UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Análise de Desempenho de Método Baseado em Rede LSTM para Classificação de Falhas em um Processo de Controle de Nível

Emerson Vilar de Oliveira

Orientador: Prof. Dr. Luiz Affonso Henderson Guedes de Oliveira

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Mestre em Ciências.

Número de ordem PPgEEC: M609
Natal, RN, agosto 2020

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Oliveira, Emerson Vilar de.
Análise de desempenho de método baseado em rede LSTM para classificação de falhas em um processo de controle de nível / Emerson Vilar de Oliveira. - 2020.
63 f.: il.

Dissertacão (mestrado) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Programa de Pós Graduação em Engenharia Elétrica e de Computação, Natal, RN, 2020.
Orientador: Prof. Dr. Luiz Affonso Henderson Guedes de Oliveira.

1. Classificação de falhas - Dissertação. 2. Redes neurais - Dissertação. 3. Redes neurais recorrentes - Dissertação. 4. Long short-term memory - Dissertação. 5. Planta piloto - Dissertação. I. Oliveira, Luiz Affonso Henderson Guedes de. II. Título.

RN/UF/BCZM CDU 681.511.4

Elaborado por Ana Cristina Cavalcanti Tinôco - CRB-15/262

Resumo

Devido ao aumento das exigências no monitoramento de operação em plantas industriais, metodologias para detecção e diagnósticos de falhas na operação desses processos vêm ganhando cada vez mais protagonismo, pois podem contribuir em soluções mais assertivas e até preditivas nos componentes que geraram tais perturbações ao bom funcionamento do sistema. Com o crescimento das abordagens orientadas a dados, as Redes Neurais Artificiais se tornaram consideráveis aliadas na solução destes problemas, sendo que as Redes Neurais Recorrentes, em especial, ganharam força devido a sua afinidade em lidar com séries que possuem ligações temporais entre suas amostras, que é o caso de monitoramento de variáveis de processos industriais. Devido a esta relevância, nesta dissertação é analisado o desempenho de um modelo baseado em rede neural recorrente do tipo *Long Short Term-Memory* (LSTM) para a detecção e classificação de falhas em um processo de controle de nível em escala piloto. Para a análise de desempenho foi empregada uma metodologia baseada em testes estatísticos de Monte Carlo, onde se analisou a influência dos hiperparâmetros das redes LSTM, como quantidade de camadas e tamanho da entrada e regressores. A métrica escolhida para quantificar o desempenho da classificação de falhas foi a acurácia. Os dados obtidos a partir da operação da planta piloto continham 23 situações de distúrbios nesse processo, que foram provenientes de perturbações aplicados em componentes como atuador, sensor, válvulas e o próprio tanque de água. A metodologia adotada se mostrou bastante eficiente para analisar tanto o desempenho quanto a robustez dessas redes neurais para a atividade de classificação de falhas, além de indicar as melhores configurações de arquitetura da rede.

Palavras-chave: Classificação de falhas, redes neurais, redes neurais recorrentes, *long short-term memory*, planta piloto.

Abstract

Due to the increasing demands in the operation monitoring of industrial plants, methodologies for fault detect and diagnose in the operation of these processes are gaining more and more importance, because they can contribute to more assertive and even predictive repairs in the components that generated such disturbances to the proper functioning of the system. With the growth of data-oriented approaches, Artificial Neural Networks have become considerable allies in solving these problems, and Recurrent Neural Networks, in particular, has gained strength due to their affinity in dealing with series that have temporal links between their samples, which is the case of industrial process variables monitoring. Due to this relevance, this dissertation analyzes the performance of Long Short-Term Memory (LSTM) recurrent neural network for the detection and classification of faults in a pilot-scaled level control process. For the performance evaluation, a methodology based on Monte Carlo statistical tests was used, in which the influence of the LSTM network hyperparameters, such as the number of layers and size of the input and regressors, was analyzed. The accuracy was the metric chosen to quantify the fault classification performance. The data set obtained from the operation of the pilot plant contained 23 situations of disturbances in this process, which resulted from disturbances applied to components such as sensors, valves, and the water tank itself. The adopted methodology proved to be quite efficient to examine both the performance and the robustness of these neural networks for the fault classification activity, in addition to indicating the best network architecture configurations.

Keywords: Fault classification, neural networks, recurrent neural network, long short-term memory, pilot plant.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Devido ao crescente avanço industrial das últimas décadas, a integração e complexidade dos processos industriais se tornaram importantes fatores para confiabilidade e segurança desses ambientes. De modo geral, um processo pode ser definido como qualquer operação artificial ou voluntária que evolui de forma progressiva, sendo esta uma série de ações controladas ou sistemáticas, dirigidas a obter determinado resultado ou objetivo. Em um sistema industrial, uma variável de processo pode ser definida como estados que de alguma maneira afetam o desempenho total do processo, sendo estes estados internos ou externos ao meio (Ogata & Severo 1998). Cada variável de processo possui uma região de valores em que ela pode transitar sem oferecer riscos para o funcionamento normal da planta. Este espaço de valores é denominado de região normal da variável de processo, de modo que, ao divergir dessa região, a variável de processo entra no que se conhece como região anormal do processo. Nos processos industriais modernos é necessário o monitoramento e controle destas variáveis.

Ações de controle de baixo nível, como abertura e fechamento de válvulas, chamadas de controle regulatório, que antes eram realizadas por operadores humanos, passaram a ser realizadas de forma automatizada com o auxílio de computadores, pelos sistemas de controle industrial. Com isso, houveram diversos benefícios em vários segmentos industrias, como as indústrias químicas, petroquímicas, aço, energia, dentre outros. Mesmos com esses avanços, a tarefa de responder a eventos anormais do processo em sua grande maioria era feita por humanos. Essas tarefas envolvem a detecção de um evento anormal, diagnóstico de suas origens e causas, seguida de uma tomada de decisão para o controle e supervisão apropriados, como também, ações para retornar o processo ao seu estado normal de funcionamento. Essa atividade passou a ser chamada de Gerenciamento de Eventos Anormais (AEM, *Abnormal Event Management*). Esse tipo de tarefa se tornou dificultosa para os operadores humanos, devido à grande quantidade de variáveis de processos a serem monitoradas dentro de um sistema de controle. Além disso, as medições dessas variáveis de processo podem ser insuficientes, incompletas e não confiáveis devido a uma variedade de causas, como polarizações ou um mau funcionamento de um sensor (Venkatasubramanian, Rengaswamy, Yin & Kavuri 2003).

É importante ressaltar que sob pressão, devido à complexidade de se operar tais processos industriais, os operadores tenderam a tomar decisões e ações errôneas, onde as estatísticas industriais mostraram que cerca de 70% dos acidentes industriais são causados

por erros humanos. Atualmente, a comunidade industrial mostrou que o controle regulatório dos processos pode ser executado de forma automática pelos computadores, removendo essa tarefa das mãos humanas, consequentemente diminuindo a chance de erros no controle acontecerem devido ao sobrecarregamento da mão-de-obra dos operadores. Porém, apesar dos avanços nos sistemas de controle de forma computacional, os acontecimentos gerados por falhas humanas no AEM tiveram impacto ambiental, econômico e de segurança significativos. Da mesma forma, o desafio de promover o AEM utilizando sistemas inteligentes está proporcionando diversas pesquisas nesta área, com o objetivo de auxiliar os operadores humanos a tomar melhores ações e decisões (Venkatasubramanian, Rengaswamy, Yin & Kavuri 2003).

As metodologias para Detecção e Isolamento de Falhas (FDI, *Fault Detection and Isolation*, em inglês) servem de auxílio para os operadores dos sistemas de controle industrial na tarefa de realizar o AEM. A atividade de FDI é considerada crítica dentro do sistema de monitoramento dos processos industriais, devido ao fato que uma rápida detecção e diagnóstico de uma anormalidade em ocorrência, enquanto o sistema está operando em uma região controlável, pode prevenir e reduzir perdas de produtividade, assim como os riscos de acidentes (Venkatasubramanian, Rengaswamy, Yin & Kavuri 2003). Assim, dada a sua importância, é necessário avaliar e validar soluções para este problema antes de serem colocadas em sistemas reais.

Geralmente na literatura acadêmica, são utilizadas de estratégias que simulam e implementam de forma computacional protótipos de sistemas e processos industriais em escala reduzida para validar as propostas de sistemas de FDI. Um bom exemplo é o *benchmark Tennessee Eastman Procces* (TEP), que é um simulador de processos industrias amplamente utilizado na literatura (Yin et al. 2012, Gajjar & Palazoglu 2016, D'Angelo et al. 2016, Gao & Hou 2016, Basha et al. 2020, Verron et al. 2006, Chiang et al. 2004). O TEP foi desenvolvido pela *Eastman Chemical Company* tendo o objetivo de promover um simulador baseado em um processo químico real para avaliar metologias de controle e monitoramento de processos (Downs & Vogel 1993). Como recurso utilizado na forma de processos não simulados, o *benchmark* DAMADICS (*Development and Application of Methods for Actuator Diagnosis in Industrial Control Systems*) foi proposto, tendo uma boa adoção nos estudos para a FDI (Chopra & Vajpai 2011, Kourd et al. 2013, Lipnickas et al. 2004, Marciniak et al. 2003, Hadroug et al. 2015). O DAMADICS também possibilita a geração de dados de forma simulada, contudo, os dados reais disponíveis neste *benchmark* são extraídos de um processo muito simples de evaporação de água em uma fábrica de açúcar polonesa (Bartyś et al. 2006).

Há na literatura diversos métodos que utilizam técnicas matemáticas e estatísticas para desenvolver sistemas de FDI. Nesta dissertação será avaliado um método baseado em histórico de processos. Mais precisamente, essa abordagem utiliza um tipo de rede neural artificial (RNA) recorrente para efetuar a classificação das falhas. De certa forma, a maioria dos métodos para FDI se utiliza de algum tipo de informação que foi gerada pelo sistema de monitoramento de processos. Porém, os métodos baseados em histórico de processos usam os dados provindos diretamente do sistema de controle, como as respostas dos sensores, atuadores, válvulas e qualquer outro componente que reaja em situações anômalas. Com o avanço do emprego de sistemas de armazenamento de dados de variáveis

em larga escala na área de processos industriais, os sistemas de FDI baseados no histórico de dados de variáveis de processo se tornam bastantes vantajosos, pois conseguem obter modelos simples, robustos e possuem menor necessidade de conhecimento especializado sobre o funcionamento dos processos quando comparados com as abordagens baseadas em modelos fenomenológicos.

1.1 Motivação

Tendo uma grande quantidade de informações a ser processada para o monitoramento de operação de um processo industrial, os sistemas de FDI são responsáveis por reduzir a carga de informações do operador. Deste modo, os sistemas de FDI têm grande relevância, pois diminuem a sobrecarga dos operadores e estes podem ter mais tempo hábil para realizar decisões finais. Como já mencionado, é necessário validar e avaliar esses sistemas de FDI com um algum tipo de conjunto de dados. Sendo que esses conjuntos de dados podem advir de simulação de processos ou de operação de processos reais, sendo que a maior parte dos estudos disponíveis na literatura que analisam o desempenho dos sistemas de FDI se baseia em dados simulados de processo. Embora sejam bastantes válidos tais estudos baseados em dados simulados, é muito importante também avaliar o desempenho dos sistemas FDI com dados de operação de processos reais. Como falhas são situações anormais e atípicas na operação de processos industrias, é bastante raro se ter dados confiáveis de tais instantes. Isto explica em boa medida o uso de dados simulados. Outro aspecto em favor do uso de dados simulados é que nessa abordagem é possível se ter controle sobre a geração dos instantes e intensidades das falhas. Uma alternativa entre essas duas abordagens é o uso de dados de processos de escala reduzida, que pode conciliar as vantagens de se utilizar simulação e de dados reais de processos, pois permite se ter mais controle sobre a geração da falhas e utiliza dados de sinais reais, que possuem características tipicamente encontradas na prática, como ruídos e erros de medição.

Outra carência que se observa na literatura é o emprego de abordagens sistemáticas para se avaliar o desempenho de sistemas de FDI. Sem o emprego de abordagens sistemáticas, não é possível reproduzir adequadamente os resultados e nem avaliar as influências que cada hiperparâmetro dos modelos têm sobre o desempenho dos mesmos.

1.2 Objetivos da Dissertação

O objetivo deste estudo é utilizar uma metodologia presente na literatura para resolução do problema de FDI em dados de um processo obtido a partir de uma planta piloto que realiza controle de nível. Esta planta produz um conjunto de dados mais próximos de um sistema industrial real, por possuir uma variedade de componentes normalmente encontrados em processos industrias reais, como atuadores, sensores, tanques e válvulas. Neste contexto, é proposto um estudo de caso onde a metodologia de FDI será aplicada para classificação das falhas de modo a identificá-las dentre os conjuntos que distinguem o tipo de distúrbio que ocorreu em determinado componente que a gerou. Além da aplicação da técnica de FDI neste conjunto de dados, será proposta uma metodologia para

avaliação do comportamento do modelo a mudanças nos seus hiperparâmetros. Esta metodologia para avaliação examina de forma exaustiva diversos valores para características internas do modelo, com o objetivo de encontrar uma combinação que seja capaz de promover uma boa classificação das falhas encontradas, aliada a um baixo custo computacional. Esse baixo custo computacional visa a aplicabilidade posterior da metodologia em um sistema que opere em tempo real.

Além destes objetivos, o trabalho observa a capacidade das redes neurais artificiais recorrentes do tipo *Long Short-Term Memory* em lidar com dados de processo que possuem ligação temporal entre suas amostras.

