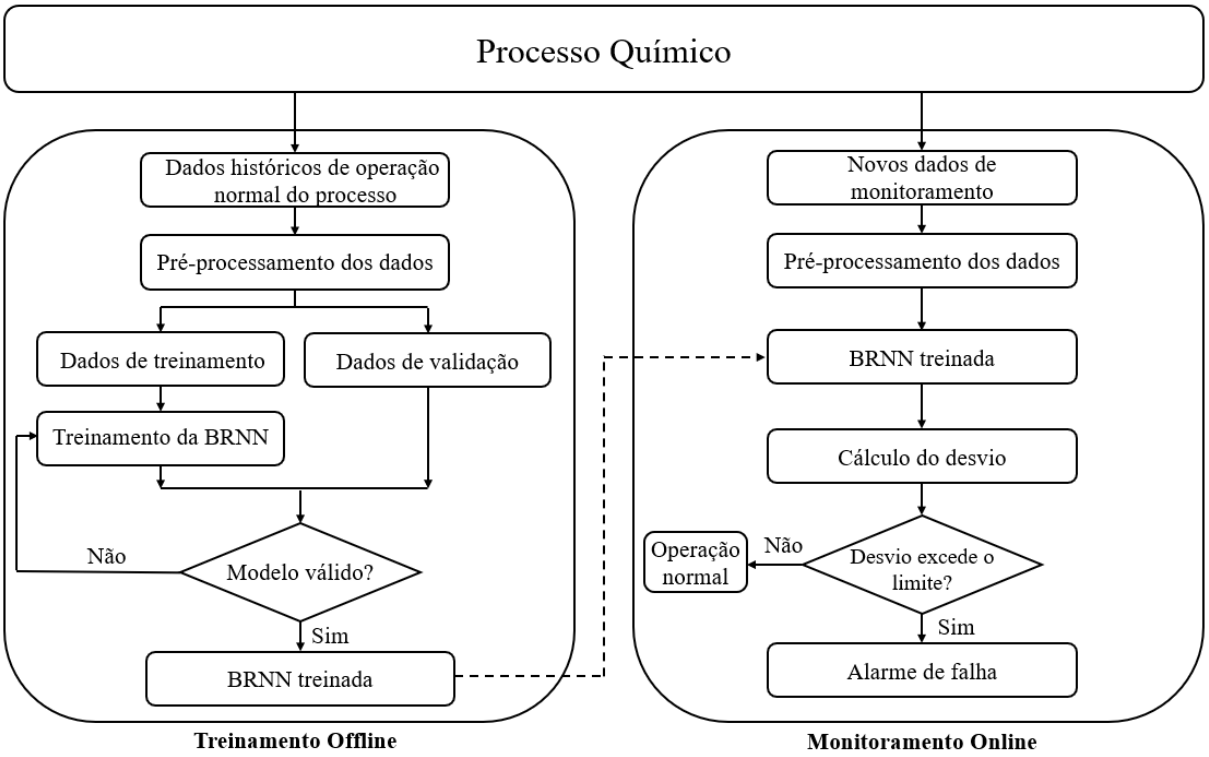


Considerando que a distribuição preditiva aproximada $\hat{x}_{t+1}$ fornecida pelo modelo é gaussiana ou é aproximadamente gaussiana, a distância quadrada de Mahalanobis (M^2) é utilizada para mensurar o desvio z_{t+1} entre x_{t+1} e $\hat{x}_{t+1}$:

$$z_{t+1} = M^2 = (x_{t+1} - \mu_{t+1})^{tr} S_{t+1}^{-1} (x_{t+1} - \mu_{t+1}) \tag{5.1}$$

Onde μ_{t+1} e S_{t+1} são respectivamente a média e covariância distribuição preditiva $\hat{x}_{t+1}$. O esquema completo para detecção de falhas em um processo químico utilizando um modelo de BRNN é ilustrado na Figura 47. Já o valor limite ϵ é definido manualmente. Nesse estudo valores no intervalo $4 < \epsilon < 9$ foram analisados.

Figura 47: Esquema da aplicação de modelo de BRNN para detecção de falhas em processos químicos



5.4 Experimento

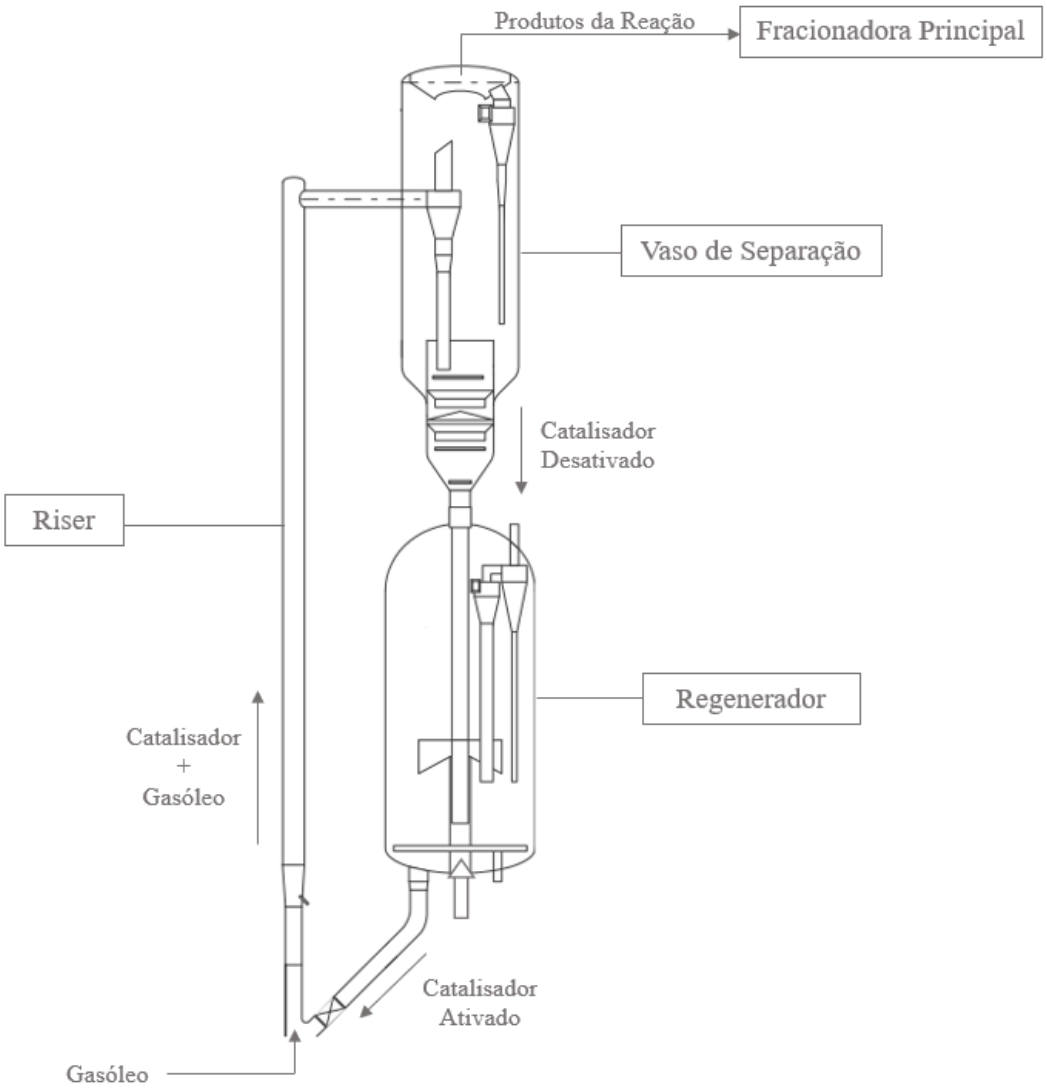
Dado a escassez de estudos sobre a aplicabilidade do método de detecção de falhas baseado em BRNN de Sun et al. (2020) para detecção de falhas em processos químicos reais, neste caso de estudo foi proposto um experimento de aplicação de uma adaptação do método apresentado por Sun et al. (2020) para a detecção de um tipo específico de falha que ocorre em uma unidade de craqueamento catalítico em leito fluidizado real. A aplicação do método é implementada utilizando dados de operação de uma unidade de craqueamento real presente em uma refinaria de petróleo brasileira. Aplicações de outros

métodos de detecção baseados em redes neurais recorrentes tradicionais (não bayesianas) também são implementados para uma análise comparativa.

5.4.1 Unidade de craqueamento catalítico em leito fluidizado FCC

Uma unidade de craqueamento catalítico em leito fluidizado, mais conhecida como unidade FCC (*Fluid Catalytic Cracking*), é responsável pela transformação de porções mais pesadas do petróleo em produtos mais leves e com maior valor agregado, tais como a gasolina e o gás liquefeito de petróleo (GLP), e por essa razão é uma das unidades mais rentáveis de uma refinaria de petróleo.

Figura 48: Exemplo de unidade FCC



Fonte: Adaptado de Sadeghbeigi (2020)

Tal como ilustrado na Figura 48, uma unidade FCC utiliza um sistema reator composto basicamente por três seções principais: o *riser* (reator tubular de fluxo ascendente), o regenerador de catalisadores e um sistema de separação. A unidade é alimentada com uma mistura de gasóleos, a qual é pré-aquecida e enviada para o *riser*. Ao ser inserida no *riser*, a mistura de gasóleo entra em contato com catalisadores aquecidos oriundos do regenerador, os quais possuem calor suficiente para vaporizar a mistura e desencadear a reação endotérmica de craqueamento catalítico. No *riser* a reação de craqueamento catalítico é desenvolvida, quebrando as moléculas mais pesadas dos gasóleos e gerando como produtos moléculas menores e mais leves. Como subproduto dessa reação, coque é depositado sobre a superfície dos catalisadores, tornando-os inativos. Do *riser*, os produtos da reação de craqueamento e os catalisadores desativados são juntos enviados para um sistema de separação, normalmente composto por múltiplos ciclones, onde são separados. Os catalisadores desativados são enviados para o regenerador, onde os catalisadores são reativados por meio de um processo de combustão do coque depositado em sua superfície e posteriormente enviados para reutilização no *riser*. Já os produtos da reação são enviados para uma fracionadora principal onde é realizada a separação dos principais produtos de uma unidade FCC, como a gasolina, o GLP, gás combustível, óleo leve de reciclo (LCO - Light Cycle Oil) e óleo decantado.

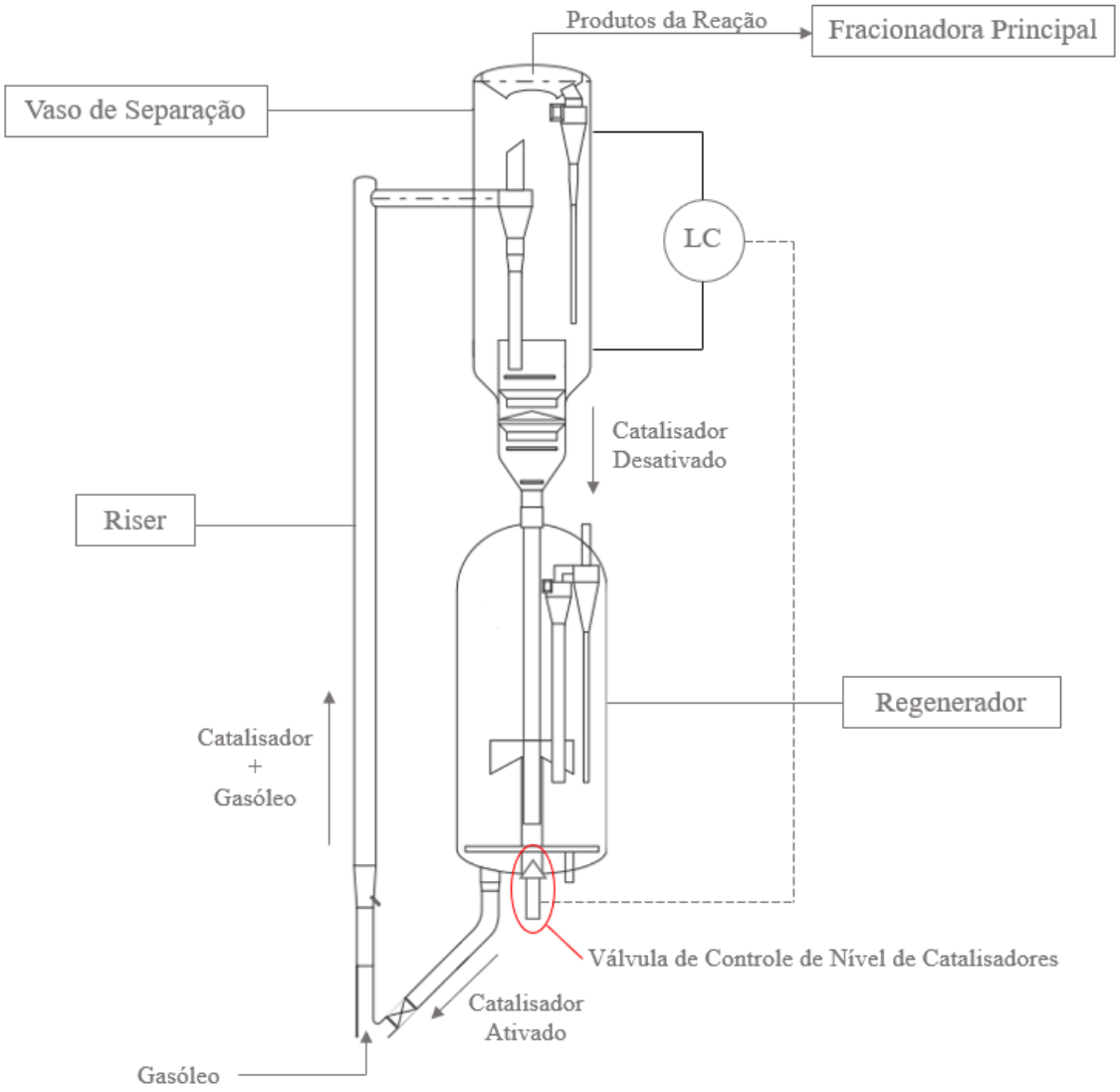
Falhas em unidades FCC são pouco frequentes, porém quando ocorrem podem gerar grandes impactos, afetando sua estabilidade e segurança operacional, além de gerar enormes prejuízos financeiros (SHU et al., 2016). Sendo assim, é de extrema importância que essas unidades possuam eficientes sistemas de monitoramento. O processo químico realizado em uma unidade FCC, no entanto, é um processo bem complexo, cujo monitoramento exige o acompanhamento e análise de um elevado número de variáveis (MCFARLANE et al., 1993). Devido a essas características, o monitoramento de uma unidade FCC é considerada uma tarefa bem desafiadora e difícil de ser realizada manualmente pelos operadores. Nesse contexto, existe um grande interesse no desenvolvimento de métodos automatizados de detecção e diagnóstico de falhas que auxiliem os operadores no monitoramento de unidades FCC (SHU et al., 2016).