1.3 Contribuições

De modo geral, este trabalho contribui com a validação de uma metodologia de FDI para classificação de falhas em um conjunto de dados gerados por um processo não simulado, além de uma metodologia de avaliação da variação dos seus parâmetros, de forma a obter o melhor *trade-off* entre desempenho e custo computacional. De uma forma mais detalhada, podem ser destacadas:

- Desenvolvimento de um *pipeline* computacional para avaliar a aplicabilidade do método em classificar falhas em dados de um processo não simulado.
- Proposta para seleção de falhas a serem utilizadas no conjunto de treinamento e de validação do modelo. Esta seleção é baseada na média entre a distância euclidiana das variáveis de processo presentes em cada uma das falhas. É determinado um parâmetro de dissimilaridade entre duas falhas diferentes geradas pelo mau funcionamento do mesmo componente.
- Aplicação de um teste que observa a capacidade do modelo em classificar corretamente as falhas de validação, dado que foi treinado com um conjunto de falhas diferentes, onde ambas pertencem ao mesmo grupo relacionado ao tipo da causa.
- Implementação e avaliação de dois novos parâmetros ainda não explorados pela proposta inicial da metologia para FDI. Estes parâmetros são a sobreposição (*overlap*) de amostras presentes na entrada do modelo e a adição de uma segunda camada na rede neural artificial utilizada.
- Avaliação individual de quatro parâmetros presentes no método, sendo eles o tamanho da rede, a quantidade de camadas, o tamanho da entrada do modelo e a sobreposição de amostras em suas entradas. Estes parâmetros são avaliados de forma exaustiva, com o objetivo de observar distribuidamente a performance do modelo a essas alterações.
- Metologia para avaliação visual dos resultados obtidos a partir dos testes exaustivos na variação dos parâmetros propostos.

1.4 Estrutura da Dissertação

Esta dissertação de Mestrado esta dividida em mais 5 capítulos. O Capítulo 2 introduz de forma breve os conceitos teóricos a respeito de Redes Neurais Artificiais, que fazem

parte da metodologia para FDI avaliada. O Capítulo 3 apresenta os tipos de metologia existentes na literatura para solução do problema de FDI, como também onde as RNA se identificam neste contexto. No Capítulo 4, a abordagem para a classificação de falhas e a metologia para avaliação dos seus parâmetros são explicadas. O Capítulo 5 mostra os detalhes da aplicação do Estudo de Caso feito para os dados de uma planta piloto para controle de nível, além das variações feitas para os parâmetros do modelo e resultados obtidos. Por fim, o Capítulo 6 traz as conclusões e considerações finais sobre o estudo.

Capítulo 2

Redes Neurais Artificiais

Dentro do contexto de diagnóstico de falhas em processos industriais utilizando redes neurais, existem alguns pontos que requerem uma iniciação teórica. O objetivo é apresentar alguns conceitos para uma melhor compreensão de recursos utilizados posteriormente. Neste capitulo serão mostrados os conceitos e princípios de funcionamento das redes neurais artificiais, além das redes neurais recorrentes e seus problemas típicos. Através dos estudos sobre esses problemas, as redes LSTM foram propostas como solução. Aqui serão mostrados os fundamentos matemáticos no qual as redes neurais se apoiam para serem capazes de executar tarefas consideravelmente complexas.

2.1 Neurônios Artificiais

O funcionamento das redes neurais artificiais foi espelhado no comportamento do cérebro humano. O cérebro humano contém bilhões de estruturas simples de processamento, denominadas de neurônios, onde cada um deles está conectado a centenas de outros através de sinapses. Os neurônios são constituídos por, basicamente, corpo, chamado de soma, axônio e dendritos, que são responsáveis pela condução e recepção dos estímulos, respectivamente. Os estímulos são recebidos de outros neurônios pelos dendritos. Uma das teorias é de que após uma reação química, o neurônio envia uma ação através do axônio em direção às sinapses conectadas aos dendritos de outros neurônios. Individualmente esses neurônios são consideravelmente simples, porém, em uma rede composta de bilhões dessas estruturas são capazes de processar informações complexas e de forma extremamente rápida (Coppin 2004, Haykin 2007).

As redes neurais artificiais são análogas a essas estruturas, funcionando de forma parecida, tendo como objetivo modelar uma função que responda a determinado problema. Essas redes têm bem menos neurônios e com bem menos conexões que um cérebro humano, o que as torna bem menos complexas. Para simular as reações ocorridas no cérebro humano, os neurônios artificiais recebem diversas entradas, que após serem processadas, produzem saídas que alimentam funções conhecidas como funções de ativação. O resultado dessa função é chamado de nível de ativação, que é a saída do neurônio. Essas funções são em alguns casos delimitadoras do quanto a saída do neurônio pode variar. A função de ativação mais comum é a de limite linear, onde após a soma do produto dos pesos pelas entradas ser efetuada, o resultado é comparado com um limiar definido. Caso

essa soma exceda o limiar, o neurônio transmite sua informação, caso o contrário uma informação nula é transmitida. Já os pesos, são análogos as sinapses dos neurônios biológicos, eles são responsáveis por ponderar os valores das entradas para em seguidas serem somados com um *bias*, configurando uma operação linear (Coppin 2004). A Figura 2.1 exemplifica a estrutura de um neurônio.

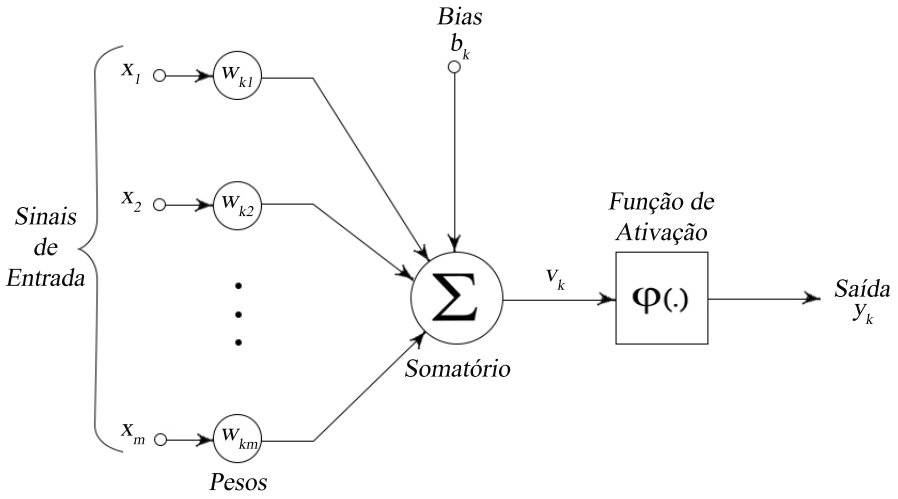


Figura 2.1: Figura representativa de um neurônio artificial.

Na Figura 2.1, $w_{k1}, w_{k2}, \ldots, w_{km}$ são os pesos de um neurônio, que recebe m entradas. Além dos pesos, um *bias* denominado b_k é adicionado na entrada da função de ativação φ. A equação (2.1) descreve o somatório das entradas de um neurônio artificial u_k.

$$u_k = \sum_{i=1}^{m} w_{ki} x_i \tag{2.1}$$

Sendo que u é a soma das n entradas x do neurônio k, ponderadas pelos pesos w. Após esse somatório, o *bias* é adicionado. Assim temos que:

$$v_k = u_k + b_k \tag{2.2}$$

O resultado v_k é, então, submetido à função de ativação φ. A função de ativação de limite limiar também é normalmente referida como a função de *Heaviside*, ou função degrau. A equação (2.3) descreve esse tipo de operação.

$$\varphi(v_k) = \begin{cases} 1, & \textit{para } v_k \geq \textit{limiar} \\ 0, & \textit{para } v_k < \textit{limiar} \end{cases} \tag{2.3}$$

No caso, y, descrito na Figura 2.1, é a saída da função de ativação para o neurônio k (Haykin 2007).

Existem diversas funções de ativação diferentes ao do limite linear. As equações (2.4) e (2.5) mostram as funções de ativação Sigmoid e Tangente Hiperbólica, respectivamente

(Haykin 2007).

$$\varphi(v) = \frac{1}{1+exp(-\alpha v)} \tag{2.4}$$

$$\varphi(v) = \frac{exp(v)-exp(-v)}{exp(v)+exp(-v)} \tag{2.5}$$

2.2 Perceptrons e *Multilayer* Perceptrons

O *perceptron* é a forma mais simples de uma rede neural artificial, a qual é composta de apenas um neurônio, sendo capaz de modelar uma função que separa suas entradas em duas classes (Coppin 2004). O *perceptron* basicamente funciona ajustando os seus pesos e *bias* para modelar uma função que descreve uma superfície capaz de delimitar duas classes linearmente separáveis. Um único *perceptron* é capaz de separar duas classes, sendo necessário uma camada com mais *perceptrons* para diferenciar três ou mais classes. Porém, essas classes devem continuar sendo linearmente separáveis para o *perceptron* funcionar adequadamente (Haykin 2007).

Embora os *perceptrons* sejam bons objetos para o estudo das redes neurais, a maioria dos problemas considerandos como sendo do mundo real não obedecem o critério de linearidade que os limita. Devido a essas limitações, os *perceptrons* são ineficientes para resolução desses problemas, por não serem capazes de modelar funções complexas. As redes *multilayers perceptrons*, comumente conhecidas como MLP, foram idealizadas para suprir a necessidade de uma estrutura mais robusta capaz de modelar funções do mundo real. A MLP trata-se de uma rede composta por mais de uma camada de *perceptrons* encadeada. A Figura 2.2 mostra de forma abstrata uma MLP com três camadas. Cada círculo (ou nó) representa um *perceptron* (ou neurônio), as setas nas extremidades da rede são as entradas e as saídas, respectivamente, enquanto as que ligam um nó a outro representam a transferência de informação entre as camadas.

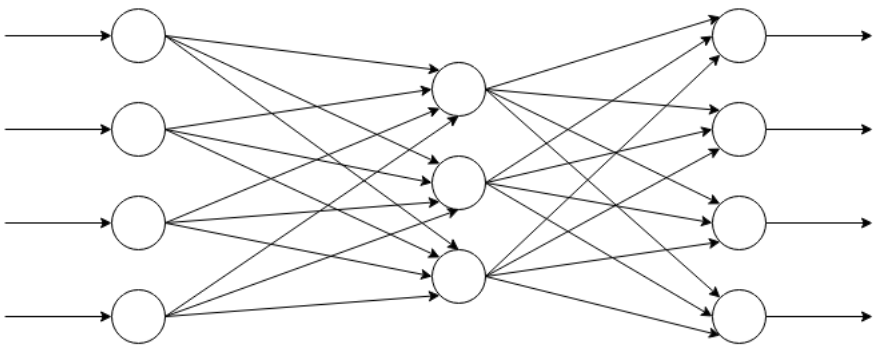


Figura 2.2: Exemplo de uma *multilayer perceptron* com três camadas.

Este é um tipo de rede *feed-forward*, onde toda a informação contida na rede segue o sentido através dos *perceptrons* na entrada em direção aos perceptrons de saída.

Cada neurônio da rede MLP funciona da mesma forma que um *perceptron* comum, onde as fórmulas expressas na Seção 2.1 descrevem o seu comportamento de forma individual. Cada neurônio possui um conjunto de pesos associado às suas entradas, levando uma rede MLP a possuir uma quantidade considerável de pesos que precisam ser ajustados para modelar a função que melhor se adapte ao problema. O método mais comum para o ajuste desses pesos recebe o nome de ***backpropagation***. Esse algoritmo inicia os valores dos pesos de cada um dos neurônios de forma aleatória, no qual normalmente são valores baixos, entre -0.5 e 0.5. Cada iteração desse algoritmo alimenta a rede MLP com informações da entrada para a saída (*feed-forward*). Em seguida, o algoritmo retorna erros pela rede, das saídas em direção a entrada, na medida que ajusta os pesos. É justamente este passo de ajuste que dá o nome ao algoritmo. Essa sequência é repetida até as saídas possuírem os valores desejados, ou seja, o erro se tornar um valor consideravelmente baixo. Dada uma função de ativação simples, uma derivada parcial é utilizada para encontrar o erro da saída em relação ao desejado. Esta operação gera um gradiente de erros, que é calculado para cada nerônio k em uma rede MLP (Coppin 2004).

2.3 Redes Neurais Recorrentes

As redes neurais *feed-forward* não possuem nenhum tipo de ciclo dentro da sua estrutura, por causa da característica de transportar a informação em apenas uma direção, partindo da primeira até última camada. Assim, as redes *feed-forward* não são capazes de ponderar novas informações com informações já existentes na rede. Essa característica faz as redes *feed-forward* não possuírem memória.

Diferentemente das redes MLP convencionais, as redes neurais recorrentes (RNN do inglês *Recurrent Neural Network*) possuem uma certa quantidade de informação obtida através da realimentação das saídas de um ou mais neurônios. Desta forma, os estados internos das redes recorrentes vão se alterando de acordo com as novas entradas e também ponderando seus valores por estados passados, configurando uma estrutura que representa uma memória. A Figura 2.3 mostra duas redes recorrentes distintas, sendo que em uma delas a realimentação é feita de um neurônio para os outros e não para ele mesmo, enquanto que na outra há somente realimentação do neurônio para ele próprio. Em uma rede com mais de uma camada, a realimentação pode ser de um neurônio para cada camada interna anterior, ou apenas da ultima camada para a primeira (Haykin 2007).

Essa realimentação entra na rede na forma de mais um valor para a somatória das multiplicações das entradas pelos pesos, como mostrado na equação (2.1). Essa habilidade da RNN é útil para a solução de problemas que não dependem apenas da entrada atual, mas que também possuem dependência de estado anteriores (Coppin 2004).