5.4.2 Caso de detecção de falha em unidade FCC

Para avaliar a aplicabilidade do método proposto por Sun et al. (2020) na detecção de falhas de um processo químico real, nesse experimento então é proposto a aplicação do método para a detecção de um tipo específico de falha que ocorre uma unidade FCC real. É proposta a aplicação do método para realizar a detecção de uma falha de agarramento na válvula de controle de nível de catalisadores (destacada na Figura 49) que ocorre com

certa frequência em uma unidade FCC real presente em uma refinaria brasileira localizada no estado de São Paulo. Essa válvula de nível de catalisadores da unidade FCC atua basicamente no controle de catalisadores que fluem do vaso de separação para o regenerador. Uma falha de agarramento dessa válvula, portanto, desregula a quantidade de catalisadores utilizada no reator e consequentemente afeta a performance do processamento da unidade FCC. De acordo com os engenheiros da refinaria, a detecção dessa falha é realizada manualmente pelos operadores e o desenvolvimento de um modelo automatizado que os ajudem a detectá-la seria de grande utilidade.

Figura 49: Válvula de controle de nível de catalisadores no reator



Fonte: Adaptado de Sadeghbeigi (2020)

5.4.3 Conjuntos de dados

Para realização do experimento, três conjuntos de dados históricos de operação da unidade FCC real foco desse estudo foram disponibilizados: uma janela de dados coletados em um período de operação normal da unidade e duas janelas de dados de operação coletados períodos em que os engenheiros da unidade têm o conhecimento de ocorrências da falha de agarramento na válvula de controle de nível de catalisadores. A janela contendo dados de operação normal é utilizada no experimento como conjunto de dados de treinamento dos modelos, enquanto que as outras duas janelas são utilizadas para teste dos modelos treinados e são denominadas no experimento como conjunto de teste 1 e conjunto de teste 2. O conjunto de treinamento contém 20161 amostras de dados de variáveis de processo coletadas durante um período de 15 dias de operação normal na unidade FCC. Já o conjunto de teste 1 contém 14401 amostras de dados coletadas durante um período 11 dias de operação da unidade no qual se sabe que ocorreram falhas de agarramento na válvula, enquanto que o conjunto de teste 2 contém 12961 amostras de dados coletadas durante um período 10 dias de operação da unidade no qual se também há conhecimento da ocorrência de falhas. Nos três conjuntos de dados, as amostras foram coletadas com uma frequência de 1 minuto e cada amostra possui valores de 17 variáveis de processo da unidade FCC no momento de sua coleta. O nome e a descrição dessas 17 variáveis são apresentados na Tabela 3.

5.4.4 Seleção dos dados

De acordo com os engenheiros do processo FCC foco deste caso de estudo, a variável de processo $x^{(1)}$, que descreve o nível do vaso superior do reator, é a variável que sofre as perturbações mais perceptíveis durante a ocorrência de um problema de agarramento na válvula de controle do nível de catalisadores no reator. Tal como pode ser observado nas séries temporais da variável $x^{(1)}$ geradas a partir dos conjuntos de dados de teste 1 e 2 (ilustradas nas Figuras 50 e 51 respectivamente), nota-se que picos na série temporal da variável $x^{(1)}$ ocorrem de acordos com as ocorrências de falhas na válvula de controle de nível da coluna FCC.

Sendo assim, durante o experimento apenas os dados da variável $x^{(1)}$ foram selecionados para treinamento e teste dos modelos. Na seção de resultados do experimento é realizada uma análise da utilização de variáveis adicionais.

5.4.5 Pré-processamento dos dados

Nenhum procedimento de limpeza dos conjuntos de dados foi necessário ser realizado. Não foram encontrados dados faltantes e *outliers* são dados que não devem ser

Tabela 3: Variáveis de processo da unidade FCC

Variável	Descrição	Unidade
$x^{(1)}$	Nível de catalisadores no vaso superior	%
$x^{(2)}$	Nível de catalisadores no vaso superior - set point	%
$x^{(3)}$	Nível de catalisadores no vaso superior - variável manipulada	%
$x^{(4)}$	Feedback output	%
$x^{(5)}$	Diferença de pressão na válvula do regenerador	kgf/cm^2g
$x^{(6)}$	Diferença de pressão na válvula do regenerador - set point	kgf/cm^2g
$x^{(7)}$	Diferença de pressão na válvula do regenerador - variável manipulada	%
$x^{(8)}$	Alimentação de gasóleo pesado (GOP)	m^3/dia
$x^{(9)}$	Alimentação de nafta	m^3/dia
$x^{(10)}$	Pressão do regenerador	kgf/cm^2g
$x^{(11)}$	Pressão do reator	kgf/cm^2g
$x^{(12)}$	Temperatura de reação	°C
$x^{(13)}$	Temperatura de reação - set point	°C
$x^{(14)}$	Temperatura de reação - variável manipulada	%
$x^{(15)}$	Temperatura de alimentação	°C
$x^{(16)}$	Fluxo de ar no regenerador	kNm^3/dia
$x^{(17)}$	Circulação de catalisadores	ton/min

tratados em problemas de detecção de falhas em processos, pois eles podem fornecer informações úteis para solução dos problemas. O único procedimento de pré-processamento realizado foi a organização dos conjuntos de dados para o treinamento e teste dos modelos. Dado que apenas os dados da variável $x^{(1)}$ foram selecionados para treinamento e teste dos modelos, os conjuntos de dados treinamento e teste foram organizados de tal forma que para um dado de entrada composto por $x_t^{(1)}$, seu respectivo dado alvo é composto $x_{t+1}^{(1)}$.

5.4.6 Modelos de detecção

Para a aplicação de uma adaptação do método apresentado por Sun et al. (2020) para a detecção de falhas de agarramento na válvula na unidade FCC, um modelo de rede neural recorrente LSTM bayesiana foi implementado e treinado. Já visando a realização de um estudo comparativo, além da aplicação desse método baseado em BRNN, dois métodos de detecção de falhas baseados no uso de redes neurais recorrentes tradicionais (não bayesianas) também foram aplicados nesse experimento. Sendo assim, para o desenvolvimento da aplicação desses métodos, modelos de rede neural recorrente LSTM tradicionais e determinísticas também foram implementadas e treinadas. Descrições dos modelos utilizados são apresentadas nas subseções a seguir.

Figura 50: Série temporal da variável $x^{(1)}$ - conjunto de teste 1

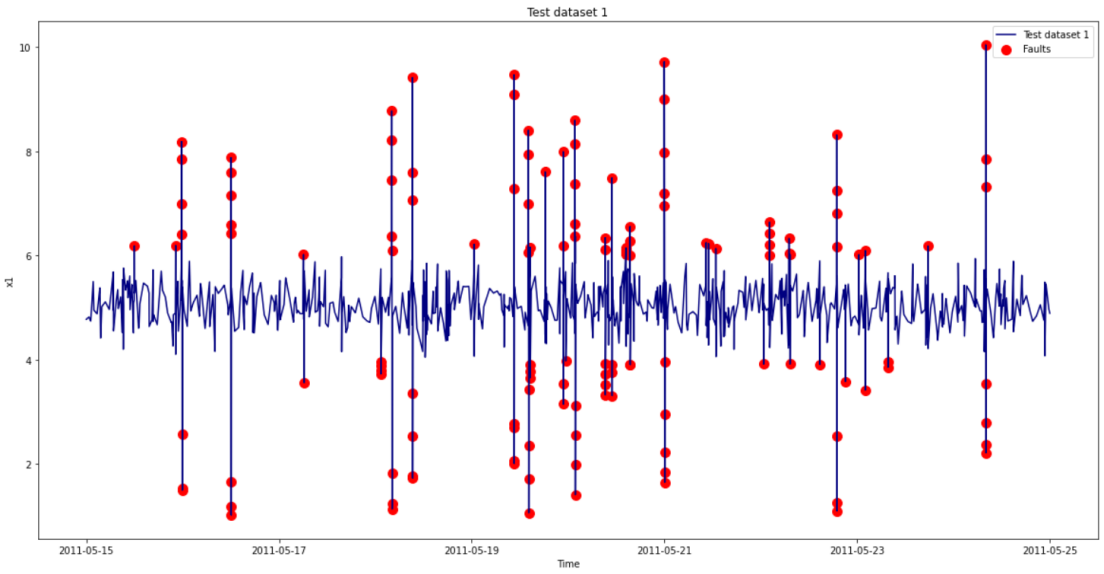
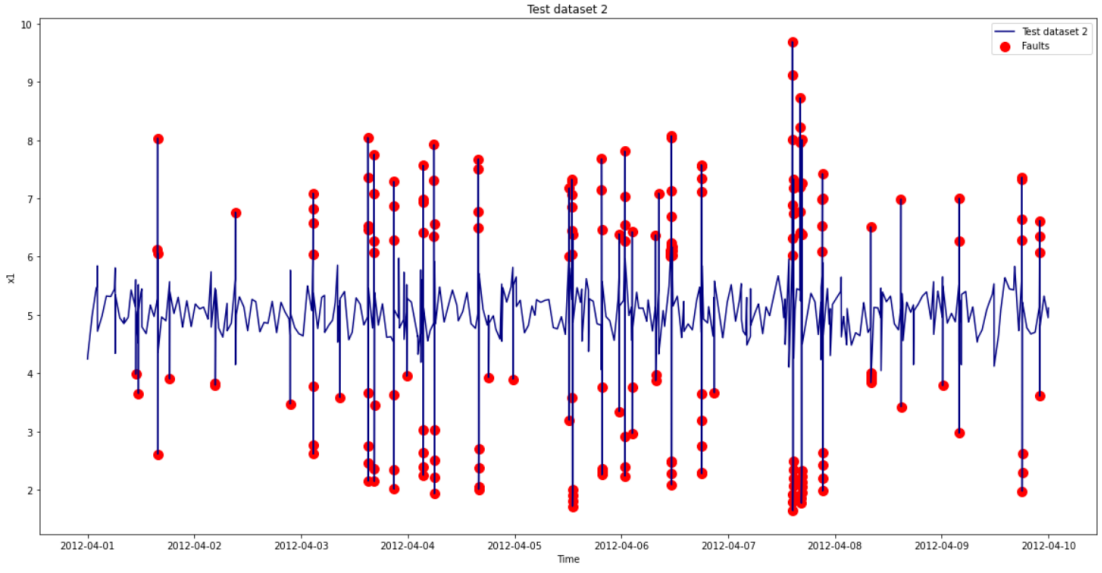


Figura 51: Série temporal da variável $x^{(1)}$ - conjunto de teste 2



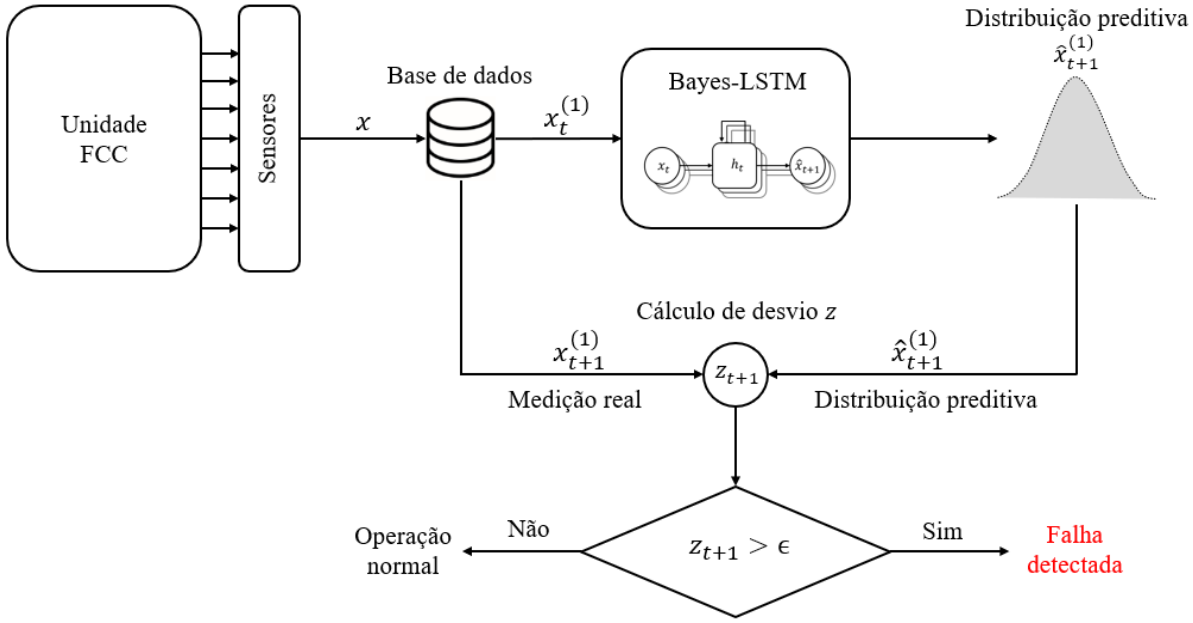
5.4.6.1 Modelo Bayes-LSTM

O método proposto por Sun et al. (2020) é descrito considerando a utilização de uma rede neural recorrente com arquitetura tradicional, porém ele pode ser facilmente adaptado para outros tipos de arquiteturas de redes neurais recorrentes, tais como as redes neurais GRU e redes neurais LSTM. Dessa forma, especificamente para esse experimento, propõe-se a aplicação de um método semelhante ao proposto por Sun et al. (2020), porém baseado na utilização de uma rede neural recorrente LSTM bayesiana (nomeada nesse experimento como Bayes-LSTM). A arquitetura de rede neural LSTM foi escolhida, dado a capacidade de redes neurais desse tipo analisarem séries temporais de dados (tal como são os dados coletados em processos químicos) e aprenderem as dependências de longas

sequências de dados, sem sofrer tanto com o problema de *gradient vanishing* (HOCHREITER; SCHMIDHUBER, 1997).