2.4 Redes Neurais Recorrentes *Long Short-Term Memory*

As redes neurais recorrentes comuns fazem uso das realimentações para armazenar representações de eventos recentes das entradas, o que recebe o nome de memória de curto prazo (*short-term memory*). Essa característica é bastante útil para diversos tipos

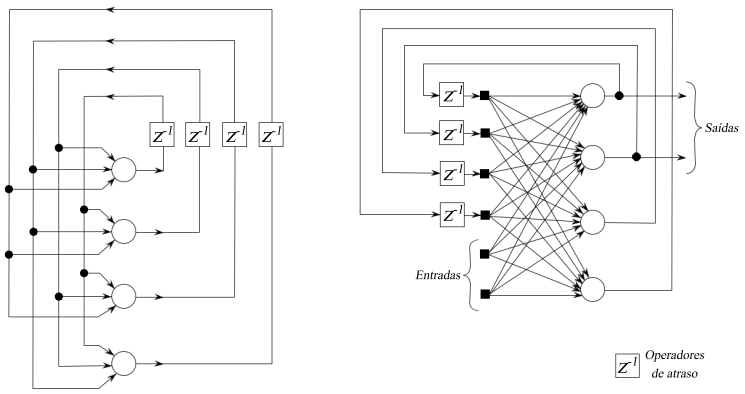


Figura 2.3: Exemplo de uma rede neural recorrente.

de aplicações que não necessitam da influência tão longa dos estados passados. Existem algoritmos e arquiteturas de rede mais complexas que tentam prover um melhor aprendizado desses termos nas redes recorrentes comuns, porém o processo de seu treinamento é muito complexo e lento e ainda podem não funcionar tão bem (Hochreiter & Schmidhuber 1997). Em termos práticos, esses algoritmos não levavam muita vantagem nos resultados em relação às redes *feed-forward* com janelas de tempo limitadas. Devido a essa dificuldade na busca de um limiar entre uma boa memória de curto prazo e uma metodologia de aprendizado que fosse praticável, dois problemas surgiram no cálculo dos pesos das redes recorrentes. Os gradientes calculados a partir da derivada do erro para o ajuste dos pesos podem ficar muito grandes ou desaparecer, o que é conhecido como *blow up* e *vanish* dos gradientes, respectivamente. O *blow up* (ou *explode*) pode gerar oscilações nos valores dos pesos e o *vanish* pode fazer a rede demorar um tempo impraticável para assimilar a capacidade de dependência temporal esperada e ainda assim podendo não funcionar como o esperado (Hochreiter & Schmidhuber 1997).

Para solucionar tais problemas de forma efetiva, a rede neural recorrente *Long Short-Term Memory* (LSTM) foi proposta. Juntamente com a proposta da estrutura da rede, foi apresentado um recálculo para os gradientes obtidos no ajuste do pesos através do *back-propagation* (Hochreiter & Schmidhuber 1997). Este tipo de rede recorrente tem a capacidade de reter informações de uma maior quantidade de entradas passadas (*long-term memory*) enquanto mantém a relevância de estados recentes maior. Para isso, a LSTM faz uso de estados denominados como *cell state* e *hidden state*, que são responsáveis pelo transporte de informações através dos neurônios da rede. A Figura 2.4 mostra uma visão geral de um neurônio (também chamado de célula) de uma rede LSTM.

A partir da Figura 2.4 é possível observar que dado um instante no tempo t, a célula LSTM possui como entradas o instante atual de alimentação de informação da rede, identificado como x_t, o estado oculto (*hidden state*) h_{t-1} e o estado da célula (*cell state*) c_{t-1}, ambos estados provindos da recorrência do instante de tempo passado $t-1$. As saídas da célula são o *cell state* c_t do instante atual, o *hidden state* h_t e a saída de informação y_t. Para o caso da célula pertencer a última camada da rede, o h_t é entendido como a saída

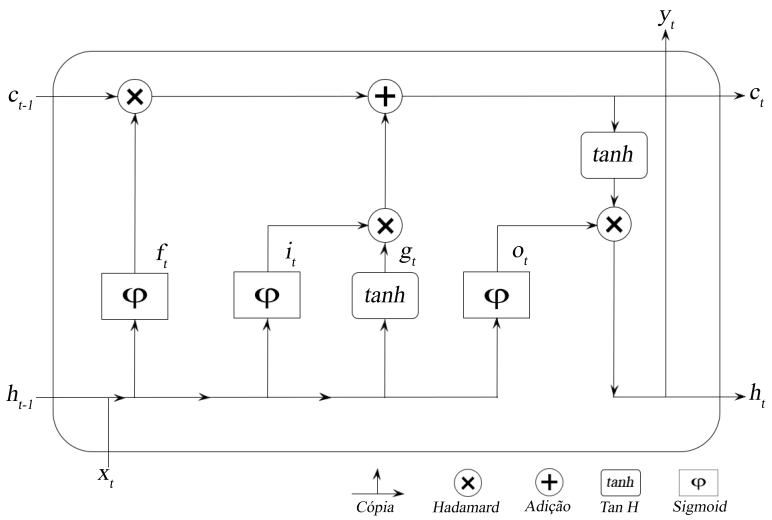


Figura 2.4: Representação de uma célula LSTM.

final y_t, para o caso da camada ser interna a rede, o h_t servirá como o h_{t-1} para a próxima camada adiante na rede.

Além das entradas e saídas, uma célula LSTM é composta internamente por combinações entre funções de ativação, adições e produtos. Estas operações internas da célula LSTM se denominam de *gates*, sendo eles, *forgot gate*, *input gate*, *cell gate* e *output gate*. Além destes *gates*, a célula LSTM possui uma região responsável por agrupar a saída de alguns destes *gates* para produzir o c_t, que é uma das saídas da célula. Cada uma destas estruturas será apresentada a seguir.

2.4.1 *Forgot Gate*

Este componente do neurônio LSTM é responsável pela capacidade da rede esquecer uma informação retida pelos estados passados. A Figura 2.5 destaca o *forgot gate* na célula LSTM.

A partir da Figura 2.5 é possível observar que duas das entradas da célula, h_{t-1} e x_t, se unem nesse ponto e em seguida passam por uma função de ativação φ. A equação (2.6) mostra o processamento feito pelo *forgot gate*.

$$f_t = \varphi(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \tag{2.6}$$

Sendo que x_t é a entrada da célula no instante de tempo atual t, h_{t-1} é o *hidden state* no instante de tempo passado e $W_{if} \in \mathbf{R}^{H\times X}$, $W_{hf} \in \mathbf{R}^{H\times H}$, $b_{if} \in \mathbf{R}^{X}$ e $b_{hf} \in \mathbf{R}^{H}$ são os pesos e *bias* da entrada do *forgot gate* (*input forgot, "if"*) e do *hidden state* (*hidden forgot, "hf"*), respectivamente. As variáveis H e X são as quantidades de neurônios na rede e de entradas, respectivamente. Por fim, a letra grega φ representa a função de ativação *Sigmoid* e f_t a saída do *forgot gate*.

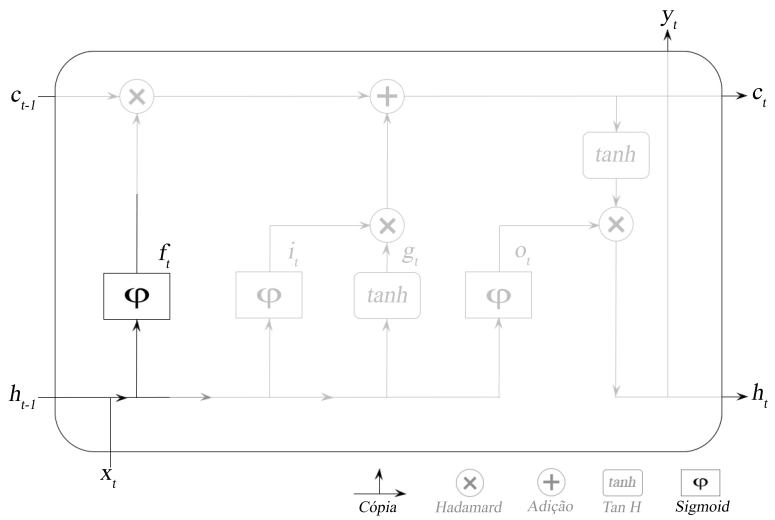


Figura 2.5: Localização do *forgot gate* na célula LSTM.

2.4.2 *Input Gate* e *Cell Gate*

O *input gate* e o *cell gate* são responsáveis por decidir o quanto de informação nova da entrada será inserida na rede. A Figura 2.6, destaca o *input gate* à esquerda e o *cell gate* à direita na célula LSTM.

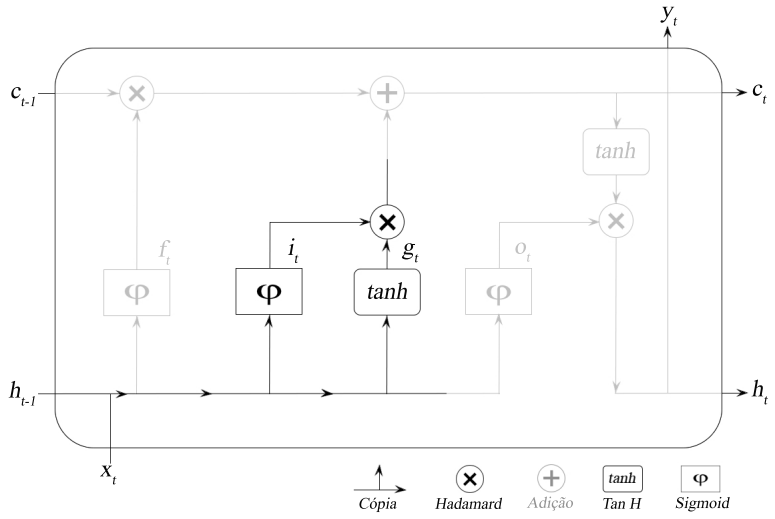


Figura 2.6: Localização do *input gate* e *cell gate* na célula LSTM.

A partir da Figura 2.6 é possível observar que a combinação das entrada h_{t-1} e x_t passam por duas funções de ativação diferentes, φ e *tanh*. O conjunto de equações (2.7) mostra o processamento feito pelos *input* e *cell gates*.

$$
\begin{aligned}
i_t &= \varphi(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\
g_t &= tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})
\end{aligned}
\tag{2.7}
$$

No caso, x_t é a entrada da célula no instante de tempo atual t, h_{t-1} é o *hidden state* no instante de tempo passado e $W_{ii} \in \mathbf{R}^{H\times X}$, $W_{hi} \in \mathbf{R}^{H\times H}$, $b_{ii} \in \mathbf{R}^{X}$ e $b_{hi} \in \mathbf{R}^{H}$ são os pesos e *bias* da entrada do *input gate* (*input input, "ii"*) e do *hidden state* (*hidden input, "hi"*), respectivamente. A letra grega φ representa a função de ativação *Sigmoid* e i_t a saída do *input gate*. Para a segunda equação, $W_{ig} \in \mathbf{R}^{H\times X}$, $W_{hg} \in \mathbf{R}^{H\times H}$, $b_{ig} \in \mathbf{R}^{X}$ e $b_{hg} \in \mathbf{R}^{H}$ são os pesos e *bias* da entrada do *cell gate* (*input cell, "ig"* [1]) e do *hidden state* (*hidden cell, "hg"*), respectivamente. A segunda função de ativação trata-se da Tangente Hiperbólica, representada por *tanh*. A saída do *cell state* é denominada de g_t.

2.4.3 *Cell Update*

O *cell update* corresponde ao conjunto de operações realizado para a obtenção do *cell state*, que será propagado da respectiva célula para a do próximo instante de tempo. O *cell update* é responsável por atualizar o valor do *cell state*, que tem a função de carregar as informações resultantes das operações internas da célula sem a participação do *output gate*. A Figura 2.7 destaca o posicionamento das operações feitas no *cell update*.

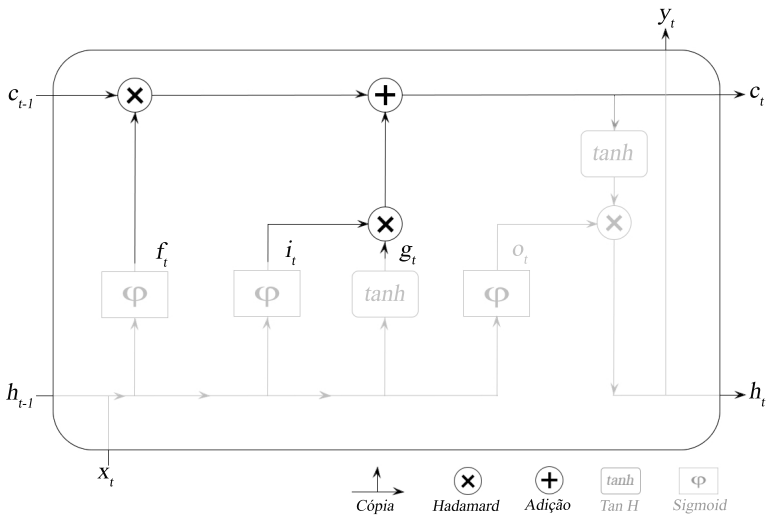


Figura 2.7: Localização da região *cell update* na célula LSTM.

A partir da Figura 2.7 é possível observar a multiplicação entre o *cell state* no instante passado c_{t-1} com a saída do *forgot gate* f_t, este valor é então somado com o resultado obtido a partir do produto entre o *input gate* i_t com o *cell gate* g_t. A equação (2.8) mostra o processamento feito pelo *cell update*

[1] Neste caso a letra *g* foi usada para não existir nenhuma confusão com o *cell state c*.

$$c_t = f_t * c_{(t-1)} + i_g * g_t \tag{2.8}$$

O símbolo $*$ representa a produto de Hadamard, também conhecido como multiplicação elemento a elemento. O resultados dessas operações resultam no *cell state* c_t, que servirá como uma das entradas da próxima célula LSTM.

2.4.4 *Output gate* e *Hidden State*

O *output gate* é o responsável por processar parte da informação que servirá como a saída y_t da rede. O *hidden state* é, como anteriormente mencionado, um estado que carrega informações de instantes anteriores que serão propagadas para os instantes a frente na rede. A Figura 2.8 indica onde o *output gate* e o *hidden state* estão situados na célula LSTM.

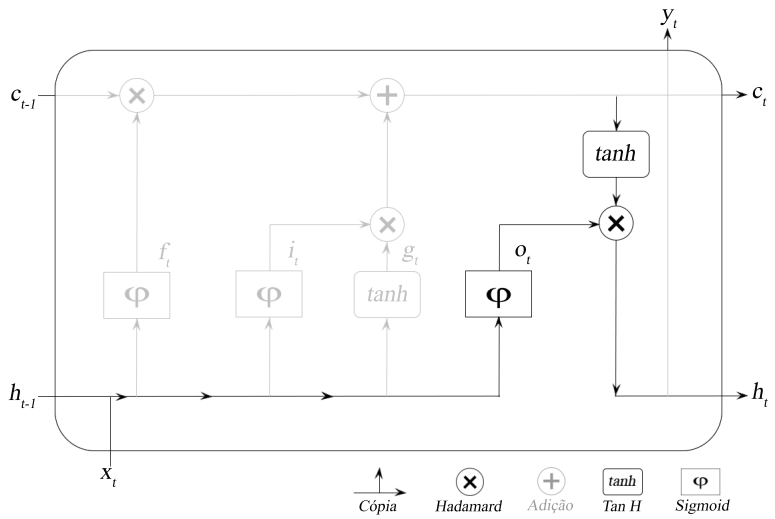


Figura 2.8: Localização do *output gate* na célula LSTM.

A partir da Figura 2.8 é possível identificar que o *output gate* processa as informações provindas da entrada x_t e do *hidden state* no instante passado h_{t-1}. Tendo isso, o *hidden state* do instante atual é gerado a partir de uma combinação do *output gate* com o *cell state*, que também foi transformado por uma função de ativação. O conjunto de equações (2.9) mostra o processamento feito pelo *output gate* e como o *hidden state* h_t é obtido.

$$\begin{aligned} o_t &= \varphi(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\ h_t &= o_t * tanh(c_t) \end{aligned} \tag{2.9}$$

Na primeira equação do conjunto de equações (2.9), $W_{io} \in \mathbf{R}^{H\times X}$, $W_{ho} \in \mathbf{R}^{H\times H}$, $b_{io} \in \mathbf{R}^X$ e $b_{ho} \in \mathbf{R}^H$ são os pesos e *bias* da entrada do *output gate* (*input output, "io"*) e do *hidden state* do instante de tempo passado (*hidden output, "ho"*), respectivamente. A

letra grega φ representa a função de ativação *Sigmoid* e o_t a saída do *output gate*. A segunda igualdade mostra a obtenção do *hidden state* h_t a partir do *output gate* o_t e da transformação do *cell state* c_t pela função de ativação Tangente Hiperbólica $tanh$.

De modo geral, esse é o funcionamento de uma célula LSTM. A composição interna dessas redes, tornam os estados internos das células responsáveis por propagar a informação dos estados passados através da rede, de modo a não necessitar de uma realimentação explicita, o que as diferencia estruturalmente das RNNs comuns. Essa estrutura de células independentes às aproxima das MLP, no modo como a informação segue o fluxo de forma direta através da entrada até a saída da rede.

Capítulo 3

Detecção e Identificação de Falhas

Dada a importância do problema de detecção e isolamento de falhas, é relevante situar onde o modelo que será avaliado e a proposta deste estudo se encontram na literatura. Assim, neste capítulo será apresentada uma breve definição sobre a área de detecção e isolamento de falhas em processos industriais, além de uma divisão de como as metodologias e seus respectivos modelos que tratam dese tema se organizam, destacando-se os principais trabalhos disponíveis na literatura.

Um dos principais conceitos para o diagnóstico de falhas em operação de processos é a definição do que venha ser a própria falha. Uma falha pode ser expressa de forma genérica como um afastamento do comportamento do processo em relação ao intervalo de valores aceitáveis para um conjunto de variáveis ou parâmetros calculados para ele (Himmelblau 1978). Essa definição mostra a falha como um comportamento anormal ou sintoma. Por exemplo, podemos citar valores excessivos para uma alta temperatura em um reator ou a baixa qualidade de um produto. Esses comportamentos possuem causas conhecidas como evento básico, que ocorrem devido a um funcionamento defeituoso de algum componente do sistema, como uma bomba de refrigeração ou um controlador (Venkatasubramanian, Rengaswamy, Yin & Kavuri 2003). A figura 3.1 mostra um sistema de processo controlado e indica os diferentes fontes de falhas presentes nele.

Baseado na figura 3.1, é possível categorizar basicamente três classes de falhas ou de mau funcionamento. As falhas de parâmetros, as falhas estruturais e as falhas em atuadores e sensores. Falhas de parâmetro são observadas quando existe uma pertubação entrando no processo por meio de uma ou mais variáveis externas. Um exemplo desse tipo de falha é a mudança no valor de concentração do reagente na alimentação de um reator. Falhas estruturais referem-se a alterações no próprio processo, que ocorrem devido a falhas graves em equipamentos, como em um tubo quebrado ou vazando. Este tipo de mau funcionamento resulta em uma mudança no fluxo de informações entre várias variáveis. Falhas em atuadores e sensores podem ocorrer devido a uma falha constante no componente, ou com um *bias* positivo ou negativo. Uma falha em um destes instrumentos pode fazer com que as variáveis de estado da planta se desviem além dos limites aceitáveis pelo processo, o que pode gerar um mau funcionamento grave (Venkatasubramanian, Rengaswamy, Yin & Kavuri 2003).

A tarefa do sistema de FDI pode ser analisada como um problema de classificação de dois estágios, sendo estes a detecção e o isolamento das falhas. A detecção de falha

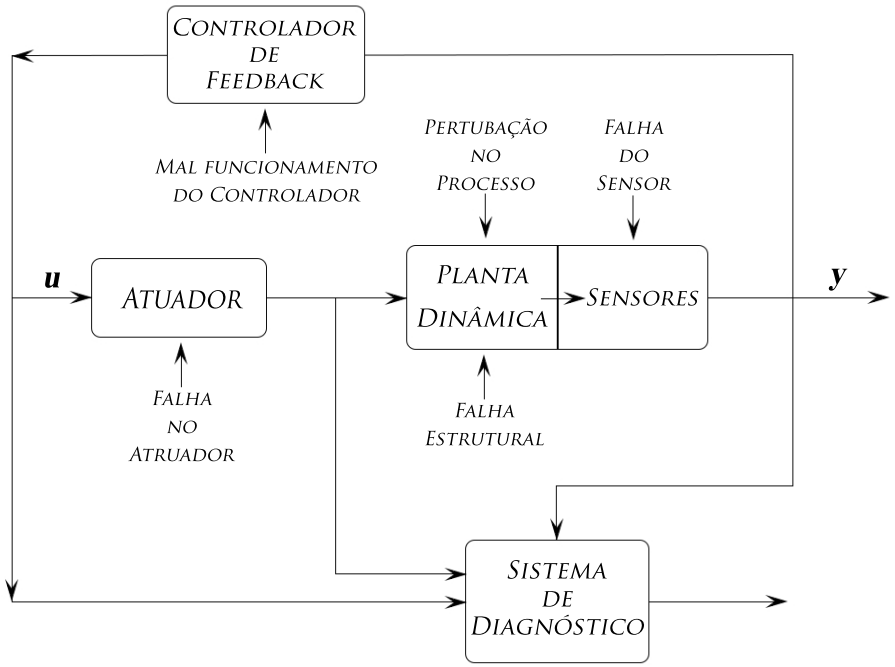


Figura 3.1: Estrutura geral para o diagnóstico de falhas.

é a atividade em que se identifica quando o sistema está operando em um estado normal ou em um estado anormal. Neste estágio, informações importantes sobre a falha, como a sua localização e duração, não estão em foco nesta etapa. A detecção de falhas pode ser considerado um problema de duas classes, onde dada uma amostra do comportamento da operação do processo, determina-se se ele encontra-se em um estado normal ou anormal. De outro modo, o isolamento de falhas corresponde a determinar qual tipo, localização e tempo para a detecção de determinada falha. O objetivo é combinar os padrões detectados com classes pré-determinadas, ou descobrir classes ainda não catalogadas previamente (Precup et al. 2015).

De modo geral, as ferramentas para um sistema FDI são divididas em basicamente três tipos: modelos baseados em métodos quantitativos, modelos baseados em métodos qualitativos e métodos baseados no histórico de processo (ou baseados em dados) (Venkatasubramanian, Rengaswamy, Yin & Kavuri 2003). A Figura 3.2 mostra a classificação dos tipos de modelos mais comuns utilizados para o problema de FDI.

É necessário deixar explícito que todos os tipos de métodos utilizam de dados para estimar parâmetros, porém a classificação desses três tipos é feita a partir da maneira no qual as metodologias abordam o problema (Venkatasubramanian, Rengaswamy, Yin & Kavuri 2003).

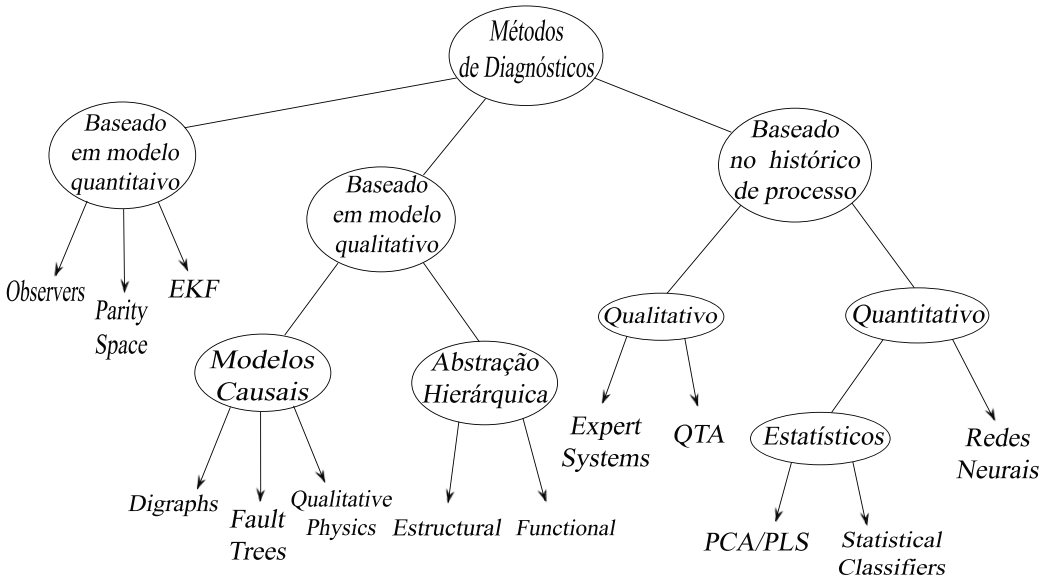


Figura 3.2: Classificação dos tipos de modelos para o problema de FDI.

3.1 Modelos Baseados em Métodos Quantitativos

Os modelos qualitativos e quantitativos são desenvolvidos para compreenderem as informações baseados em características e informação físicas do processo. Em particular, os modelos quantitativos têm essa compreensão expressa de modo a obter as relações entre as entradas e saídas do processo através de números. Dessa maneira, funções matemáticas e estatísticas são as formas mais comuns para descrevê-las. A maioria desses modelos é desenvolvida baseada no conceito de sinais de entradas e saídas do processo. Mas, além destes, existem modelos baseados em primeiros princípios, resposta em frequência, dentre outros. Modelos de primeiros princípios não são muito populares para o problema de FDI, devido à sua alta complexidade computacional, o que compromete a sua implementação em problemas reais (Venkatasubramanian, Rengaswamy, Yin & Kavuri 2003).

Alguns dos modelos mais utilizados na categoria dos quantitativos são os *Diagnostic observers for dynamic systems* (Frank & Wunnenberg 1989), *Parity Relation* (Gertler 1991), *Kalman Filters* (Willsky & Jones 1976) e *Parameters Estimation* (Isermann 1984). A Figura 3.3 mostra de modo geral como são as estruturas de modelos quantitativos para o problema de FDI. A partir dessa classificação, é possível observar que as saídas reais da planta e estimadas pelo modelo são comparadas e, então, algum tipo de medida de resíduo é obtida para representar essa diferença. Com isso, o valor resultante segue para o sistema FDI, que é o responsável por retornar informações a respeito das falhas.

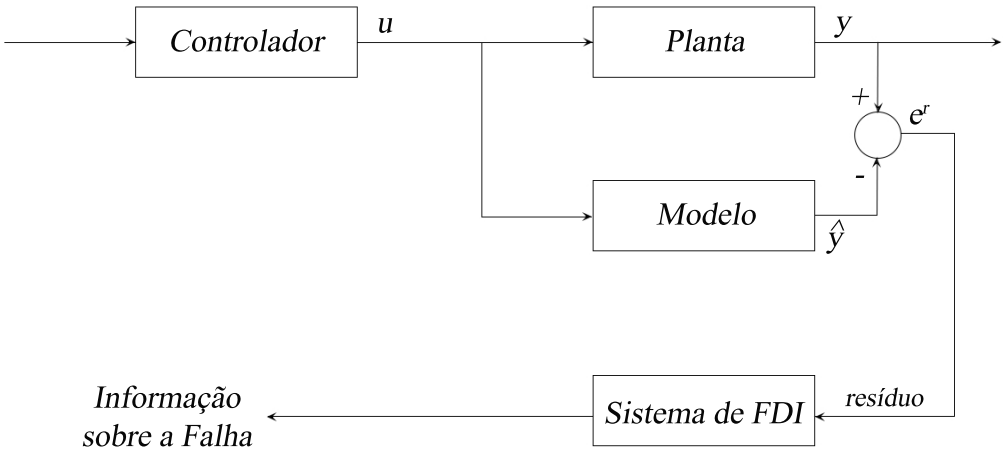


Figura 3.3: Estrutura geral de um modelo quantitativo para FDI.