Tal como ilustrado na Figura 52, o modelo do experimento foi estruturado desse modo a permitir que ele receba como entrada valor da variável $x^{(1)}$ no instante t e retorne uma aproximação da distribuição preditiva $\hat{x}^{(1)}$ de possíveis valores da variável $x^{(1)}$ no instante $t+1$. Utilizando essa estrutura, portanto, seguindo o procedimento descrito na seção 5.3, a detecção da falha na válvula de controle de nível de catalisadores no reator é realizada a partir da comparação entre o valor real de $x^{(1)}$ no instante $t+1$ e aproximação da distribuição preditiva $\hat{x}^{(1)}$ de possíveis valores da variável $x^{(1)}$ no instante $t+1$.

Figura 52: Aplicação do modelo de Bayes-LSTM para detecção da falha na unidade FCC



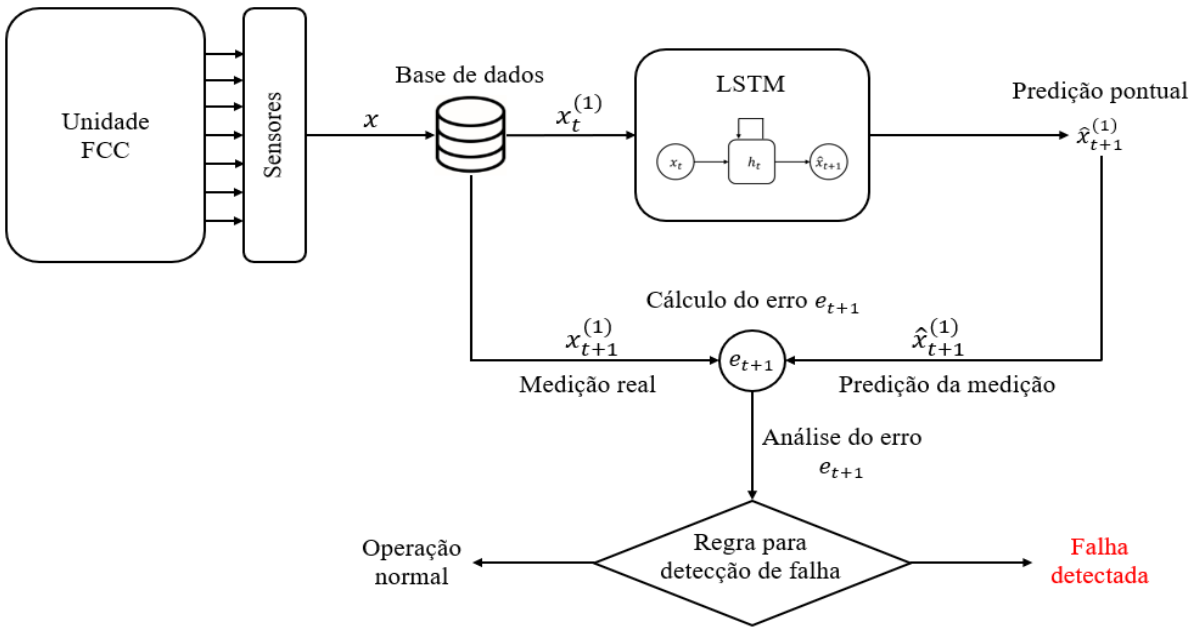
A estrutura do modelo foi otimizada manualmente e o modelo final treinado foi estruturado com uma camada oculta LSTM com 32 unidades de processamento em cada camada de processamento de seu módulo LSTM e uma camada de saída com 1 unidade processamento. Seguindo o método proposto por Sun et al. (2020), o modelo proposto é treinado utilizando apenas a janela de dados de operação normal do reator, com objetivo que ele aprenda a prever os valores da variável $x^{(1)}$ na condição de operação normal do reator. Para o treinamento do modelo, o método de aproximação de inferência bayesiana o método *MC-Dropout* foi utilizado, o qual no caso de redes neurais recorrentes consiste na aplicação do algoritmo BPTT para bateladas de sequência de dados com uma máscara de *dropout* variacional amostrada para cada sequência. Durante o processo de treinamento, foi utilizada uma taxa de *dropout* igual a 0.3 e o otimizador selecionado foi o *Adam* (KINGMA; BA, 2014), com uma taxa de aprendizado padrão igual a 0.001. O treinamento

foi realizado por 20 épocas (iterações). O modelo e todo seu procedimento de treinamento foram implementados utilizando a linguagem *Python* e a biblioteca *TensorFlow* (ABADI et al., 2016).

5.4.6.2 Modelos LSTM

Visando comparar o desempenho do método baseado no modelo de Bayes-LSTM em relação aos desempenhos de métodos de detecção de falhas baseados em redes neurais recorrentes não bayesianas, dois métodos baseados no uso de redes neurais recorrentes LSTM determinísticas também foram implementados, um método que utiliza a regra de detecção proposta por (AHMAD et al., 2017) e outro método que utiliza a regra de detecção proposta por (HUNDMAN et al., 2018). Para realizar uma comparação entre métodos de detecção de falhas com abordagens semelhantes, tal como o método baseado no modelo Bayes-LSTM, ambos os métodos baseados em modelos LSTM não bayesianos implementados utilizam as redes neurais na abordagem de regressão para realizar as detecções de falhas. Entretanto, pelo fato desses métodos serem baseados em modelos não bayesianos, seus procedimentos e regras para detecção de falhas seguem uma abordagem diferente comparada à abordagem utilizada pelo método proposto por Sun et al. (2020).

Figura 53: Aplicação do modelo de LSTM para detecção da falha na unidade FCC



Tal como ilustrado na Figura 53, para esses métodos o modelo de LSTM é treinado de modo a viabilizar que ele receba como entrada valor da variável $x^{(1)}$ no instante t e retorne uma predição numérica pontual $\hat{x}^{(1)}$ do valor esperado para a variável $x^{(1)}$ no instante $t+1$ em condição de operação normal do processo. Dessa forma, em ambos os

métodos, a detecção da falha é realizada a partir de uma regra específica para detecção de falhas baseada na análise do erro absoluto entre o valor real de $x_{t+1}^{(1)}$ no instante $t+1$ e sua predição $\hat{x}_{t+1}^{(1)}$.

- **Modelo LSTM-*Gaussian Tail***

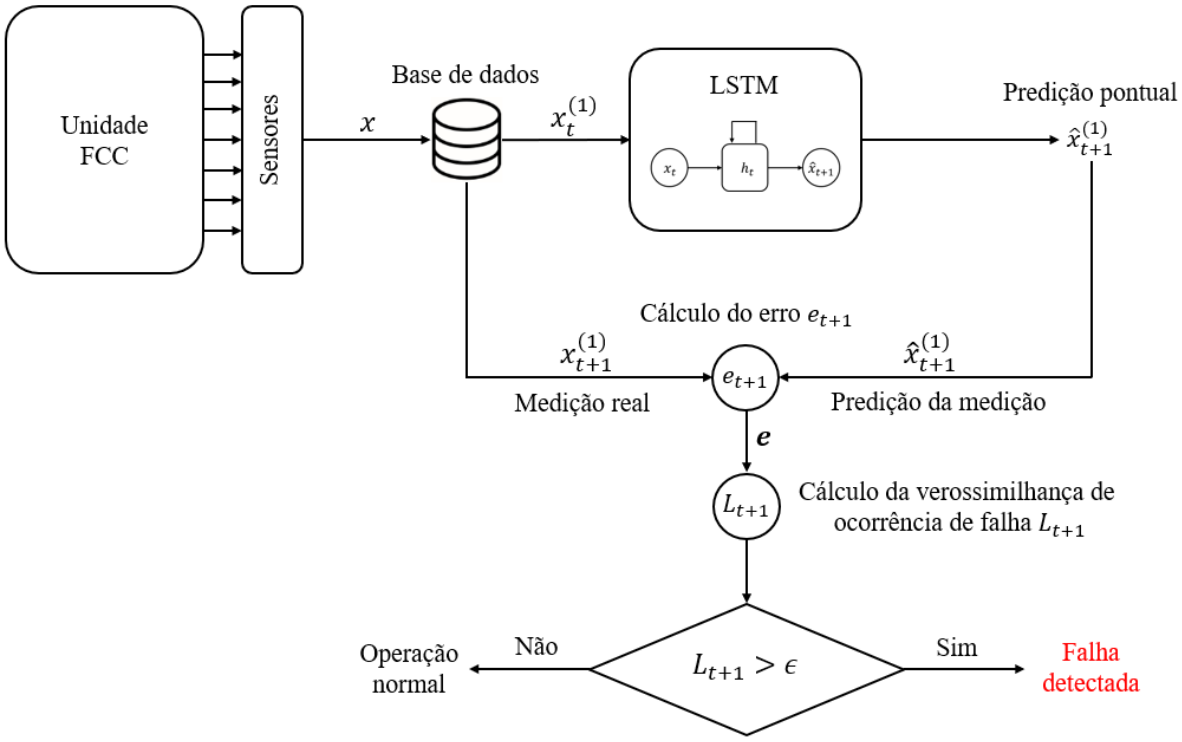
No método que utiliza a regra de detecção proposta por Ahmad et al. (2017), chamada de regra *Gaussian Tail*, o critério para detecção da falha se baseia em uma análise de comparação entre o erro calculado no instante $t+1$ e um histórico de erros calculados anteriormente. Para o desenvolvimento dessa análise comparativa, o seguinte procedimento é executado. Dado o valor real de $x_{t+1}^{(1)}$ no instante $t+1$ e sua predição $\hat{x}_{t+1}^{(1)}$, o erro e_{t+1} é calculado e inserido em um vetor de erros e_{t+1}:

$$e_{t+1} = |x_t^{(1)} - x_t^{(1)}| \tag{5.2}$$

$$\boldsymbol{e} = [e_{t+1-h}, ..., e_{t+1-l_s}, ..., e_{t-1}, e_t, e_{t+1}] \tag{5.3}$$

Onde h é o número de erros calculados anteriormente.

Figura 54: Regra *Gaussian Tail* para detecção de falhas



Em seguida, tal como ilustrado na Figura 54, utilizando esse vetor de erros $\boldsymbol{e}$, a verossimilhança de ocorrência da falha no instante $t+1$, denotada por L_{t+1}, é calculada pela função Q, a qual é a função de distribuição de cauda da distribuição normal padrão:

$$L_{t+1} = 1 - Q\left(\frac{\tilde{\mu}(\boldsymbol{e}) - \mu(\boldsymbol{e})}{\sigma(\boldsymbol{e})}\right) \tag{5.4}$$

Onde $\mu(\boldsymbol{e})$ é a média do vetor de erros $\boldsymbol{e}$ e $\sigma(\boldsymbol{e})$ é o desvio padrão do $\boldsymbol{e}$:

$$\mu(\boldsymbol{e}) = \frac{\sum_{i=0}^{i=h} e_{t+1-i}}{h+1} \tag{5.5}$$

$$\sigma(\boldsymbol{e}) = \sqrt{\frac{\sum_{i=0}^{i=h}(e_{t+1-i} - \mu(\boldsymbol{e})^2}{h}} \tag{5.6}$$

E $\tilde{\mu}(\boldsymbol{e})$ é uma média móvel de curto prazo, tal que $l_s << h$:

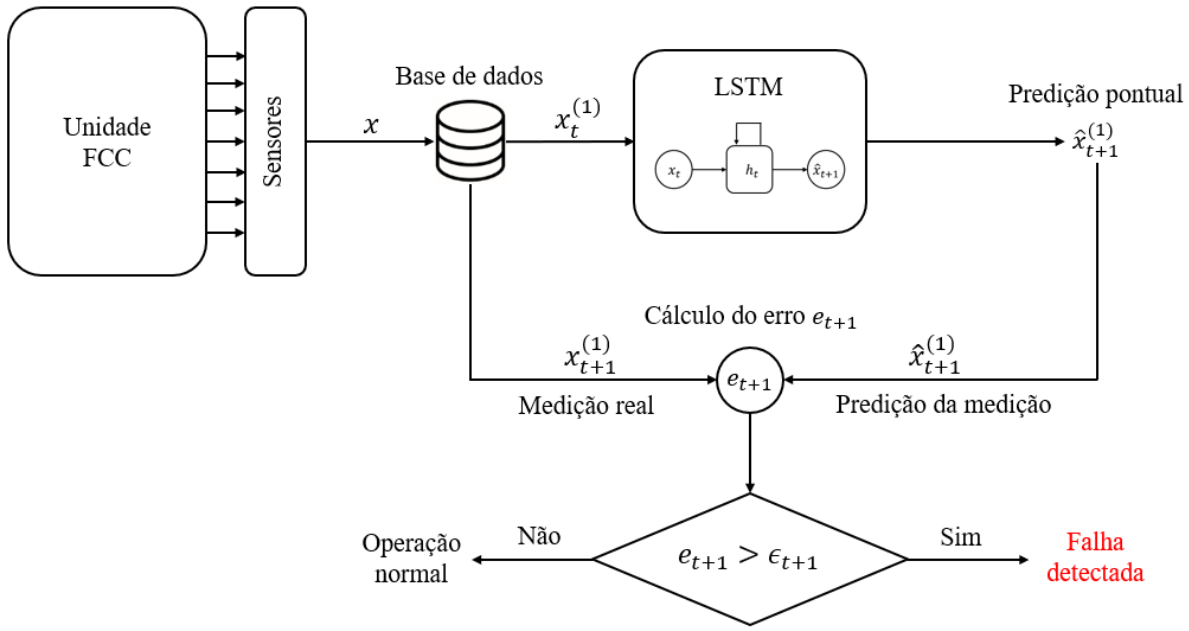
$$\tilde{\mu}(\boldsymbol{e}) = \frac{\sum_{i=0}^{i=l_s} e_{t+1-i}}{l_s+1} \tag{5.7}$$

O valor verossimilhança de ocorrência da falha no instante L_{t+1} é então comparado com um valor limite ϵ para realizar a detecção de falha. Se $L_{t+1} > \epsilon$ uma falha no instante $t+1$ é detectada. Caso contrário, a operação é considerada normal. Para esse método, o valor limite ϵ é definido manualmente. Valores no intervalo $0.99 \leq \epsilon < 1$ são normalmente testados.

- **Modelo LSTM-*Dynamic Error Thresholding***

O método baseado na regra de detecção proposta por (HUNDMAN et al., 2018), por sua vez, propõe a utilização uma técnica de limite de erro dinâmico (*dynamic error thresholding technique*). Tal como ilustrado na Figura 55, nessa técnica, dado o valor real de $x_{t+1}^{(1)}$ no instante $t+1$ e sua predição $\hat{x}_{t+1}^{(1)}$, o erro e_{t+1} é calculado e comparado a um valor limite de erro dinâmico ϵ_{t+1}. Se $e_{t+1} > \epsilon_{t+1}$ uma falha no instante $t+1$ é detectada. Caso contrário, a operação é classificada como normal. Ou seja, diferentemente do método baseado na regra de *Gaussian Tail* em que um único valor limite fixo ϵ é utilizado para realizar todas as detecções, nesse método o valor limite de erro utilizado é dinâmico e específico para cada instante de tempo.

Figura 55: Técnica de limite de erro dinâmico para detecção de falhas



Para determinação do valor limite de erro dinâmico ϵ_{t+1} para cada instante de tempo $t+1$, o seguinte procedimento é seguido. Tal como no método baseado na regra de *Gaussian Tail*, ao ser calculado, o erro e_{t+1} é inserido em um vetor de erros e_{t+1}:

$$\boldsymbol{e} = [e_{t+1-h}, ..., e_{t+1-l_s}, ..., e_{t-1}, e_t, e_{t+1}] \tag{5.8}$$

Então, diferentemente do outro método, o vetor e_{t+1} é submetido a uma função de EWMA (*Exponentially-Weighted Moving Average*) para geração de um vetor de erros suavizados $\boldsymbol{e^s}$:

$$\boldsymbol{e^s} = [e^s_{t+1-h}, ..., e^s_{t+1-l_s}, ..., e^s_{t-1}, e^s_t, e^s_{t+1}] \tag{5.9}$$

Utilizando esse vetor de erros suavizados, o valor limite de erro dinâmico ϵ_{t+1} é calculado por meio de uma espécie de problema de otimização:

$$\epsilon_{t+1} = \underset{\epsilon_{t+1}}{\operatorname{argmax}} \frac{\Delta\mu(\boldsymbol{e^s})/\mu(\boldsymbol{e^s}) + \Delta\sigma(\boldsymbol{e^s})/\sigma(\boldsymbol{e^s})}{|\boldsymbol{e^a}| + |\boldsymbol{E}^{\mathrm{seq}}|^{\mathbf{2}}} \tag{5.10}$$

Tal que:

$$\epsilon_{t+1} = \mu(\boldsymbol{e^s}) + z \cdot \sigma(\boldsymbol{e^s}) \tag{5.11}$$

$$z = \{2.5, 3.0, 3.5, ..., 10.0\} \tag{5.12}$$

$$\Delta\mu(\boldsymbol{e^s}) = \mu(\boldsymbol{e^s}) - \mu(\{e_i^s \in \boldsymbol{e^s} | e_i^s < \epsilon_{t+1}\}) \tag{5.13}$$

$$\Delta\sigma(\boldsymbol{e^s}) = \sigma(\boldsymbol{e^s}) - \sigma(\{e_i^s \in \boldsymbol{e^s} | e_i^s < \epsilon_{t+1}\}) \tag{5.14}$$

$$\boldsymbol{e^a} = \{e_i^s \in \boldsymbol{e^s} | e_i^s > \epsilon_{t+1}\} \tag{5.15}$$

$$\boldsymbol{E}^{\text{seq}} = \text{sequências contínuas de } e_i^a \in \boldsymbol{e^a} \tag{5.16}$$

As estruturas dos modelos LSTM utilizados para implementação dos dois métodos foram otimizadas manualmente. Ambos os modelos finais treinado foram estruturados com uma camada oculta LSTM com 32 unidades de processamento em cada camada de processamento de seu módulo LSTM e uma camada de saída com 1 unidade processamento. Os modelos foram treinados utilizando apenas a janela de dados de operação normal do reator, com objetivo que ele aprenda a prever os valores da variável $x^{(1)}$ na condição de operação normal do reator. O otimizador utilizado no treinamento foi o *Adam* (KINGMA; BA, 2014), com uma taxa de aprendizado padrão igual a 0.001. O treinamento foi realizado por 20 épocas (iterações). O modelo e todo seu procedimento de treinamento foram implementados utilizando a linguagem *Python* e a biblioteca *TensorFlow* (ABADI et al., 2016).

Os códigos de programa utilizados nesse estudo de caso podem ser encontrados no repositório do Github ⟨https://github.com/RyujiGT96/masters-project/tree/master/2-fcc-fault-detection⟩.

5.4.7 Métricas de avaliação

Após o treinamento, os modelos foram aplicados para as duas janelas de dados de teste visando realizar a detecção das falhas de agarramento na válvula de controle de nível de catalisadores presentes nessas janelas. Para cada um dos conjuntos de teste, utilizando suas respectivas regras de detecção, os modelos foram utilizados para classificar cada uma de suas amostras como dado de operação normal ou dado de operação com falha. Após a classificação das amostras, as predições dos modelos foram então confrontadas com as classificações reais das amostras. Para avaliar o desempenho dos modelos treinados na detecção das falhas, matrizes de confusão tal como ilustrado na Tabela 4 foram calculadas.

Tabela 4: Matriz de confusão

		Predição	
		Falha	**Normal**
Real	**Falha**	TP True Positive	FN False Negative
	Normal	FP False Positive	TN True Negative

As matrizes de confusão são compostas pelo número de verdadeiros positivos (TP), número de falsos positivos (FP), número de falsos negativos (FN) e número de verdadeiros negativos TN:

- **TP (True Positive)**: definido como o número de falhas reais detectadas corretamente.
- **FP (False Positive)**: definido como o número de falsos alarmes de falhas.
- **FN (False Negative)**: definido como o número de falhas reais não detectadas.
- **TN (True Negative)**: definido como o número de casos de operação normal detectados corretamente.

Utilizando as matrizes de confusão, diversas métricas podem ser utilizadas avaliação do desempenho dos modelos. Sendo assim, a seguintes métricas de avaliação de desempenho foram utilizadas no experimento:

- **TPR (True Positive Rate) — Sensibilidade — Recall**

Taxa de verdadeiros positivos (TPR), também conhecida como sensibilidade ou *recall*, é definida como a razão entre o número falhas detectadas corretamente e o número total de falhas reais. É uma métrica que deve ser avaliada em situações em que casos de falsos negativos são intoleráveis, tal como é a situação da detecção de falhas em processos químicos.

$$TPR = \frac{TP}{TP + FN} \tag{5.17}$$

- **TNR (True Negative Rate) — Especificidade**

Taxa de verdadeiros negativos (TNR), também conhecida como especificidade, é definida como a razão entre o número de casos de operação normal detectados corretamente e o número total de casos reais de operação normal. É uma métrica que deve ser avaliada para análise de falsos alarmes.

$$TNR = \frac{TN}{FP + TN} \tag{5.18}$$

- **Precisão**

Precisão é definida como a razão do número de falhas detectadas corretamente e o número total de falhas detectadas (corretamente e incorretamente). É uma métrica que deve ser avaliada em situações em que casos de falsos positivos são intoleráveis.

$$Prec = \frac{TP}{TP + FP} \tag{5.19}$$

- **Acurácia**

A acurácia (*accuracy*) é razão entre o número amostras categorizadas corretamente pelo modelo e o número total de amostras.

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.20}$$

- **Acurácia Balanceada**

A acurácia só uma boa métrica se o conjunto de dados for simétrico, ou seja, se o número de amostras positivas reais e o número de amostras negativas reais forem similares, o que não é o caso dos conjuntos de dados utilizados nesse experimento. Os conjuntos de dados testes utilizados possuem muito mais amostras de operação normal do que amostras de falhas. Sendo assim, dado que há um desequilíbrio entre a quantidade de dados de operação normal e a quantidade de dados de falhas, a métrica de acurácia balanceada (*bAcc*) (BRODERSEN et al., 2010) também é utilizada no experimento:

$$bAcc = \frac{TPR + TNR}{2} \tag{5.21}$$

A acurácia balanceada consiste basicamente na média aritmética entre a taxa de verdadeiro positivos (TPR) e a taxa de verdadeiros negativos (TNR).

- ***F_1 Score***

O F_1 *Score* é definido como a média harmônica da precisão e sensibilidade (TPR), e por essa razão combina a precisão e sensibilidade em uma única métrica. Sendo assim, o F_1 *Score* é considerada a métrica que permite avaliar o equilíbrio da precisão e sensibilidade dos modelos de detecção.

$$F_1 = 2\frac{Prec \cdot TPR}{Prec + TPR} \tag{5.22}$$

5.5 Resultados e discussão

5.5.1 Resultados

Os resultados das detecções de falhas realizadas pelos três modelos de detecção treinados no experimento em relação aos conjuntos de dados de teste 1 e 2 são ilustrados nas Figuras 56-61. Os resultados das detecções realizadas pelo modelo LSTM-Gaussian Tail são apresentados nas Figuras 56 e 57, os resultados das detecções do modelo LSTM-Dynamic Error Threshold são apresentados nas Figuras 58 e 59, e os resultados das detecções do modelo Bayes-LSTM são apresentados nas Figuras 60 e 61. Um resumo das métricas de desempenho apresentadas pelos modelos é apresentado na Tabela 5.

Figura 56: Resultados das detecções do modelo LSTM-Gaussian Tail - conjunto de teste 1

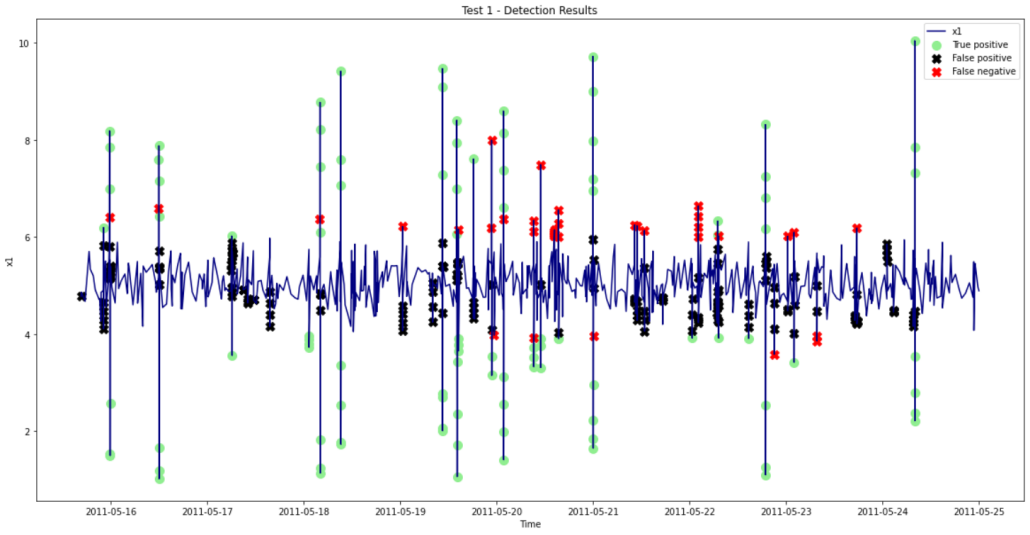


Figura 57: Resultados das detecções do modelo LSTM-Gaussian Tail - conjunto de teste 2

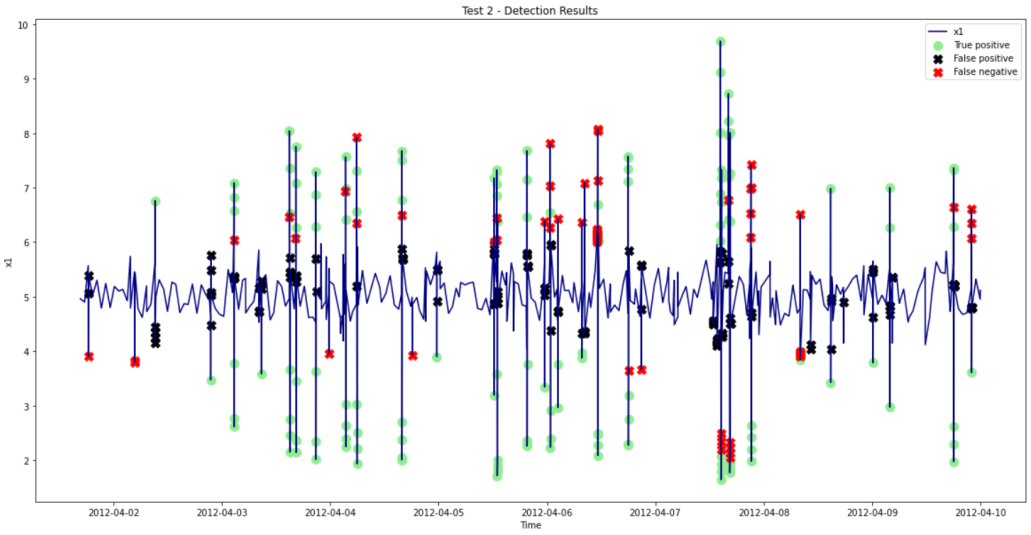


Figura 58: Resultados das detecções do modelo LSTM-Dynamic Error Threshold - conjunto de teste 1