3.2 Modelos Baseados em Métodos Qualitativos

Para compreender as informação presentes no processo, os modelos qualitativos fazem uso de funções que não apenas buscam medir valores. Esses modelos observam relações entre diferentes unidades do processo, de modo a avaliá-lo de forma não numérica, e sim com intuito de obter resultados de forma particular a respeito das causas de tais comportamentos. Além disso, não possuem entendimento da física do sistema. Assim não apresentam bom desempenho em casos quando há aparecimento de novas condições não pré-definidas. Esse tipo de modelo é dividido em modelos causais e em abstração hierárquica (Figura 3.2). Dentre destes, existe a busca topográfica e a busca sintomática (Venkatasubramanian, Rengaswamy & Kavuri 2003).

A busca topográfica analisa o mau funcionamento do processo, utilizando-se como base um modelo que descreva o funcionamento normal da operação. Enquanto que na busca sintomática é feita uma procura por indicadores que direcionem a análise para o ponto de ocorrência do mau funcionamento. Alguns dos principais modelos desses tipos de busca são os *Signed Digraphs* (SDG) (Iri et al. 1979), *Fault trees* (Lapp & Powers 1977), *Qualitative physics* (Iwasaki & Simon 1986) e *Hypothesis and test search* (Petti et al. 1990). A Figura 3.4 mostra de modo geral como são as estruturas de modelos qualitativos para o problema de FDI. É possível observar a partir dessa classificação que nesse tipo de modelo se obtém representações para formar uma base de características qualitativas, que são enviadas a um detector de discrepâncias entre as representações e as saídas do processo. Após essa fase, as discrepâncias são enviadas ao sistema responsável de retornar informações sobre as falhas (Sistema FDI).

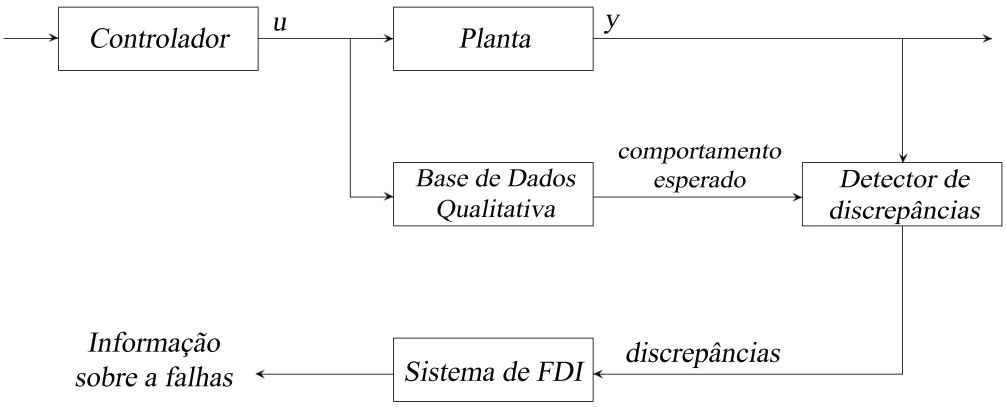


Figura 3.4: Estrutura geral de um modelo qualitativo para FDI.

3.3 Métodos Baseados no Histórico das Variáveis do Processo

Nestes tipos de métodos, a única informação utilizada é o histórico de dados sobre comportamento do processo. Existem maneiras de como apresentar estes dados, de forma a ser feita uma extração de recursos ou características. Esse tipo de ação pode ser de natureza qualitativa ou quantitativa (Venkatasubramanian, Rengaswamy, Kavuri & Yin 2003). A Figura 3.5 mostra a taxonomia dos modelos baseados em histórico de variáveis de processo.

Nos modelos qualitativos, o destaque são para os tipo *Expert systems* (Chester et al. 1984) e *Qualitative trend analysis* (QTA) (Konstantinov & Yoshida 1992). Para os modelos baseados em histórico de variáveis do tipo quantitativas, há os métodos estatísticos e não-estatísticos. Os principais métodos estatísticos são o *Principal Component Analysis* (PCA) (Raich & Çinar 1997) e os *Statistical classifiers* (Leonard & Kramer 1993), enquanto que para os não-estatísticos são as Redes Neurais. A Figura 3.6 mostra de modo geral como são as estruturas de modelos baseados em histórico de variáveis de processo para o problema de FDI. O destaque dessa representação é para o extrator de características e para o especialista. O especialista é responsável por apontar se aquele conjunto de dados analisado corresponde a uma situação de falha ou não. A saída do extrator de características do processo será comparada com a base de características já catalogada. Após essa fase, as diferenças são enviadas ao sistema FDI, que é o responsável de retornar as informações sobre as falhas.

Como é possível observar a partir da Figura 3.5, os modelos baseados em histórico de variáveis de processo também possuem as categorias qualitativa e quantitativa, sendo nesta segunda que se encontram o objeto de estudo deste trabalho. Mais especificamente,

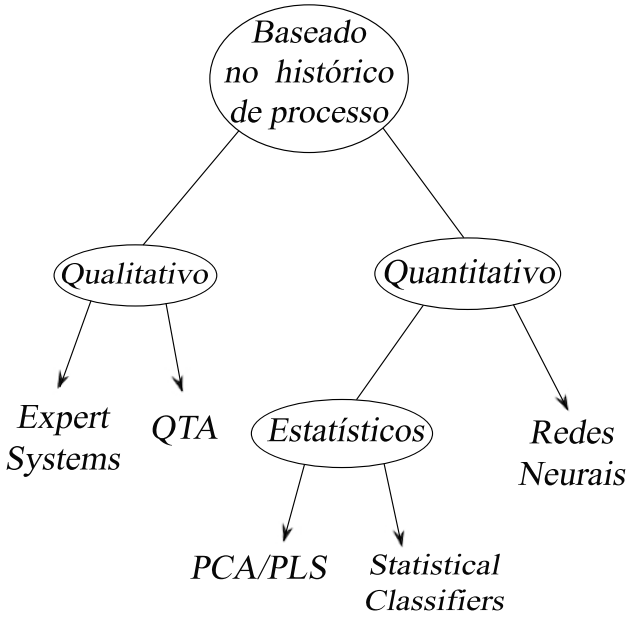


Figura 3.5: Divisão dos tipos de modelos baseados em histórico de variáveis de processo para o problema de FDI.

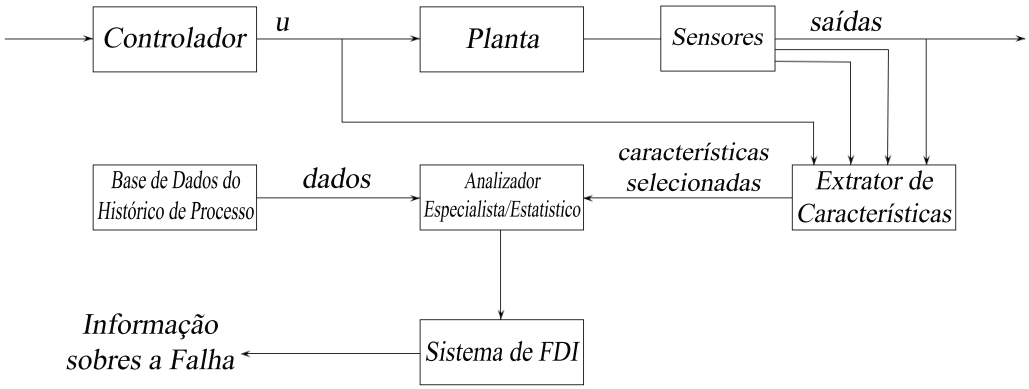


Figura 3.6: Estrutura geral de um modelo baseado em histórico do processo para o problema de FDI.

modelos baseados em Redes Neurais (*Neural Networks*).

3.4 Modelos Baseados em Redes Neurais para FDI

Nesta seção serão apresentadas abordagens disponíveis na literatura que são baseadas no histórico de processo e utilizam de redes neurais para solucionar o problema de FDI. Serão mostrados métodos que fazem uso de redes mais tradicionais, como a MLP, até redes recorrentes, além de outras técnicas de *deep learning* e processamento de sinais. É relevante ressaltar que nesta dissertação será avaliado um método que utiliza uma rede recorrente LSTM dentro de sua metologia.

Os primeiros trabalhos apresentados para solucionar o problema de FDI utilizando redes neurais, naturalmente, fizeram uso do tipo mais simples de rede neural, a MLP. Estes trabalhos serviram para mostrar a capacidade desse tipo de metodologia em lidar com as informações contidas no histórico de processo dos sistemas industriais. Sorsa et al. (1991) utilizam uma MLP com uma tangente hiperbólica como função de ativação, para um estudo de caso em um sistema de troca de calor continua em um reator de agitação. Este sistema possui dez falhas a serem classificadas, onde a rede neural utilizada foi detectou e classificou as falhas de forma satisfatória, após ser treinada por aproximadamente 3000 ciclos. Em Ayoubi (1994), é proposta a utilização de um modelo processador elementar dinâmico (*Dynamic Elementary Processor*, DEP) junto com uma MLP, com o objetivo de adicionar uma dinâmica distribuída a rede. A partir destes modelos dinâmicos foi criado um bando de esquemas, que foram comparados com a saída do processo, gerando um resídio utilizado para detectar falhas caso tenham ocorrido. Este sistema foi aplicado para monitorar os estados de uma turbina de um turbo-alimentador. Abordagens que usam de redes MLP ainda são propostas em conjunto com outras técnicas de aprendizado de máquina, como uma alternativa simples para solucionar o problema (Jedliński & Jonak 2015, Heo & Lee 2018).

A medida que os estudos a cerca do problema do FDI foram se desenvolvendo, o potencial das redes neurais recorrentes foi enxergado, isso em muito se deve a sua particular afinidade em lidar com processos que possuem alguma dependência temporal entre suas amostras. Essa afinidade das RNN foi explorado de maneira a apresentar um certa tendência de combinar diferentes técnicas para a FDI. Para estes casos, um método é usada para extrair características das falhas e outro fica responsável por classificá-las. Técnicas como, *Spectral Kurtosis* (SK) com *Extreme Machine Learning* (EML) (Udmale & Singh 2019) foram utilizadas juntas, além das *Convolutional Neural Networks* (CNN). As CNNs se demonstraram uma tendência para a solução do problema da FDI (Chen et al. 2015, Lee et al. 2017, Tian et al. 2018, Dey et al. 2017, Hemmer et al. 2018, Liu et al. 2018). Outro método proposto que faz uso de *Convolutional Networks* é apresentado por Wen et al. (2017). No estudo, é mostrado uma CNN baseada na LeNet-5 (LeCun & others 2015) para diagnósticos de falhas através da conversão de sinais em imagens 2-D. Essas imagens são geradas observando trechos do sinal que serão passados por uma função (proposta pelos autores) que retorna valores em escala de cinza para cada pixel da imagem a ser gerada. Cada um desses trechos é responsável por atribuir valores a 2^n *pixels*, onde n é o tamanho de cada saída da função proposta, gerando assim uma imagem

de tamanho $M \times M$ *pixels*, com $M = 2^n$. O valor de M varia de acordo com o tamanho da variável de entrada e da quantidade de dados presente no conjuntod e dados. Após esse pre-processamento, uma imagem é gerada no qual será inserida na rede LSTM para ser classificada. Este modelo foi testado em três estudos de caso, sendo um para diagnósticos de falhas em rolamento de motor, outro para diagnóstico de falhas em bombas centrífugas auto-escovantes e um último no diagnósticos de falhas em bombas hidráulicas.

As redes LSTM passaram a ganhar destaque por sua capacidade melhorada de armazenar informações de curto e longo prazo, sendo combinadas com diferentes tipos de métodos. Yang et al. (2018) faz uso de *Fast Fourier Transform* (FFT) para extração de componentes de frequência do sinal, seguido de uma projeção aleatória gaussiana para reduzir o vetor resultante da FFT. Após esses processamentos, uma rede LSTM é treinada para efetuar a classificação de falhas em uma abordagem que pode ser empregada de forma *off-line* e *on-line*, para classificar falhas de um sistema de transmissão de uma turbina eólica (Yang et al. 2018). Seguindo a linha de uso de artifícios que utilizam da frequência do sinal, Li et al. (2018) propõe um modelo baseado em *Discrete Wavelet Transform* (DWT) e redes LSTM para detectar e classificar falhas em motores síncronos de ímãs permanentes. Inicialmente o sinal é decomposto em seis bandas de frequência pela DWT, em seguida, a energia de cada banda é calculada para servir de recurso para a classificação. Após isso, são obtidos quatro características de classificação, no qual são utilizadas como entrada da rede LSTM.

Além dos métodos já citados, alguns outros tipos de técnicas ja foram utilizadas juntamente com a LSTM. Zhang et al. (2017) propõe uma rede LSTM com o objetivo de fazer a captura das características do sinal, que passa por uma normalização e em seguida é amostrado em entradas com regressores. Posteriormente, se fazer uso de uma *Support Vector Machine* (SVM) para realizar a classificação das falhas. O método tem a sua fase de aprendizado, ou treinamento, feita de forma *off-line* e segue sendo atualizado de forma *on-line*. O modelo é utilizado para compreender cada tipo de falha e classificá-las (Zhang et al. 2017).