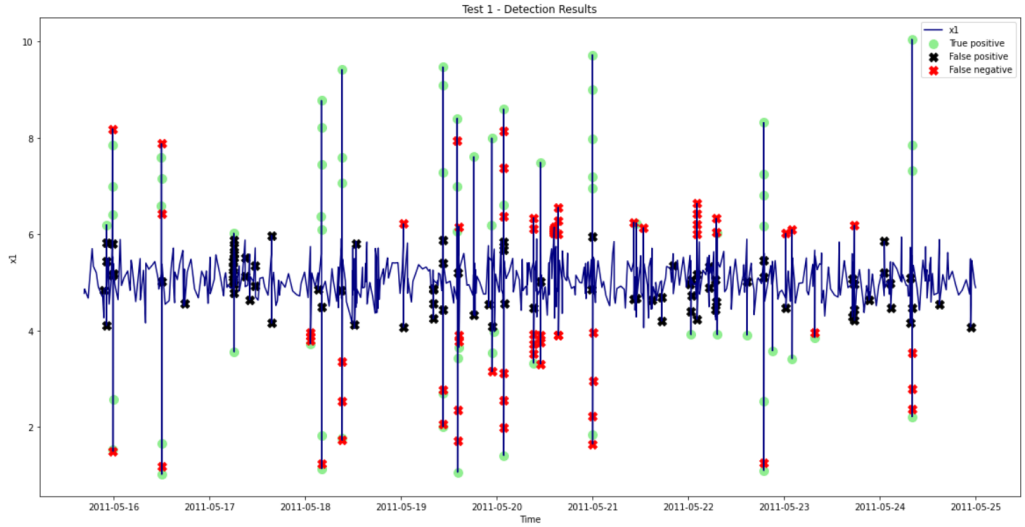


Figura 59: Resultados das detecções do modelo LSTM-Dynamic Error Threshold - conjunto de teste 2

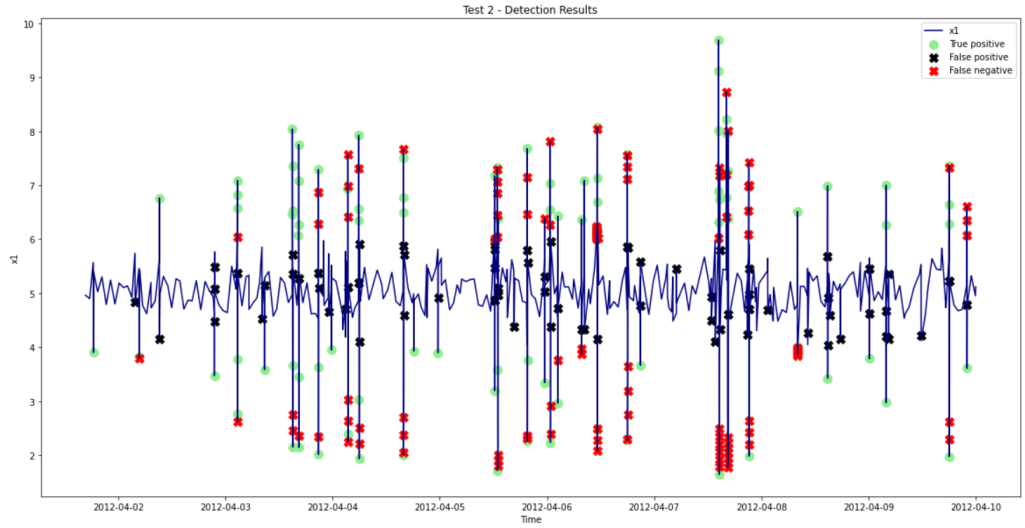


Figura 60: Resultados das detecções do modelo Bayes-LSTM - conjunto de teste 1

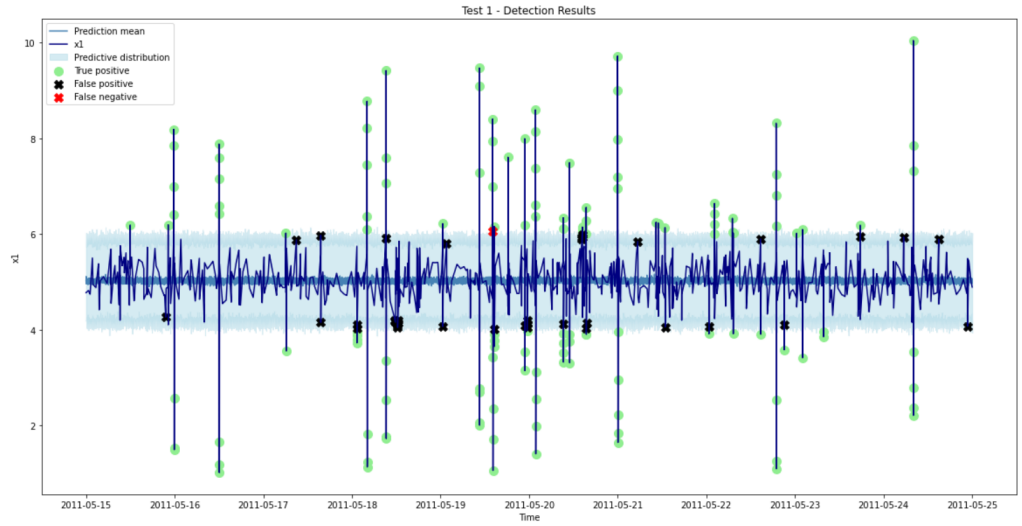


Figura 61: Resultados das detecções do modelo Bayes-LSTM - conjunto de teste 2

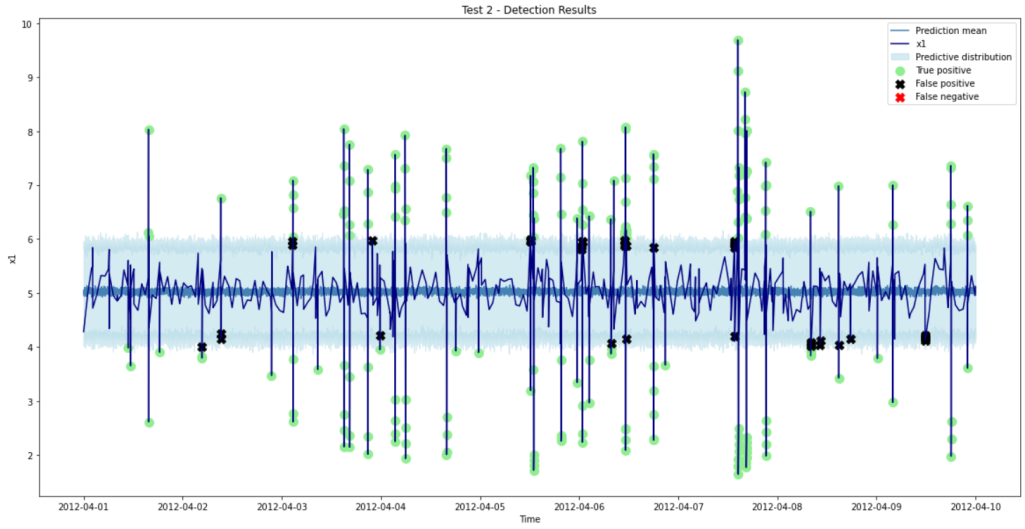


Tabela 5: Resumo dos desempemhos do modelo de detecção

Modelo de Detecção	TN	FP	FN	TP	TPR	TNR	Prec	Acc	bAcc	F_1 score
LSTM-Gaussian Tail										
Conjunto de teste 1	13115	149	36	100	0.735	0.989	0.402	0.986	0.862	0.519
Conjunto de teste 2	11651	112	59	138	0.701	0.990	0.552	0.986	0.845	0.617
Total	24766	261	95	238	0.715	0.990	0.477	0.986	0.852	0.572
LSTM-Dynamic Error Threshold										
Conjunto de teste 1	13178	86	66	70	0.515	0.994	0.449	0.989	0.754	0.479
Conjunto de teste 2	11690	73	106	91	0.462	0.994	0.555	0.985	0.728	0.504
Total	24868	159	172	161	0.483	0.994	0.503	0.987	0.739	0.493
Bayes-LSTM										
Conjunto de teste 1	14224	39	1	136	**0.993**	**0.997**	**0.777**	**0.997**	**0.995**	**0.872**
Conjunto de teste 2	12718	39	0	203	**1.000**	**0.997**	**0.839**	**0.997**	**0.998**	**0.912**
Total	26942	78	1	339	**0.997**	**0.997**	**0.813**	**0.997**	**0.997**	**0.896**

5.5.2 Análise comparativa

Comparando as métricas de desempenho dos modelos apresentadas na Tabela 5 e analisando visualmente os resultados da detecções realizadas pelos modelos por meio das Figuras 56-61, de forma geral, o modelo Bayes-LSTM utilizado no experimento foi o que apresentou o melhor desempenho na detecção das falhas de agarramento na válvula de controle de nível da unidade FCC descritas pelos dados dos conjuntos de teste 1 e 2.

Analisando primeiramente os valores de taxa de verdadeiros (TPR), o modelo Bayes-LSTM apresentou um valor de TPR total igual a 0.997, sendo o modelo que apresentou um valor de TPR mais próximo de 1. Essa métrica demonstra que o modelo deixou de detectar um número bem pequeno de falhas que ocorreram nos períodos descritos pelos conjuntos de dados de teste. Já comparando os outros dois modelos, o modelo LSTM-Gaussian Tail apresentou um valor de TPR total igual a 0.715, enquanto que o

modelo LSTM-Dynamic Error Threshold apresentou um valor igual a 0.483. Por meio desses valores nota-se, portanto, que o modelo LSTM-Gaussian Tail foi mais preciso na detecção das falhas comparado ao modelo LSTM-Dynamic Error Threshold. Analisando visualmente os resultados das detecções realizadas pelo modelo LSTM-Dynamic Error Threshold ilustrados nas Figuras 58 e 59, para ambos os conjuntos de dados de teste, um número elevado de falhas que ocorreram nos períodos descritos pelos conjuntos de dados não foram detectados pelo modelo, o que explica seu baixo valor de TPR.

Comparando os valores de taxa de negativos (TNR), o modelo Bayes-LSTM foi o modelo que apresentou um valor de TPR mais próximo de 1, com um valor de TNR total igual a 0.997, mostrando que o modelo foi capaz de gerar um número de falsos alarmes de falhas bem reduzido para ambos os conjuntos de dados de teste. Já os outros dois modelos também apresentaram valores de TNR totais bem próximos de 1, porém pelo fato de terem apresentado um número mais elevado de falsos alarmes, seus valores de TNR foram menores comparado ao valor apresentado pelo modelo Bayes-LSTM. Comparando o modelo LSTM-Gaussian Tail e o modelo LSTM-Dynamic Error Threshold, o modelo LSTM-Dynamic Error Threshold foi o que apresentou valor de TNR mais elevado. Ou seja, apesar do modelo LSTM-Dynamic Error Threshold ter sido menos efetivo na detecção das falhas comparado ao modelo LSTM-Gaussian Tail, ele foi melhor em termos de uma menor geração de falsos alarmes.

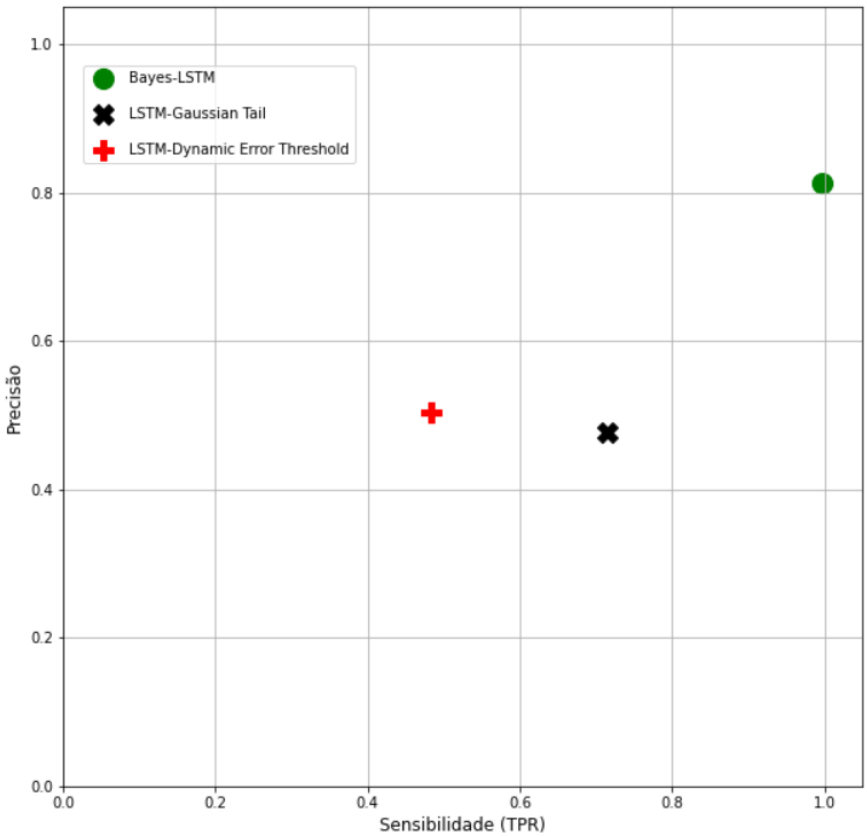
Com relação à precisão dos modelos, o modelo Bayes-LSTM foi o modelo com maior precisão, com um valor igual a 0.813, seguido do modelo LSTM-Dynamic Error Threshold com um valor igual a 0.503, e do modelo LSTM-Gaussian Tail com um valor igual a 0.477. O elevado valor de precisão do modelo Bayes-LSTM demonstra que, entre as falhas detectadas por ele, sua grande maioria eram falhas que realmente ocorreram e apenas um número bem reduzido de falsos alarmes eram gerados. Já comparando o modelo LSTM-Gaussian Tail e o modelo LSTM-Dynamic Error Threshold, o modelo LSTM-Dynamic Error Threshold apresenta uma melhor precisão dado sua capacidade de gerar uma proporção menor de falsos alarmes, comprovada pelo seu valor de TNR total.

Sobre a acurácia dos modelos, todos eles apresentaram valores elevados de acurácia, sendo o modelo Bayes-LSTM com o valor mais elevado. Já os outros dois modelos apresentaram valores de acurácia bem similares. Valores elevados de acurácia foram apresentados por todos os modelos pelo fato deles terem sido capazes de classificar corretamente a maioria dos dados de operação normal, os quais são predominantes nos conjuntos de dados de teste. Já avaliando a acurácia balanceada dos modelos, a qual é uma métrica que permite uma melhor avaliação da acurácia de modelos no caso da utilização de conjuntos de dados desiquilibrados, o modelo Bayes-LSTM apresentou a acurácia balanceada

mais elevada, com um valor igual a 0.997. O modelo LSTM-Gaussian Tail apresentou o valor de acurácia balanceada igual 0.852, o qual é maior que o apresentado pelo modelo LSTM-Dynamic Error Threshold, igual a 0.739. Isso se deve principalmente ao fato do modelo LSTM-Gaussian Tail ter apresentado um valor de TPR bem mais elevado que o apresentado pelo modelo LSTM-Dynamic Error Threshold.

Por fim, analisando os valores totais de F_1-score apresentados pelos modelos, novamente o modelo Bayes-LSTM com o valor mais elevado, seguido do modelo LSTM-Gaussian Tail e do modelo LSTM-Dynamic Error Threshold. Pelo fato ser definido como a média harmônica da precisão e sensibilidade (TPR), o F_1-*score* é uma métrica que combina a precisão e a sensibilidade dos modelos em uma única métrica. Dessa forma, ele permite avaliar o equilíbrio entre a precisão e sensibilidade dos modelos, ou seja, ele permite a analisar quão bem o modelo detectou as falhas que realmente ocorrerão e quão bem o modelo foi em termos da não geração de falsos alarmes. Sendo assim, pelos valores de F_1-score apresentados pelos modelos, o modelo Bayes-LSTM demonstra ser o mais equilibrado em termos de precisão e a sensibilidade, seguido do modelo LSTM-Gaussian Tail, enquanto que o modelo LSTM-Dynamic Error Threshold é o menos equilibrado dentre os três. Uma comparação gráfica da relação entre precisão e sensibilidade dos modelos pode ser visualizado na Figura 62.

Figura 62: Comparação gráfica da relação entre precisão e sensibilidade dos modelos



Com base nessa análise comparativa, portanto, conclui-se que dentre os três modelos de detecção testados, o modelo Bayes-LSTM foi o que melhor detectou as falhas de agarramento que ocorrem na válvula de controle de nível de catalisadores de uma unidade FCC, sendo capaz de realizar as detecções de forma precisa e com bom nível de sensibilidade. Esse melhor desempenho do modelo Bayes-LSTM, comparado aos outros modelos testados, pode ser explicado pela abordagem probabilística do método para detecção de falhas, a qual utiliza as incertezas para realizar melhor detecções. Nessa abordagem, pelo fato do modelo de detecção ser baseado em uma rede neural recorrente bayesiana e estimar continuamente a incerteza de suas predições, o modelo fornece intervalos de confiança adaptativos que caracterizam totalmente a dinâmica do sistema com base nas informações atuais e passadas. Desse modo, utilizando esses intervalos de confiança adaptativos, as detecções são realizadas de forma mais precisa e sensível.

Além disso, uma outra vantagem notada do modelo Bayes-LSTM comparado aos outros modelos testados é o fato dele não precisar de um histórico de predições e erros para realizar detecções. Para realizar detecções, os outros dois modelos realizam a comparação do erro da predição calculada por eles com relação a erros calculados anteriormente, enquanto o modelo Bayes-LSTM depende apenas da comparação entre a distribuição preditiva calculada por ele e o dado real coletado para realizar detecções. A não necessidade de um histórico de predições e erros para realizar detecções é vantajoso, pois para realizar sua primeira detecção, o modelo não precisa acumular um número específico de predições e erros calculados anteriormente, evitando assim problemas de *delay*.

Com relação ao custo computacional para implementação, o custo para treinamento do modelo Bayes-LSTM é maior que os custos do outros modelos, pelo fato dele utilizar um método de aproximação de inferência bayesiana. Entretanto, com relação à operação realizada para detecção de uma falha, o modelo Bayes-LSTM não é o mais custoso. O modelo Bayes-LSTM é mais custoso que modelo LSTM-Gaussian Tail, mas menos custoso que LSTM-Dynamic Error Threshold, dado que esse último modelo necessita solucionar um problema de otimização para realizar cada operação de detecção de falha.

Já com relação a uma limitação do modelo Bayes-LSTM, acredita-se que modelo pode apresentar problemas quando os dados de operação normal do processo químico não são estacionários, ou seja, quando o processo possui mais de uma configuração de operação normal. Devido sua abordagem de treinamento em que dados de operação normal do processo químico são utilizados para treinamento do modelo, é esperado que ele aprenda a modelar apenas as condições de operação normal descritas nos dados de treinamento. Caso a condições de operação normal se alterem, o modelo não será capaz de predizer corretamente as novas condições de operação normal e consequentemente o

inviabiliza de realizar detecções de forma precisa. No caso estudado no experimento, as condições de operação normal da variável $x^{(1)}$ eram estacionárias e por essa razão o modelo não teve problemas. Para solução dessa limitação, uma opção seria a identificação pelos operadores de todas as configurações de operação normal do processo químico e a utilização de dados de todas essas configurações para treinamento do modelo, porém esse tornaria o método dependente de operadores humanos. Desse modo, uma opção promissora seria o desenvolvimento de uma adaptação do método em que os modelos sejam treinados em uma abordagem *online*, viabilizando assim que os modelos continuamente aprendam quais são as condições operação do processo.

5.5.3 Análise comparativa com modelo multivariável

Analisando a correlação das variáveis dos conjuntos de dados disponibilizados para o desenvolvimento do experimento, notou-se que algumas variáveis possuem certa correlação com a variável $x^{(1)}$ e possivelmente poderiam auxiliar na detecção de falhas na válvula. Por meio da análise das das matrizes de correlação dos conjuntos de dados de teste, ilustrados nas Figuras 63 e 64, verificou-se que as variáveis $x^{(3)}$ e $x^{(5)}$ possuem correlações significativas com a variável $x^{(1)}$.

Figura 63: Matriz de correlação - conjunto de teste 1

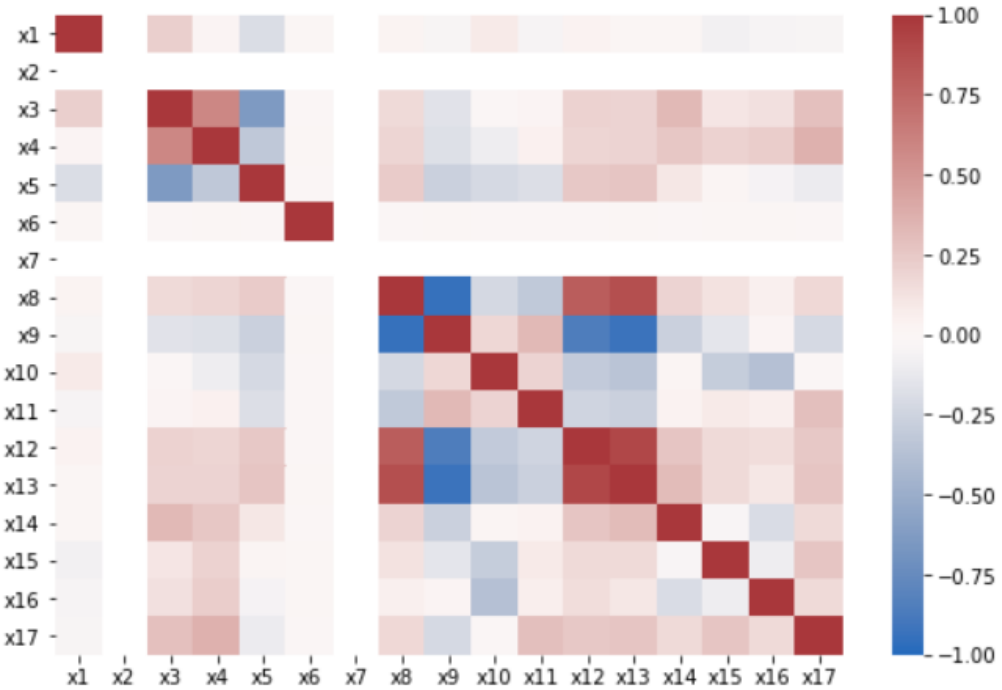
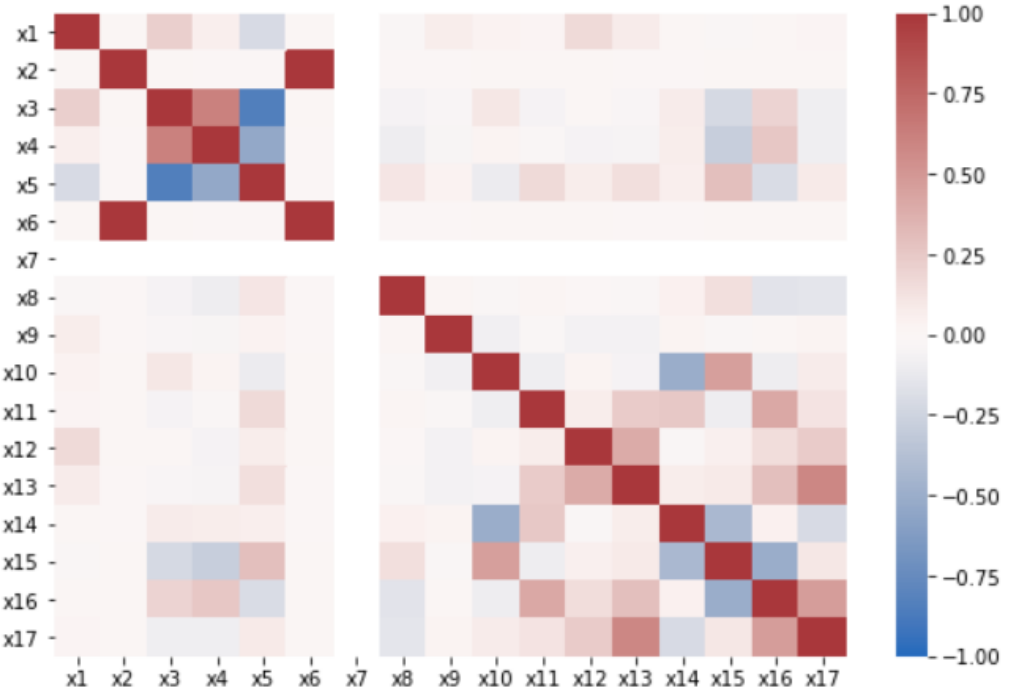


Figura 64: Matriz de correlação - conjunto de teste 2



Tal como pode ser visto nas Figuras 65 e 66, ao confrontar as séries temporais das variáveis $x^{(1)}$, $x^{(3)}$ e $x^{(5)}$ dos conjuntos de dados de testes 1 e 2, nota-se que os picos na série da variável $x^{(1)}$ são acompanhados por picos nas variáveis $x^{(3)}$ e $x^{(5)}$.

Figura 65: Séries temporais das variáveis $x^{(1)}$, $x^{(3)}$ e $x^{(5)}$ - conjunto de teste 1

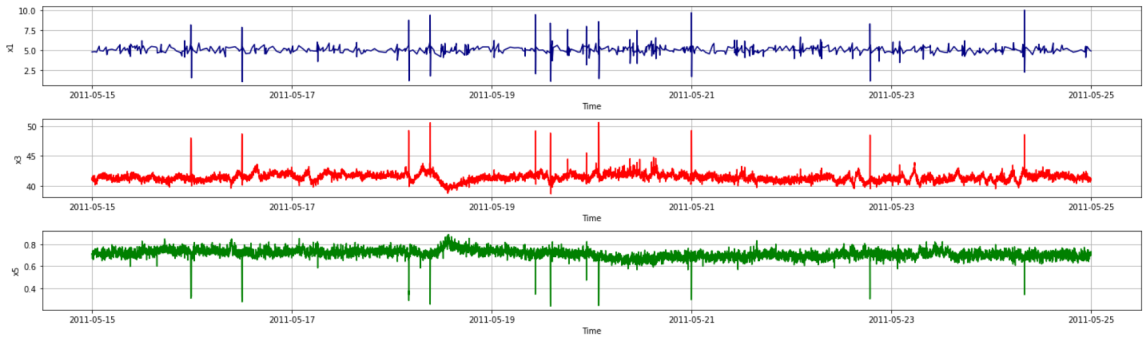
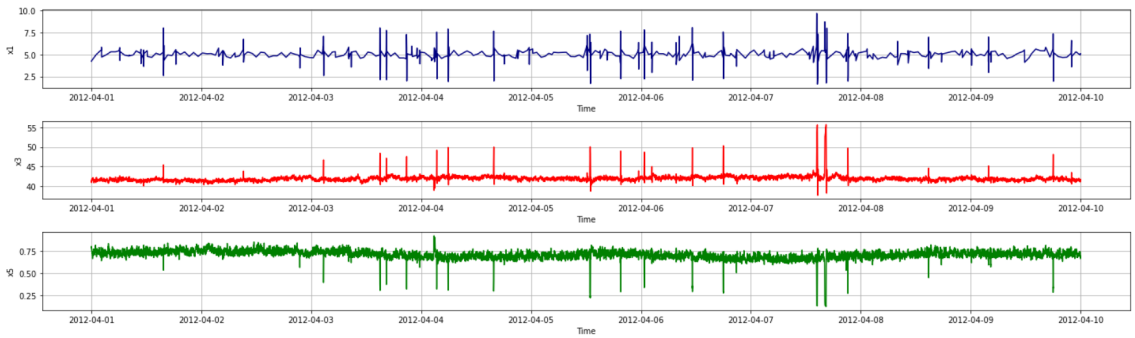
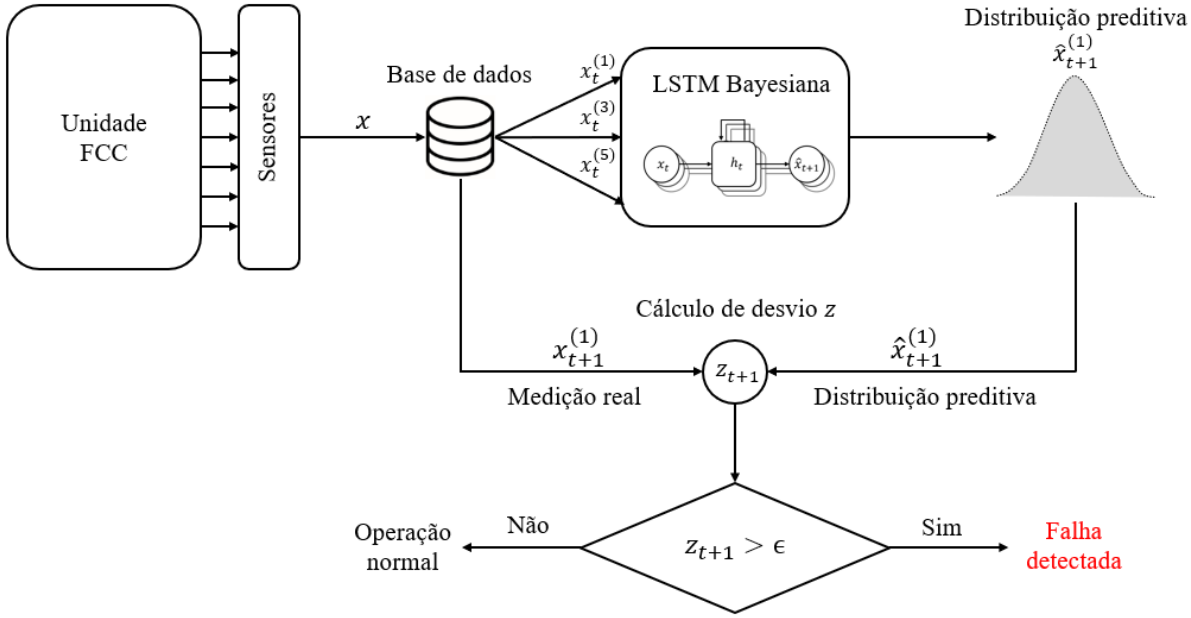