Em Yuan et al. (2016) é proposto o uso da LSTM para construir um modelo capaz de diagnosticar múltiplas falhas e falhas híbridas. Além do diagnóstico, o modelo é capaz de estimar o tempo de vida útil de Aero Motores. Inicialmente, foi usada uma SVM para detecção de falhas no conjunto de treinamento, categorizando-as e estimando o tempo de vida restante para o motor. Os dados são então normalizados e enviado para o treinamento da rede. Após a LSTM ser treinada, ela é capaz de projetar duas saídas, onde uma se trata do valor de tempo de vida restante do Aero Motor e a segunda saída indica a falha (caso haja) que corresponde à aquela entrada (Yuan et al. 2016).

No trabalho desenvolvido por De Bruin et al. (2016), é afirmado que as redes LSTM se adéquam melhor para a detecção e classificação de falhas em circuitos ferroviários que os demais tipos de redes. A proposta aborda o problema de FDI determinando a causa da falha, através da criação de um conjunto de dados que simula o comportamento real de um circuito ferroviário. A rede LSTM é treinada, utilizando de uma camada *Softmax*, responsável por normalizar a distribuição de probabilidades da saída, com o objetivo de decidir qual das falhas correspondem a respectiva entrada (De Bruin et al. 2016).

He et al. (2019) propõe um método que combina uma CNN e uma LSTM para reali-

zar a detecção e classificação de falhas graduais. A CNN é utilizada para executar uma extração de características dos diagramas de abstração multinível do processo, retornando esses indicadores para a LSTM executar a classificação das falhas. Além destes citados, outros trabalhos também fazem uso de redes LSTM em diferentes tipos de aplicação, como solução para o problema de FDI (Lei et al. 2019, Yu et al. 2019).

Capítulo 4

Metodologia para Classificação de Falhas

Como discutido no Capítulo 3, existem diversos tipos de abordagens para o tratamento do problema de FDI. Nesta dissertação, o estudo se limita ao problema de classificar falhas previamente detectadas. Para executar essa tarefa, aqui escolhemos uma abordagem baseada em rede neural recorrente do tipo LSTM (Zhao et al. 2018) em conjunto com um algoritmo de reparametrização de camada, chamado de *Batch Normalization* (BN) (Ioffe & Szegedy 2015). Neste capítulo será mostrado em detalhes a estrutura utilizada para a classificação das falhas, como também a metodologia proposta para avaliar o seu desempenho com relação aos diversos hiper-parâmetro da rede neural LSTM.

No Capítulo 2 foram mostradas as funções, as estruturas e o funcionamento de uma célula LSTM. Na proposta original da LSTM (Hochreiter & Schmidhuber 1997), há uma fator associado com o cálculo de truncamento dos gradientes. Porém, com o avanço nas pesquisas em torno desse tipo de rede neural recorrente, várias variações foram propostas e esse fator foi alterado. Estudos mostraram uma melhor convergência no processo de treinamento de rede e dos resultados no armazenamento de informações com um recálculo completo desses gradientes (Graves & Schmidhuber 2005). Esta mudança caracterizou a criação da *Vanilla LSTM*, que foi difundida e atualmente é a variação mais amplamente utilizada em implementações de redes do tipo LSTM segundo Greff et al. (2016). No nosso estudo sobre desempenho de redes neurais recursivas do tipo LSTM para o problema de FDI, adotamos essa abordagem alinhado com o proposto por Zhao et al. (2018).

4.1 Abordagem Utilizada para FDI

Esta seção apresentará as informações sobre a abordagem proposta por Zhao et al. (2018) para classificação de falhas, que servirá de base para os nosso estudo sobre avaliação de desempenho de redes neurais do tipo LSTM no problema de FDI. Aqui serão mostradas as definições da BN, como também das funções *LogSoftMax*, *Negative Log Likelihood Loss* e do algoritmo de otimização *Adam*, usados para o treinamento da rede LSTM. Por fim, será mostrado o fluxo seguido entre cada etapa da proposta para o seu treinamento e classificação das falhas.

4.1.1 *Batch Normalization*

Nas redes neurais artificiais, a saída da primeira camada serve de entrada para a segunda camada, a saída da segunda alimenta a terceira e assim por diante. A medida que a rede ajusta os parâmetros de cada camada, a distribuição destes dados também se altera. Esse fenômeno é denominado de *Internal Covariate Shift* (ICS), que pode causar uma perda de efetividade na velocidade de treinamento da rede neural. Para resolver esse problema, uma solução é fixar a distribuição das entradas na camada inicial da rede neural à medida que o treinamento avança. Essa ideia se baseia no conhecimento prévio de que o treinamento de uma rede neural converge de maneira mais rápida quando a entrada obedece a uma distribuição que possui pouca variação de suas unidades correlacionadas (Wiesler & Ney 2011). Como cada camada da rede neural recebe como entradas os resultados produzidos de outras camadas da rede, seria vantajoso se todas as camadas recebessem como entradas o mesmo comportamento de distribuição. Ao normalizar as entradas de cada camada da rede neural, é possível obter distribuições fixas, o que removeria os efeitos negativos do ICS (Ioffe & Szegedy 2015).

Nesse contexto, a BN foi proposta com o objetivo de evitar o ICS e melhorar a convergência dos modelos nas iterações de treinamento da rede neural LSTM. De modo geral, a técnica padroniza as saídas das funções de ativação de cada camada da rede neural, utilizando-se para isto as estimativas da média $\hat{E}(\mathbf{h})$ e da variância $\hat{Var}(\mathbf{h})$. A equação (4.1) define a BN.

$$BN(\mathbf{h};\eta,\rho) = \rho + \eta * \frac{\mathbf{h} - \hat{E}(\mathbf{h})}{\sqrt{\hat{Var}(\mathbf{h}) + \varepsilon}} \tag{4.1}$$

No caso, as entradas $\mathbf{h}$, η e ρ representam o vetor de características a ser normalizado, o parâmetro que define a média e o parâmetro que define o desvio padrão dessas características normalizadas, respectivamente. Sendo ε o parâmetro de regularização e $*$ representa a multiplicação elemento-a-elemento (Zhao et al. 2018).

Nesta abordagem, o passo da BN possui uma particularidade que é considerada fundamental. A normalização não é feita no termo atual $W_{ii}x_t$, mas sim no termo recorrente $W_{hi}h_{(t-1)}$. Segundo os autores da proposta original, essa alteração proporciona à camada LSTM um melhor controle da contribuição relativa dos termos passados. Além disso, o parâmetro de entrada ρ que define o desvio padrão, foi utilizado como sendo zero, com o objetivo de evitar redundância desnecessária e *overfitting* (Zhao et al. 2018).

4.1.2 LogSoftMax

Após o processamento das entradas ser feito nas camadas da rede LSTM e na camada de reparametrização BN, foi utilizada uma última camada de ativação para se obter a saída final da rede. Para isto o algoritmo *LogSoftMax* foi escolhido. Essa função calcula o logaritmo da razão entre o valor exponencial de uma entrada *i* com relação a soma dos valores exponenciais de todas as entradas exponenciais da rede no mesmo instante de tempo. De modo geral, a camada com esta função é o último passo para a atividade de classificação. A equação (4.2) mostra a operação feita pelo algoritmo.

$$LogSoftMax(y_i) = log\left(\frac{exp(y_i)}{\sum_j exp(y_j)}\right) \tag{4.2}$$

Assim, essa função retorna um conjunto de probabilidades, que indica a qual classe a i-ésima saída da rede pertence, amplificando os resultados com maior probabilidade e diminuindo os com menor.

4.1.3 *Negative Log Likelihood Loss*

Como citado no Capítulo 2, é necessário uma medida para estimar o quão próximo do desejado o modelo consegue retornar uma saída para uma dada entrada na rede neural. Existem diversos tipos de funções para o cálculo desse valor. Dependendo do tipo de saída que se obtém do modelo, essas funções têm o nome de funções de perda ou *loss functions*. Nesta dissertação utilizamos a função *Negative Log Likelihood Loss* (NLLLoss) para a função de perda.

Em um conjunto de classes C, as classes c_1, c_2, ..., c_N pertencem a esse grupo C e representam as falhas a serem classificadas. A função $f(\cdot)$ é uma função indicadora, que irá representar de modo simples a rede que faz a tomada de decisão para qual classe determinada entrada pertence. A equação (4.3) mostra como essa função indicadora funciona para uma entrada booleana s, que é uma expressão a ser avaliada.

$$f(s) = \begin{cases} 1, & \text{se } s \text{ avaliado como verdadeiro} \\ 0, & \text{se } s \text{ avaliado como falso} \end{cases} \tag{4.3}$$

Desse modo, a função de perda NLLLoss é definida na equação (4.4), tendo como argumento da quíntupla $I = (W_h, W_x, W_{yh}, b, b_y)$ (Zhao et al. 2018). No caso, W_h, W_x, W_{yh} representam os pesos dos *hidden states*, da entrada x e da saída y, respectivamente. De mesmo modo, b e b_y representam o *bias*.

$$\begin{aligned} d(\mathbf{x}_t, c_j) &= f(\mathbf{x}_t \in c_j) \\ \mathbf{Loss}(I) &= -\frac{1}{T-J+1} \sum_{t=1}^{T-J+1} \sum_{j=1}^{C} d(\mathbf{x}_t, c_j)\, y_{t,j} \end{aligned} \tag{4.4}$$

A equação (4.4) mostra a função que define o NLLLoss. Sendo que $\mathbf{y}_t$ é a saída da rede para o instante de tempo t. A função indicadora $d(\cdot)$ é usada para contabilizar os acertos da rede, baseados nas classes esperadas das entradas $\mathbf{x_t}$. Neste caso, T é a dimensão do conjunto de entradas $\mathbf{x_t}$ e J é o tamanho de cada entrada.

4.1.4 Adam

O algoritmo escolhido para otimização dos parâmetros da função de cálculo do gradiente estocástico, foi o *Adam* (*Adaptive Moment Estimation*). A vantagem deste método é requerer apenas os gradientes de primeira ordem e pouca memória para o seu uso. O

Adam calcula as taxas de aprendizado de forma adaptativa e individual para os diferentes parâmetros das estimativas dos momentos dos gradientes (Kingma & Ba 2014).

Seja $z(\cdot)$ uma função de objetivo que deseja minimizar os seus valores esperados $\mathbf{E}[z(\cdot)]$ de acordo com o conjunto de parâmetros θ. Com o avanço dos instantes de tempo t, tem-se $z_1(\theta), z_2(\theta), \ldots, z_t(\theta)$. O algoritmo *Adam* atualiza as médias móveis exponenciais do gradiente e do gradiente quadrado utilizando parâmetros que controlam suas taxas de decaimento. As médias móveis são estimativas do primeiro e segundo momento do gradiente, que são a média e a variação não centrada, respectivamente. O conjunto de equações (4.5) descreve os cálculos sequenciais feitos pelo algoritmo de otimização (Kingma & Ba 2014).

$$
\begin{aligned}
g_t &= \nabla_\theta \, z_t(\theta_{t-1}) \\
m_t &= \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \\
v_t &= \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot (g_t * g_t) \\
\hat{m}_t &= m_t / (1-\beta_1^t) \\
\hat{v}_t &= v_t / (1-\beta_2^t) \\
\theta_t &= \theta_{t-1} - (\alpha * \hat{m}_t) / (\sqrt{\hat{v}_t} + \varepsilon)
\end{aligned}
\tag{4.5}
$$

Sendo que g_t denota o vetor gradiente das derivadas parciais de z_t, de acordo com o conjunto θ de parâmetros no instante de tempo t. Os valores das médias móveis exponenciais do gradiente e gradiente quadrado são representados por m_t e v_t, respectivamente. As suas taxas de decaimento exponencial são β_1 e β_2. Essas médias móveis são inicializadas com vetores de zeros, o que pode levar uma estimativa de momento com tendência a esse valor (zero), principalmente nos instantes iniciais de tempo. Para neutralizar esse comportamento, é feito uma correção desses parâmetros utilizando-se as taxas de decaimento daquele respectivo instante de tempo, β_1^t e β_2^t, resultando em $\hat{m}_t$ e em $\hat{v}_t$. A saída desse algoritmo é o conjunto otimizado dos parâmetros de correção θ_t da função objetivo z (Kingma & Ba 2014).

4.1.5 Fluxo para o Treinamento do Modelo

Com as definições de cada um desses componentes e sabendo o funcionamento das redes LSTM (capítulo 2), é possível elaborar a sequência efetuada para treinamento do modelo. A Figura 4.1 mostra um fluxograma do procedimento executado pela metodologia adotada para realizar o treinamento do modelo para classificação das falhas.

É possível observar a partir da Figura 4.1 um destaque para o processo feito camada-a-camada no treinamento da rede LSTM proposta por Zhao et al. (2018). Inicialmente é feito um janelamento dos dados da variável de processo, sendo que J representa o tamanho da entrada da rede, composto pela amostra no instante de tempo atual do processo, e $J-1$ amostras dos instantes de tempo passados. Cada uma dessas entradas é atribuída a uma classe a qual ela pertence, sendo utilizado um valor inteiro para representar a respectiva classe. Em seguida, cada entrada é submetida à rede, no qual passam pelas camadas LSTM, de *Batch Normalization* e *LogSoftMax*. Até esse ponto, se é atribuído

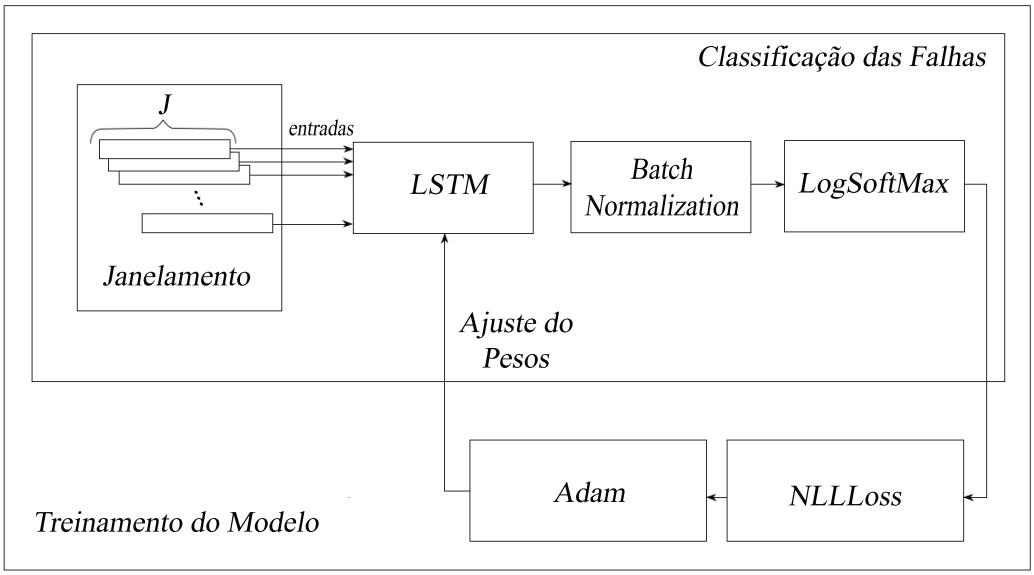


Figura 4.1: Fluxograma da metodologia de classificação de falhas.

o objetivo de classificar falhas, onde os próximos passos são destinados apenas a fase de treinamento e ajustes da rede. Ao final da passagem pela rede, cada entrada do sinal das variáveis de processo possui um saída atribuída correspondente a uma classe de falhas que a rede determinou. Dessa forma, através da função de perda NLLLoss, as saídas da rede são comparadas com as classes reais que aquelas entradas pertencem, configurando um aprendizado supervisionado. Uma vez possuindo os valores dos erros ajustados, o passo de otimização é feito pelo algoritmo *Adam*. Esses passos se repetem no treinamento pela quantidade de épocas escolhida, sendo que em cada época é percorrido todo o conjunto de dados.

4.2 Metodologia Proposta para Avaliação de Desempenho

Uma das propostas desta dissertação, é apresentar uma metodologia para avaliação do comportamento da abordagem utilizada para classificação de falhas. Esta avaliação se baseia em observar a resposta do modelo ao se variar alguns de seus hiperparâmetros. Um dos principais pontos a se dar ênfase, é encontrar quais destes hiperparâmetros são responsáveis pela melhora ou piora dos resultados na classificação. Isso mostra quais fatores podem contribuir para se obter um modelo ótimo, onde não se tenha a necessidade de utilizar redes grandes com um longo processo de treinamento. A figura 4.2 mostra o fluxograma a ser seguido para avaliar a abordagem apresentada na seção 4.1.

O primeiro passo a ser seguido é a escolha de uma abordagem para realizar a FDI. Seguido da seleção do conjunto de dados a ser utilizado para avaliar o modelo. Este passo de escolha do conjunto de dados, inclui definir as falhas que serão utilizadas e qual

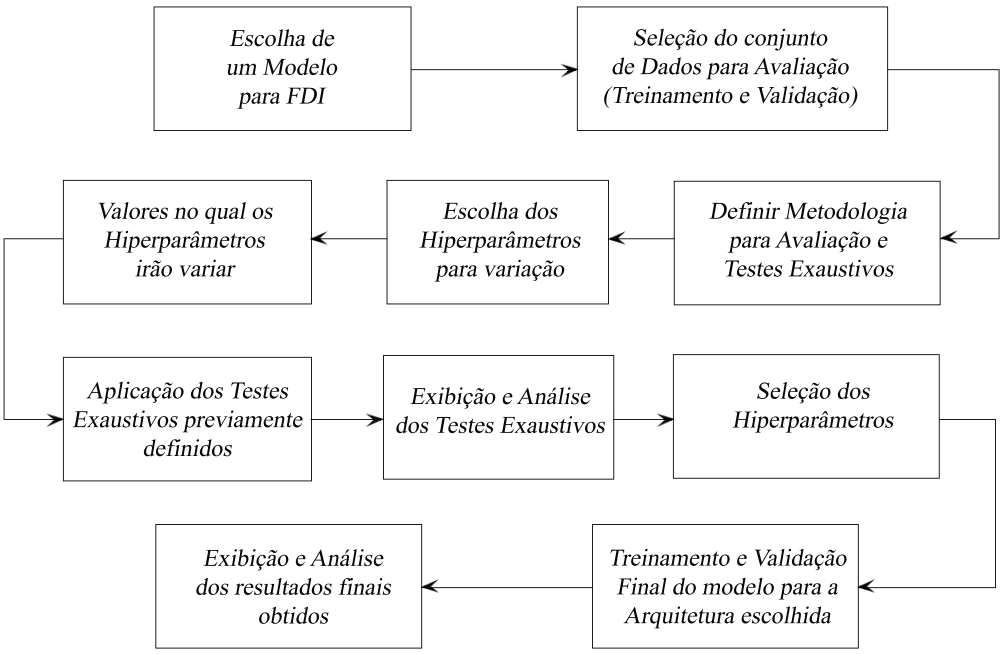


Figura 4.2: Fluxograma da metodologia para avaliação do modelo à variações nos seus hiperparâmetros.

a proporção entre os dados de treinamento e de validação. Na sequência, é definida a metodologia para avaliação e aplicação dos testes exaustivos a serem efetuados. O quarto passo apresentado no fluxograma é a escolha de quais hiperparâmetros serão variados, seguido de quais valores serão utilizados. Com todos estes pontos definidos, os testes exaustivos são aplicados. Obtendo os resultados dos testes, é preciso se definir como eles serão mostrados e analisados. Uma vez que a analise é feita, é preciso definir critérios para selecionar uma arquitetura resultante da combinação dos hiperparâmetros. Após a seleção da arquitetura, o teste final para modelo é feito, onde ele é treinado e validado com a arquitetura selecionada. Por fim, são feitas as observações a respeito dos resultado obtidos na validação final.

O capítulo 5 mostra os detalhes dos métodos e valores utilizados na aplicação da metodologia aqui proposta, com o objetivo de avaliar o comportamento de um modelo baseado na rede neural LSTM apresentado por Zhao et al. (2018) e explicado na seção 4.1.

4.3 Recursos Utilizados

O método para FDI utilizado foi desenvolvido na linguagem de programação *Python*, com auxílio do *framework* para aprendizado de máquina, *Pytorch*. A implementação

original da abordagem proposta por Zhao et al. (2018) está disponível de forma *online*[1] no repositório do *GitHub* do autor. Além do *Pytorch*, foram utilizadas bibliotecas como, *Matplotlib* e *Seaborn* para geração do gráficos, e *Numpy* e *Pandas* para o processamento dos dados. Para utilizar os recursos de GPU oferecidos pelo *Pytorch*, os testes exaustivos foram executados no *Google Colab*.

[1] `https://github.com/haitaozhao/LSTM_fault_detection`

Capítulo 5

Estudo de Caso

Para avaliação do método apresentado no Capítulo 4, será feito um estudo de caso que utilizará dados reais de processo de controle de nível de uma planta piloto. Em um primeiro momento serão apresentados os componentes da planta, o funcionamento do processo realizado e todas as falhas geradas. Então, será explicada a metodologia proposta para selecionar as falhas de treinamento e validação do modelo. Em seguida, será apresentado o método usado para avaliar a variação dos parâmetros do modelo, além dos resultados obtidos em cada fase dos experimentos.

5.1 Planta Piloto

Para o estudo de caso foi utilizada uma planta piloto para controle de nível desenvolvida pela De Lorenzo (Marins 2009). Na planta é possível realizar estratégias de controle de processos contínuos baseadas nas variáveis de pressão, temperatura, vazão e nível. Estas variáveis são definidas pela entrada $S = (u,t,f,y)$ (Costa 2014). A planta piloto é composta dos seguintes componentes:

- Painel com o controlador lógico programável (CLP).
- Componentes elétricos para o controle da planta.
- Reservatório pressurizado feito de acrílico T_1.
- Reservatório pressurizado feito de aço inoxidável T_2.
- Bomba de recirculação centrífuga.
- Inversor de frequência, responsável por controlar a bomba de recirculação.
- Sistema de aquecimento e troca de calor.
- Válvula direcional V_1.
- Válvula direcional V_2.
- Sensor de temperatura.
- Sensor de vazão.
- Sensor de pressão.
- Sensor de nível.

Além desses componentes, a planta piloto também inclui indicadores que convertem o sinal físico em elétrico, no qual vem a ser processados pelo CLP. A planta também conta com um barramento de comunicação terminal, que disponibiliza todos os sinais elétricos

para o controlador externo e um *software* supervisório e de aquisição de dados (SCADA), que é responsável pela configuração dos parâmetros do controlador e visualização das variáveis do processo (Costa 2014). A Figura 5.1 mostra a uma foto da planta piloto utilizada.

Figura 5.1: Foto da planta piloto utilizada.

A partir da Figura 5.1 é possível identificar o posicionamento dos dois tanques e da bomba de recirculação centrífuga da planta. Os tanques T_1 e T_2 são conectados por uma sistema de tubulação, que torna possível o fluxo do fluído (água) entre eles. O tanque T_1 está posicionado acima de T_2 em referência ao solo, permitindo a transferência por gravidade do líquido de T_1 para T_2. Para o fluxo do líquido do tanque T_2 para o T_1, é necessário haver geração de pressão a partir da bomba centrífuga. A Figura 5.2 apresenta o diagrama esquemático da planta piloto (Costa 2014).

Pelo esquemático apresentado na Figura 5.2 se tem de forma representativa como funciona o fluxo do fluído através das válvulas e tanques. Fazendo uso desse processo e de uma aplicação para o controle do nível da planta, 23 falhas diferentes foram geradas. Algumas dessas falhas foram geradas fisicamente, como a inserção temporária de um vazamento em um dos tanques. Outras falhas foram inseridas por *software*, como é o caso da variação do valor do *off-set* de um controlador (Bezerra et al. 2016). Essas falhas foram divididas em grupos de acordo com a sua localização no processo e de como foram concebidas, sendo estas: Atuador, Sensor, Estrutural e Pertubação. Esses grupos estão descritos a seguir:

- **Falhas de Atuador e de Sensor**: São as falhas geradas a partir de *software*, através da aplicação de valores diferentes nos *off-sets* da bomba centrífuga e no sensor de nível.
- **Falhas Estruturais**: São as falhas geradas na estrutura da planta, que podem ser geradas tanto de forma física como por *software*. Essas falhas representam problemas nos equipamentos, como válvulas e tanques.

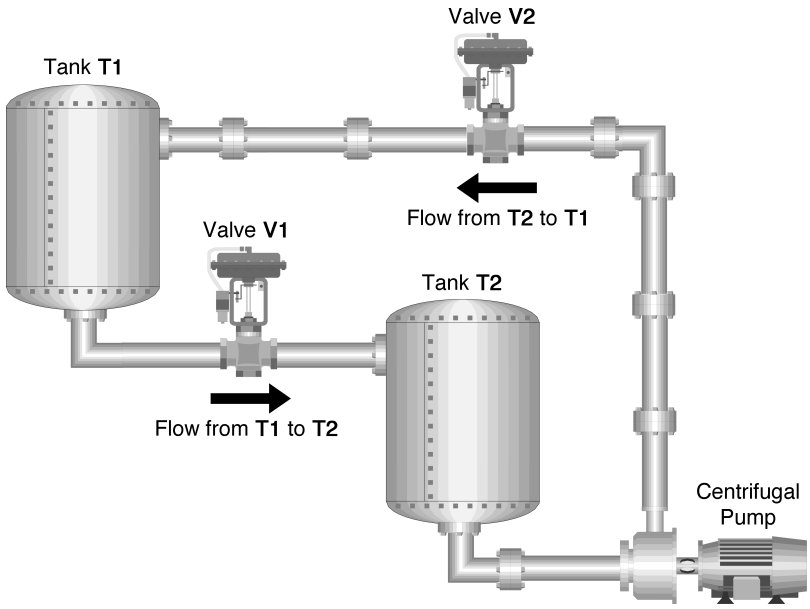


Figura 5.2: Diagrama esquemático da planta piloto.

- **Falhas de Pertubação**: Esse tipo de falha representa uma mudança não esperada na saída da planta. Ela pode ser gerada por uma adição de quantidades diferentes de fluído no tanque T_1 no meio da operação. A rigor elas são distúrbios no processo.

Os dados deste experimento foram coletados a partir do estado normal de operação da planta, seguido pela respectiva falha e novamente o estado normal. Para a aquisição desses dados foi utilizada uma amostragem de 100 milissegundos. Então foram gerados arquivos contendo dados para cada cenário com cada uma das falhas geradas (Bezerra et al. 2016). A Tabela 5.1 mostra todas as falhas geradas na planta.

Na Tabela 5.1 existem três colunas, onde a primeira de nome Falha, apresenta todos os identificadores das falhas aplicadas no processo. A segunda coluna separa as falhas nos grupos anteriormente citados. A terceira coluna exibe a informação sobre qual tipo de distúrbio foi responsável por causar a respectiva falha. A Figura 5.3 exibe os conjuntos de amostras referente a reação do processo a cada uma das falhas listadas na Tabela 5.1.

O gráfico da influência de cada falha exibido na Figura 5.3 possui uma variável observada (em laranja), o *set-point* (em azul) e a pressão da bomba (em verde) (Costa 2014). A região em vermelho delimita o intervalo de amostras em que a falha permaneceu atuante sobre o comportamento da dinâmica do processo. A Tabela 5.2 mostra os índices das amostras em que as falhas iniciaram, finalizaram e suas durações dadas em quantidade de amostras.