Figura 66: Séries temporais das variáveis $x^{(1)}$, $x^{(3)}$ e $x^{(5)}$ - conjunto de teste 2



Sendo assim, como uma análise extra ao experimento, realizou-se teste de um modelo semelhante ao modelo Bayes-LSTM, porém com uma abordagem multivariável. Tal como ilustrado na Figura 67, esse modelo é estruturado desse modo a permitir que ele receba como entrada valores das variáveis $x^{(1)}$, $x^{(3)}$ e $x^{(5)}$ no instante t e retorne uma aproximação da distribuição preditiva $\hat{x}^{(1)}$ de possíveis valores da variável $x^{(1)}$ no instante $t+1$. Ou seja, utilizando essa estrutura, o modelo utiliza informações das variáveis $x^{(1)}$, $x^{(3)}$ e $x^{(5)}$ no instante t para geração de uma aproximação da distribuição preditiva $\hat{x}^{(1)}$ de possíveis valores da variável $x^{(1)}$ no instante $t+1$, a qual, igualmente ao modelo Bayes-LSTM, é comparada ao valor real de $x^{(1)}$ no instante $t+1$ para realizar a detecção da falha.

Figura 67: Estrutura do modelo de Bayes-LSTM multivariável para detecção da falha na unidade FCC



Uma comparação das métricas de desempenho desse modelo em relação ao desempenho do modelo Bayes-LSTM multivariável é apresentada na Tabela 6.

Tabela 6: Comparção dos desempemhos do modelo de detecção

Modelo de Detecção	TN	FP	FN	TP	TPR	TNR	Prec	Acc	bAcc	F_1 score
Bayes-LSTM univariável										
Conjunto de teste 1	14224	39	1	136	**0.993**	**0.997**	**0.777**	**0.997**	**0.995**	**0.872**
Conjunto de teste 2	12718	39	0	203	**1.000**	**0.997**	**0.839**	**0.997**	**0.998**	**0.912**
Total	26942	78	1	339	**0.997**	**0.997**	**0.813**	**0.997**	**0.997**	**0.896**
Bayes-LSTM multivariável										
Conjunto de teste 1	14158	105	1	136	0.993	0.993	0.564	0.993	0.993	0.720
Conjunto de teste 2	12646	111	1	202	0.995	0.991	0.645	0.991	0.993	0.783
Total	26804	216	2	338	0.994	0.992	0.610	0.992	0.993	0.756

Analisando a Tabela 6, nota-se que o modelo Bayes-LSTM univariável em geral apresentou um desempenho melhor comparado ao modelo Bayes-LSTM multivariável. Comparando suas métricas de desempenho, nota-se que modelo Bayes-LSTM multivariável gerou um número mais elevado de falsos alarmes e também deixou de detectar um número maior falhas. Com base nessa análise extra ao experimento, conclui-se que a adição das variáveis $x^{(3)}$ e $x^{(5)}$ para o treinamento do modelo Bayes-LSTM não gera melhorias em termos de desempenho na detecção das falhas.

5.6 Conclusões

Visando estudar a aplicabilidade do método baseado em BRNN proposto por Sun et al. (2020) para detecção de falhas em processos químicos reais, este estudo de caso propôs a realização de um experimento de aplicação desse método para detecção de falhas de agarramento que ocorrem na válvula de controle de nível de catalisadores de uma unidade FCC real presente em uma refinaria brasileira. Os resultados do experimento mostram que a aplicação do método obteve um excelente desempenho na detecção das falhas de agarramento que ocorreram nos períodos descritos pelos conjuntos de dados teste, sendo capaz de detectar a maioria das falhas, gerando um número bem reduzido de falsos alarmes. Comparado aos outros métodos testados, o método baseado em BRNN, utilizando sua abordagem probabilística de detecção de falhas, mostrou ser capaz de realizar detecções automatizadas mais sensíveis e robustas das falhas de agarramento que ocorrem na válvula de controle de nível de catalisadores de uma unidade FCC. Já com relação a limitações, o método baseado em BRNN ainda é demonstra ser dependente de dados de operação normal estacionários para performar bem, porém existem alternativas para futuramente eliminar essas limitações. Sendo assim, de forma geral, o método prova ser um método com bom potencial de aplicação para processos químicos reais.

6 CONCLUSÃO

6.1 Conclusões finais

Visando provar a aplicabilidade de modelos de redes neurais bayesianas para solução de problemas reais, dois estudos de casos foram desenvolvidos neste trabalho. O primeiro caso consistiu no estudo da aplicação de um modelo de rede neural bayesiana para a solução de um problema de calibração de sensores de poluição do ar de baixo-custo. Já o segundo caso consistiu no estudo da aplicação de um modelo de rede neural bayesiana para a solução de um problema de detecção de falhas em um processo químico real.

No estudo de caso 1, um novo método de calibração de sensores de poluição do ar foi apresentado e um experimento de aplicação desse método foi realizado utilizando um conjunto de dados *benchmark*. O experimento de aplicação mostrou que um modelo de calibração baseado em rede neural bayesiana é capaz de fornecer medições calibradas com bom nível de precisão, com as vantagens de não sofrer com o problema de sobreajuste e de fornecer estimativas das incertezas das medições calibradas de forma direta, sem a necessidade de mecanismo adicionais para o cálculo desses valores. De forma geral, o método demonstrou ser uma opção promissora para calibração de sensores de poluição do ar de baixo-custo, com um desempenho que tão bom quanto o de outros métodos de calibração baseados em técnicas de aprendizado de máquina tradicionais. Como resultado desse estudo caso, portanto, concluiu-se que o método de calibração baseado em rede neural bayesiana é aplicável para calibração de sensores de poluição do ar de baixo-custo.

Já no segundo caso 2, visando estudar a aplicabilidade de um método baseado em rede neural bayesiana para detecção de falhas em processos químicos reais, realizou-se um experimento de aplicação do método baseado em BRNN proposto por Sun et al. (2020), no qual o método foi aplicado detecção de falhas de agarramento que ocorrem na válvula de controle de nível de catalisadores de uma unidade FCC real presente em uma refinaria brasileira. Os resultados do experimento mostraram que a aplicação do método obteve um excelente desempenho na detecção das falhas de agarramento que ocorreram nos períodos descritos pelos conjuntos de dados de teste, sendo capaz de detectar a maioria das falhas, gerando um número bem reduzido de alarmes falsos. Comparado aos outros métodos testados, o método baseado em BRNN, utilizando sua abordagem probabilística de detecção de falhas, mostrou ser capaz de realizar detecções automatizadas mais sensíveis e robustas das falhas de agarramento que ocorrem na válvula de controle da unidade FCC estudada. Já com relação a limitações, o método baseado em BRNN ainda é demonstra ser depen-

dente de dados de operação normal estacionários para performar bem, porém existem soluções alternativas para essa limitação. Sendo assim, de forma geral, o método prova ser um método com bom potencial para detecção automatizada de falhas em processos químicos reais.

Sendo assim, como conclusão geral, em ambos os estudos de caso, as redes neurais bayesianas foram aplicadas com sucesso, demonstrando assim o potencial do uso de modelos desse tipo para solução de problemas reais.

6.2 Recomendações para tabalhos futuros

- **Estudo de caso 1**

Com relação a trabalhos futuros relacionados ao estudo de caso 1, um estudo da aplicação desse método de calibração para a calibração de medições de um sensor em operação atualmente precisa ser realizado para analisar o desempenho do método em uma situação prática real. Além disso, estudos de adaptações do método baseados nos usos de redes neurais bayesianas com outras arquiteturas, tal como redes neurais recorrentes bayesianas e redes neurais convolucionais bayesianas, precisam ser realizadas.

- **Estudo de caso 2**

Com relação a trabalhos futuros relacionados ao estudo de caso 2, visando acabar com a limitação do método de detecção baseado em BRNN proposto por Sun et al. (2020) na detecção de falhas em processos químicos que não possuem condições de operação normal estacionárias, recomenda-se estudos sobre o desenvolvimento de adaptações do método que viabilizem que os modelos sejam treinados em uma abordagem *online*, permitindo assim que os modelos sejam continuamente retreinados de acordo com a identificação de uma configuração de operação normal e consequentemente possam ser aplicados para processos químicos que possuem múltiplas configurações de operação normal.

- **Recomendações gerais**

Para ambos os estudos de caso, o método de aproximação de inferência bayesiana *MC-Dropout* foi o utilizado para implementação dos modelos de redes neurais bayesianas. Esse método foi escolhido pelo fato de ser simples de implementar, não exigir modificações nas estruturas das redes neurais e garantir uma boa capacidade de generalização aos modelos. Entretanto, de acordo com estudos recentes, o método apresenta alguns problemas tais como o fato de não capturar totalmente a incerteza associada às predições do modelo,

além de ser menos flexível em comparação com outros métodos bayesianos para aprendizado online ou ativo de redes neurais (JOSPIN et al., 2022). Sendo assim, recomenda-se estudos de aplicação de métodos de aproximação de inferência bayesianas que vêm surgindo como alternativa ao método *MC-Dropout*, tais como os métodos de Maddox et al. (2019) e Lakshminarayanan, Pritzel e Blundell (2017).

REFERÊNCIAS

ABADI, M. et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

AHMAD, S. et al. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, Elsevier, v. 262, p. 134–147, 2017.

AKIBA, T. et al. Optuna: A next-generation hyperparameter optimization framework. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. [S.l.: s.n.], 2019. p. 2623–2631.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

BAND, N. et al. Benchmarking bayesian deep learning on diabetic retinopathy detection tasks. In: *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*. [S.l.: s.n.], 2021.

BARBER, D.; BISHOP, C. M. Ensemble learning in bayesian neural networks. *Neural Networks and Machine Learning*, Springer, v. 168, p. 215–238, 1998.

BLUNDELL, C. et al. Weight uncertainty in neural network. In: *International Conference on Machine Learning*. [S.l.: s.n.], 2015. p. 1613–1622.

BRODERSEN, K. H. et al. The balanced accuracy and its posterior distribution. In: IEEE. *2010 20th International Conference on Pattern Recognition*. [S.l.], 2010. p. 3121–3124.

BRUSAFERRI, A. et al. Bayesian deep learning based method for probabilistic forecast of day-ahead electricity prices. *Applied Energy*, Elsevier, v. 250, p. 1158–1175, 2019.

CASTELL, N. et al. Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates? *Environment international*, Elsevier, v. 99, p. 293–302, 2017.

CAVALCANTE, R. C. et al. Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications*, Elsevier, v. 55, p. 194–211, 2016.

CETESB. *CETESB inaugura nova estação automática de monitoramento do ar em Limeira*. 2019. Disponível em: ⟨https://cetesb.sp.gov.br/blog/2019/07/16/cetesb-inaugura-nova-estacao-automatizada-de-monitoramento-do-ar-em-limeira/⟩.

CETESB. *Mapa da Qualidade do Ar*. 2022. Disponível em: ⟨https://servicos.cetesb.sp.gov.br/qa/⟩.

CHEN, C. et al. Deepdriving: Learning affordance for direct perception in autonomous driving. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2015. p. 2722–2730.

CHEN, T.; FOX, E.; GUESTRIN, C. Stochastic gradient hamiltonian monte carlo. In: *International Conference on Machine Learning*. [S.l.: s.n.], 2014. p. 1683–1691.

CHENG, F.; HE, Q. P.; ZHAO, J. A novel process monitoring approach based on variational recurrent autoencoder. *Computers & Chemical Engineering*, Elsevier, v. 129, p. 106515, 2019.

CHIANG, L. H.; KOTANCHEK, M. E.; KORDON, A. K. Fault diagnosis based on fisher discriminant analysis and support vector machines. *Computers & Chemical Engineering*, Elsevier, v. 28, n. 8, p. 1389–1401, 2004.

CHIANG, L. H.; RUSSELL, E. L.; BRAATZ, R. D. *Fault Detection and Diagnosis in Industrial Systems*. [S.l.]: Springer Science & Business Media, 2000.

CHIANG, L. H.; RUSSELL, E. L.; BRAATZ, R. D. Fault diagnosis in chemical processes using fisher discriminant analysis, discriminant partial least squares, and principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, Elsevier, v. 50, n. 2, p. 243–252, 2000.

CHO, K. et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

CHUNG, J. et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

CIODARO, T. et al. Online particle detection with neural networks based on topological calorimetry information. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2012. v. 368, n. 1, p. 012030.

CONCAS, F. et al. Low-cost outdoor air quality monitoring and sensor calibration: A survey and critical analysis. *ACM Transactions on Sensor Networks (TOSN)*, ACM New York, NY, USA, v. 17, n. 2, p. 1–44, 2021.

CORDERO, J. M.; BORGE, R.; NARROS, A. Using statistical methods to carry out in field calibrations of low cost air quality sensors. *Sensors and Actuators B: Chemical*, Elsevier, v. 267, p. 245–254, 2018.

DAHL, G. E. et al. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, IEEE, v. 20, n. 1, p. 30–42, 2011.

DEODATO, G.; BALL, C.; ZHANG, X. Bayesian neural networks for cellular image classification and uncertainty analysis. *bioRxiv*, Cold Spring Harbor Laboratory, p. 824862, 2020.

DONG, Y.; QIN, S. J. A novel dynamic pca algorithm for dynamic data modeling and process monitoring. *Journal of Process Control*, Elsevier, v. 67, p. 1–11, 2018.

DOWNS, J. J.; VOGEL, E. F. A plant-wide industrial process control problem. *Computers & Chemical Engineering*, Elsevier, v. 17, n. 3, p. 245–255, 1993.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, v. 12, n. 7, p. 2121–2159, 2011.

ESPOSITO, E. et al. Dynamic neural network architectures for on field stochastic calibration of indicative low cost air quality sensing systems. *Sensors and Actuators B: Chemical*, Elsevier, v. 231, p. 701–713, 2016.

FORTUNATO, M.; BLUNDELL, C.; VINYALS, O. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017.

GAL, Y. *Uncertainty in Deep Learning*. Tese (Doutorado) — University of Cambridge, 2016.

GAL, Y.; GHAHRAMANI, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: *International Conference on Machine Learning*. [S.l.: s.n.], 2016. p. 1050–1059.

GAL, Y.; GHAHRAMANI, Z. A theoretically grounded application of dropout in recurrent neural networks. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2016. p. 1019–1027.

GAL, Y.; ISLAM, R.; GHAHRAMANI, Z. Deep bayesian active learning with image data. p. 1183–1192, 2017.

GE, Z.; SONG, Z.; GAO, F. Review of recent research on data-based process monitoring. *Industrial & Engineering Chemistry Research*, ACS Publications, v. 52, n. 10, p. 3543–3562, 2013.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016.

GRAVES, A. Practical variational inference for neural networks. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2011. p. 2348–2356.

GRAVES, A. *Supervised Sequence Labelling with Recurrent Neural Networks*. [S.l.]: Springer, 2012.

HASENFRATZ, D.; SAUKH, O.; THIELE, L. On-the-fly calibration of low-cost gas sensors. In: SPRINGER. *European Conference on Wireless Sensor Networks*. [S.l.], 2012. p. 228–244.

HASENFRATZ, D. et al. Deriving high-resolution urban air pollution maps using mobile sensor nodes. *Pervasive and Mobile Computing*, Elsevier, v. 16, p. 268–285, 2015.

HAYKIN, S. *Neural Networks and Learning Machines*. [S.l.]: Pearson Education, 2010.

HERNÁNDEZ, E. et al. Rainfall prediction: A deep learning approach. In: SPRINGER. *International Conference on Hybrid Artificial Intelligence Systems*. [S.l.], 2016. p. 151–162.

HERNÁNDEZ-LOBATO, J. M.; ADAMS, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In: *International Conference on Machine Learning*. [S.l.: s.n.], 2015. p. 1861–1869.

HINTON, G. et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, IEEE, v. 29, n. 6, p. 82–97, 2012.

HINTON, G. E.; CAMP, D. V. Keeping the neural networks simple by minimizing the description length of the weights. In: *Proceedings of the sixth annual conference on Computational Learning Theory*. [S.l.: s.n.], 1993. p. 5–13.

HINTON, G. E.; OSINDERO, S.; TEH, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, MIT Press, v. 18, n. 7, p. 1527–1554, 2006.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HUNDMAN, K. et al. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. [S.l.: s.n.], 2018. p. 387–395.

JOSPIN, L. V. et al. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, IEEE, v. 17, n. 2, p. 29–48, 2022.

KANO, M. et al. Monitoring independent components for fault detection. *AIChE Journal*, Wiley Online Library, v. 49, n. 4, p. 969–976, 2003.

KENDALL, A.; GAL, Y. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, v. 30, 2017.

KER, J. et al. Deep learning applications in medical image analysis. *IEEE Access*, IEEE, v. 6, p. 9375–9389, 2017.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

KINGMA, D. P.; WELLING, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, v. 25, p. 1097–1105, 2012.

KU, W.; STORER, R. H.; GEORGAKIS, C. Disturbance detection and isolation by dynamic principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, Elsevier, v. 30, n. 1, p. 179–196, 1995.

LAKSHMINARAYANAN, B.; PRITZEL, A.; BLUNDELL, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, v. 30, 2017.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

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

LEE, J. H.; SHIN, J.; REALFF, M. J. Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers & Chemical Engineering*, Elsevier, v. 114, p. 111–121, 2018.

LEE, J.-M. et al. Nonlinear process monitoring using kernel principal component analysis. *Chemical Engineering Science*, Elsevier, v. 59, n. 1, p. 223–234, 2004.

LENAIL, A. Nn-svg: Publication-ready neural network architecture schematics. *J. Open Source Softw.*, v. 4, n. 33, p. 747, 2019.

LEUNG, M. K. et al. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, Oxford University Press, v. 30, n. 12, p. i121–i129, 2014.

LEWIS, A.; EDWARDS, P. Validate personal air-pollution sensors. *Nature News*, v. 535, n. 7610, p. 29, 2016.

LIBELIUM. *Calibrated Air Quality Sensors for Smart Cities*. 2015. Disponível em: ⟨https://www.libelium.com/libeliumworld/calibrated-air-quality-gas-dust-particle-matter-pm10-smart-cities/⟩.

LIBELIUM. *Air Quality Station Technical Guide*. 2022. Disponível em: ⟨https://development.libelium.com/air-quality-station-technical-guide/hardware⟩.

LIBELIUM. *Air Quality Stations*. 2022. Disponível em: ⟨https://www.libelium.com/iot-products/air-quality-station/⟩.

LV, F. et al. Fault diagnosis based on deep learning. In: IEEE. *2016 American Control Conference (ACC)*. [S.l.], 2016. p. 6851–6856.

MAAG, B.; ZHOU, Z.; THIELE, L. A survey on sensor calibration in air pollution monitoring deployments. *IEEE Internet of Things Journal*, IEEE, v. 5, n. 6, p. 4857–4870, 2018.

MACGREGOR, J. F. et al. Process monitoring and diagnosis by multiblock pls methods. *AIChE Journal*, Wiley Online Library, v. 40, n. 5, p. 826–838, 1994.

MACKAY, D. J. A practical bayesian framework for backpropagation networks. *Neural Computation*, MIT Press, v. 4, n. 3, p. 448–472, 1992.

MACKAY, D. J. *Information Theory, Inference and Learning algorithms*. [S.l.]: Cambridge University Press, 2003.

MADDOX, W. *Bayesian Neural Networks: A Tutorial*. 2020. CILVR Lab Tutorial. Disponível em: ⟨https://wjmaddox.github.io⟩.

MADDOX, W. J. et al. A simple baseline for bayesian uncertainty in deep learning. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2019. p. 13153–13164.

MALHOTRA, P. et al. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

MALHOTRA, P. et al. Long short term memory networks for anomaly detection in time series. In: *ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. [S.l.: s.n.], 2015. v. 89, p. 89–94.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

MCFARLANE, R. C. et al. Dynamic simulator for a model iv fluid catalytic cracking unit. *Computers & chemical engineering*, Elsevier, v. 17, n. 3, p. 275–300, 1993.

MINSKY, M.; PAPERT, S. A. *Perceptrons*. [S.l.]: MIT Press, 1969.

MORAWSKA, L. et al. Applications of low-cost sensing technologies for air quality monitoring and exposure assessment: How far have they gone? *Environment International*, Elsevier, v. 116, p. 286–299, 2018.

MURPHY, K. P. *Machine Learning: A Probabilistic Perspective*. [S.l.]: MIT Press, 2012.

NEAL, R. M. *Bayesian Learning for Neural Networks*. Tese (Doutorado) — University of Toronto, 1995.

NG, A. Y. Feature selection, l1 vs. l2 regularization, and rotational invariance. In: *Proceedings of the twenty-first international conference on Machine learning*. [S.l.: s.n.], 2004. p. 78.

OLAH, C. *Understanding LSTM Networks*. 2015. Disponível em: ⟨https://colah.github.io/posts/2015-08-Understanding-LSTMs/⟩.

PASZKE, A. et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, v. 32, p. 8026–8037, 2019.

POUYANFAR, S. et al. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 51, n. 5, p. 1–36, 2018.

RITTER, H.; BOTEV, A.; BARBER, D. A scalable laplace approximation for neural networks. In: INTERNATIONAL CONFERENCE ON REPRESENTATION LEARNING. *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*. [S.l.], 2018. v. 6.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning Internal Representations by Error Propagation*. [S.l.], 1985.

SADEGHBEIGI, R. *Fluid Catalytic Cracking Handbook: An Expert Guide to the Practical Operation, Design, and Optimization of FCC Units*. [S.l.]: Butterworth-Heinemann, 2020.

SAUKH, O.; HASENFRATZ, D.; THIELE, L. Reducing multi-hop calibration errors in large-scale mobile sensor networks. In: *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*. [S.l.: s.n.], 2015. p. 274–285.

SHU, Y. et al. Abnormal situation management: Challenges and opportunities in the big data era. *Computers & Chemical Engineering*, Elsevier, v. 91, p. 104–113, 2016.

SILVER, D. et al. Mastering the game of go with deep neural networks and tree search. *Nature*, Nature Publishing Group, v. 529, n. 7587, p. 484–489, 2016.

SPINELLE, L. et al. Field calibration of a cluster of low-cost available sensors for air quality monitoring. part a: Ozone and nitrogen dioxide. *Sensors and Actuators B: Chemical*, Elsevier, v. 215, p. 249–257, 2015.

SPINELLE, L. et al. Field calibration of a cluster of low-cost commercially available sensors for air quality monitoring. part b: No, co and co2. *Sensors and Actuators B: Chemical*, Elsevier, v. 238, p. 706–715, 2017.

SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

SUN, W. et al. Fault detection and identification using bayesian recurrent neural networks. *Computers & Chemical Engineering*, Elsevier, v. 141, p. 106991, 2020.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2014. p. 3104–3112.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. [S.l.]: MIT Press, 2018.

SZEGEDY, C. et al. Going deeper with convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2015. p. 1–9.

TIELEMAN, T.; HINTON, G. *Lecture 6.5 - RmsProp: Divide the gradient by a running average of its recent magnitude*. 2012. COURSERA: Neural Networks for Machine Learning.

TOURETZKY, D. S.; HINTON, G. E. Symbols among the neurons: Details of a connectionist inference architecture. In: *IJCAI*. [S.l.: s.n.], 1985. v. 85, p. 238–243.

TROYANSKAYA, O. et al. Missing value estimation methods for dna microarrays. *Bioinformatics*, Oxford University Press, v. 17, n. 6, p. 520–525, 2001.

UNION, C. of E. Directive 2008/50/ec of the european parliament and of the council of 21 may 2008 on ambient air quality and cleaner air for europe. *Official Journal of the European Union*, 2008.

VENKATASUBRAMANIAN, V.; CHAN, K. A neural network methodology for process fault diagnosis. *AIChE Journal*, Wiley Online Library, v. 35, n. 12, p. 1993–2002, 1989.

VITO, S. D. et al. Calibrating chemical multisensory devices for real world applications: An in-depth comparison of quantitative machine learning approaches. *Sensors and Actuators B: Chemical*, Elsevier, v. 255, p. 1191–1210, 2018.

VITO, S. D. et al. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical*, Elsevier, v. 129, n. 2, p. 750–757, 2008.

VITO, S. D. et al. Co, no2 and nox urban pollution monitoring with on-field calibrated electronic nose by automatic bayesian regularization. *Sensors and Actuators B: Chemical*, Elsevier, v. 143, n. 1, p. 182–191, 2009.

WANG, L. et al. Wind turbine gearbox failure identification with deep neural networks. *IEEE Transactions on Industrial Informatics*, IEEE, v. 13, n. 3, p. 1360–1368, 2017.

WESTHUIZEN, J. van der; LASENBY, J. Bayesian lstms in medicine. *arXiv preprint arXiv:1706.01242*, 2017.

WHO. Ambient air pollution: A global assessment of exposure and burden of disease. World Health Organization, 2016.

WHO. *Ambient (outdoor) air pollution*. 2021. Disponível em: ⟨https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health⟩.

WIDROW, B.; HOFF, M. E. *Adaptive Switching Circuits*. [S.l.], 1960.

WILLIAMS, R. J.; ZIPSER, D. Gradient-based learning algorithms for recurrent networks and their computational complexity. *Backpropagation: Theory, Architectures, and Applications*, p. 433–486, 1995.

WISE, B. M. et al. A theoretical basis for the use of principal component models for monitoring multivariate processes. *Process Control and Quality*, v. 1, n. 1, p. 41–51, 1990.

WU, H.; ZHAO, J. Deep convolutional neural network model based chemical process fault diagnosis. *Computers & Chemical Engineering*, Elsevier, v. 115, p. 185–197, 2018.

YIN, S. et al. A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark tennessee eastman process. *Journal of Process Control*, Elsevier, v. 22, n. 9, p. 1567–1581, 2012.

ZHANG, Z.; ZHAO, J. A deep belief network based fault diagnosis model for complex chemical processes. *Computers & Chemical Engineering*, Elsevier, v. 107, p. 395–407, 2017.

ZHAO, H.; SUN, S.; JIN, B. Sequential fault diagnosis based on lstm neural network. *IEEE Access*, IEEE, v. 6, p. 12929–12939, 2018.

ZIMMERMAN, N. et al. A machine learning calibration model using random forests to improve sensor performance for lower-cost air quality monitoring. *Atmospheric Measurement Techniques*, Copernicus GmbH, v. 11, n. 1, p. 291–313, 2018.