Na Tabela 5.2, a primeira coluna representa a falha observada, a segunda coluna exibe a amostra na qual o distúrbio começou a atuar sobre o processo, seguida pela terceira e quarta colunas, que mostram a amostra do fim da falha e sua duração em amostras, respectivamente. Ao se avaliar a Tabela 5.2, é notável a diferença entre a quantidade de amostras em que cada um das falhas permanecem afetando o comportamento do sistema.

Tabela 5.1: Conjunto de falhas geradas.

Falha	Grupo	Descrição
F1	Atuador	+ 2% de *off-set*
F2		+ 4% de *off-set*
F3		+ 8% de *off-set*
F4		- 2% de *off-set*
F5		- 4% de *off-set*
F6		- 8% de *off-set*
F7	Sensor	+ 2% de *off-set*
F8		+ 4% de *off-set*
F9		+ 8% de *off-set*
F10		- 2% de *off-set*
F11		- 4% de *off-set*
F12		- 8% de *off-set*
F13	Estrutural	100% de vazamento do tanque
F14		66% de vazamento do tanque
F15		30% da válvula presa V1
F16		50% da válvula presa V1
F17		85% da válvula presa V1
F18		100% da válvula presa V1
F19		25% da válvula presa V2
F20		50% da válvula presa V2
F21		75% da válvula presa V2
F22	Perturbação	Baixa perturbação
F23		Alta pertubação

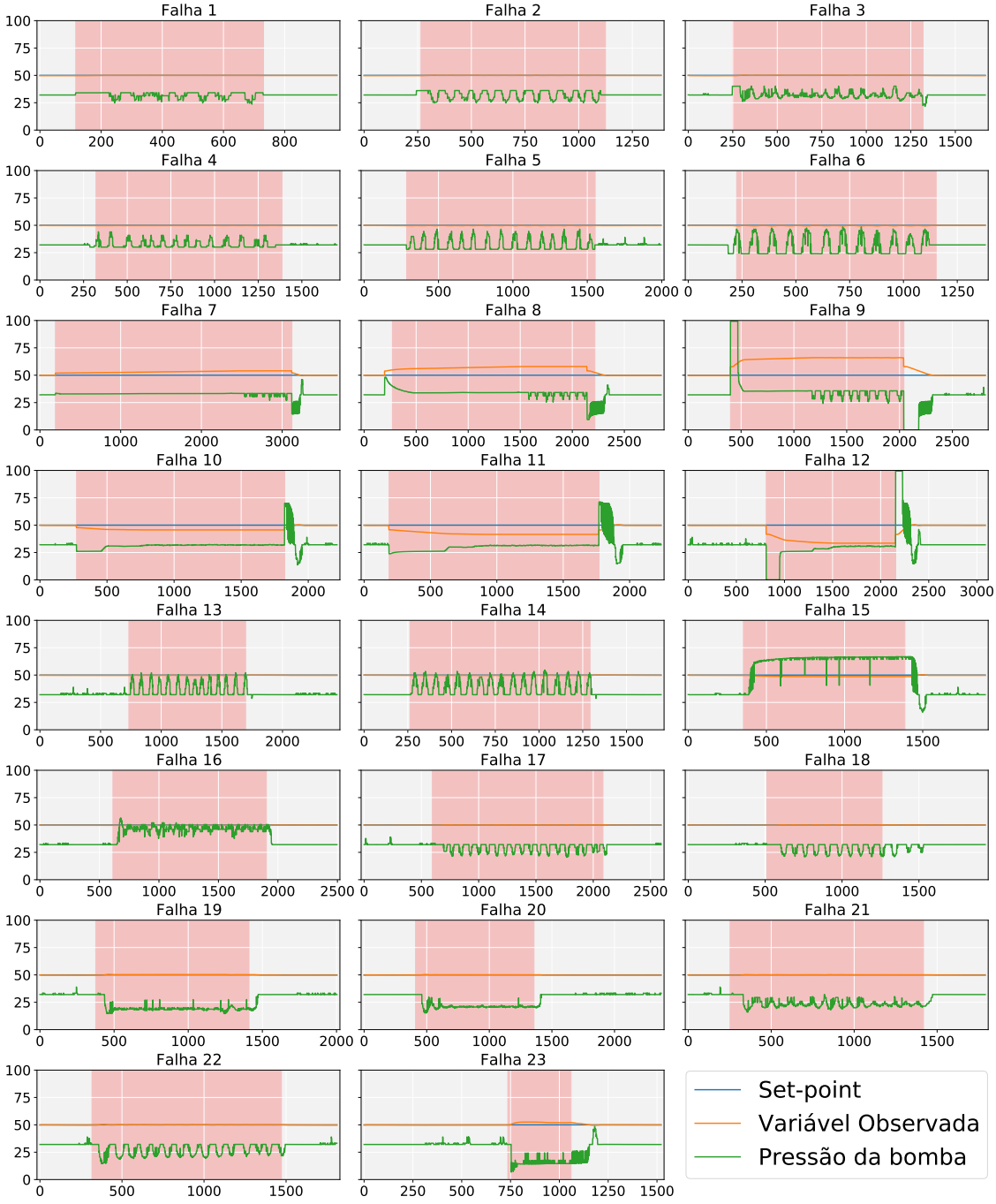


Figura 5.3: Exibição da influência das 23 falhas sobre uma variável observada na planta piloto.

Tabela 5.2: Conjunto de falhas geradas.

Falha	Início	Fim	Duração (amostras)
F01	117	731	614
F02	265	1125	860
F03	248	1319	1071
F04	321	1386	1065
F05	286	1553	1267
F06	225	1152	927
F07	192	3121	2929
F08	272	2217	1945
F09	400	2041	1641
F10	273	1825	1552
F11	188	1770	1582
F12	811	2157	1346
F13	733	1696	963
F14	262	1292	1030
F15	353	1386	1033
F16	613	1903	1290
F17	596	2085	1489
F18	511	1260	749
F19	376	1410	1034
F20	411	1357	946
F21	252	1418	1166
F22	317	1472	1155
F23	736	1060	324

Tendo isto em mente e visando obter um conjunto de dados mais balanceado, foi escolhida uma quantidade de amostras fixa para servir de entrada para o modelo. Devido à baixa quantidade de amostras disponíveis para a falha $F23$, o grupo Pertubação que também inclui a falha $F22$ (vide Tabela 5.1) não foi incluído nos experimentos. Assim, com o conjunto de falhas que vão da falha $F01$ até $F21$, o valor fixo de amostras escolhido foi a menor centena mais próxima da quantidade de amostras mais baixas disponível entre estas falhas. De acordo com a Tabela 5.2, a falha $F01$ possui o menor número de amostras em distúrbio, 614 amostras, assim, o valor fixo de amostras escolhido foi 600. A contagem de amostras se inicia a partir da primeira amostra em que a falha é atuante no sistema, que é referente à coluna Início da Tabela 5.2. Essa contagem segue de forma sequencial no conjunto de dados, finalizando 599 amostras depois, totalizando 600 amostas onde o distúrbio ainda está ativo. Pensando na relevância das informações que o conjunto de dados pode fornecer para o modelo, a variável que representa o *set-point* não foi utilizada nos experimentos. O valor constante e igual desta variável em todos os distúrbios não agrega nenhuma informação que possa ser relevante na classificação das falhas. A Figura 5.4 mostra o conjunto de amostras da falha $F01$ que foi utilizado nos experimentos.

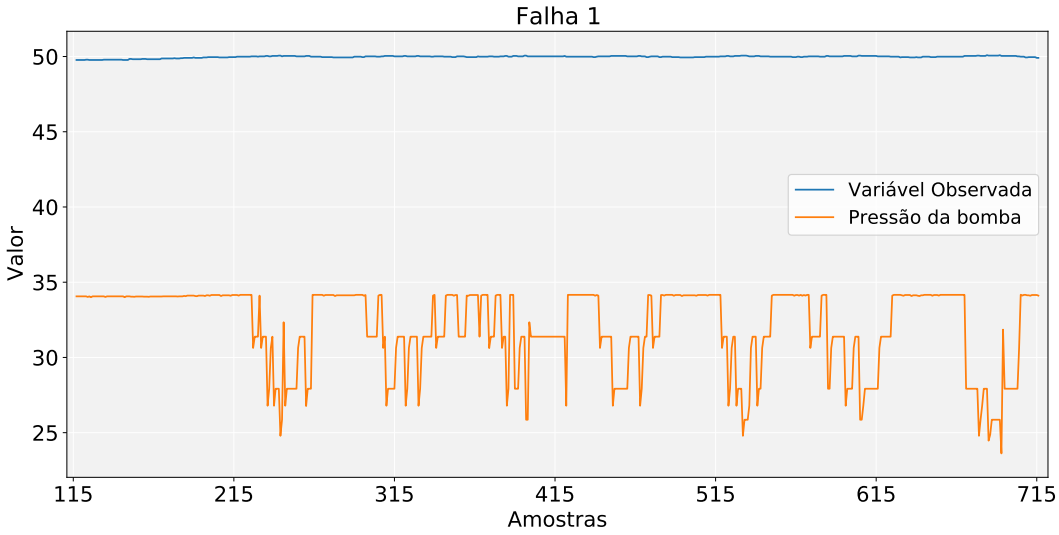


Figura 5.4: Influência da falha $F01$ sobre a dinâmica de uma variável do processo da planta piloto.

Na Figura 5.4 é possível visualizar os comportamentos das variável da pressão da bomba e da variável observada em reação ao acréscimo de $+2\%$ no *off-set* devido à aplicação da falha $F01$.

5.2 Seleção de Falhas

Um dos aspectos relevantes a ser realizado nesta dissertação é avaliar a capacidade do modelo baseado em redes neurais do tipo LSMT em classificar diferentes falhas que são

geradas de forma semelhante. Isso significa treinar a rede com uma falha F_a e coletar o desempenho em classificar a falha F_b, onde ambas foram geradas de formas semelhantes. Observando a Tabela 5.1, a coluna Descrição referente ao grupo Atuador exibe seis diferentes porcentagens de *off-sets*. Dessas seis porcentagens, três são positivas e três são negativas. Dentre as três positivas, duas serão selecionadas para os experimentos, onde uma fará parte do conjunto de falhas para o treinamento do modelo e outra irá compor o conjunto de validação. Esta seleção de duas falhas será utilizada para todas as variações das causas das falhas dentro dos seus grupos.

Para servir de critério na seleção das falhas, foi escolhida a média das distâncias euclidianas entre as variáveis presentes em um par de falhas. A equação (5.1) define a distância euclidiana entre as variáveis de duas falhas, na qual $v^i_{F_a}$ e $v^i_{F_b}$ são a mesma variável para as distintas falhas (*falha a* e *falha b*). Quanto maior o valor dessa média de distâncias euclidianas, mais as falhas são dissimilares.

$$D(v^i_{F_a}, v^i_{F_b}) = \sqrt{\Sigma_{i=1}^{N} \left(v^i_{F_a} - v^i_{F_b} \right)^2} \tag{5.1}$$

No caso, N representa a quantidade de amostras em cada variável e i é um índice que identifica cada uma dessas amostras. Após se obter a distância euclidiana entre todas as variáveis presentes nas falhas, uma média é calculada e se obtém o índice de dissimilaridade entre duas falhas. A Tabela 5.3 mostra o resultado do cálculo desse índice para as possíveis combinações de falhas dentro dos grupos.

Os valores da coluna Dissimilaridade demonstram o quanto as falhas da coluna Combinação são distintas. Para esta seleção, foram escolhidas as combinações de falhas que obtiveram o menor valor para a dissimilaridade. Os valores mais baixos para o índice se encontram em destaque na coluna Dissimilaridade da Tabela 5.3. De acordo com valores resultantes, as falhas selecionadas para o conjunto de treinamento do modelo foram as falhas F01, F04, F07, F10, F13, F17 e F19. Para o conjunto de validação foram selecionadas as falhas F03, F05, F08, F11, F14, F17 e F20.

5.3 Teste Exaustivo

Para observar o desempenho da metodologia para classificação de falhas a variações dos seus hiperparâmetros foram executados alguns testes exaustivos, resultando em possíveis arquiteturas para o modelo baseado em redes neurais do tipo LSTM. Para cada uma dessas combinações foram criados 100 modelos do método avaliado, onde cada um deles é treinado por 100 épocas com os conjunto de falhas selecionados para o treinamento. A medida escolhida para estimar a qualidade modelo, foi a acurácia. Ao final de cada treinamento, o modelo é executado uma única vez com as falhas selecionadas para validação. A partir dessa validação é extraído o valor de acurácia para classificação das falhas, que ao final de cada teste de arquitetura resulta em 100 valores de acurácia. A partir desses valores de acurácia é montada uma distribuição que servirá para a análise daquela arquitetura no momento da seleção dos hiperparâmetros.

No contexto apresentado neste estudo, o objetivo do teste exaustivo é diminuir o fator

Tabela 5.3: Seleção das Falhas.

Sub Grupo	Combinação	Dissimilaridade
Atuador (+2%,+4%,+8%)	F01 ↔ F02	61,30
	F01 ↔ F03	**46,45**
	F02 ↔ F03	51,89
Atuador (-2%,-4%,-8%)	F04 ↔ F05	**82,57**
	F04 ↔ F06	114,28
	F05 ↔ F06	156,03
Sensor (+2%,+4%,+8%)	F07 ↔ F08	**75,75**
	F07 ↔ F09	427,42
	F08 ↔ F08	356,74
Sensor (-2%,-4%,-8%)	F10 ↔ F11	**72,98**
	F10 ↔ F12	282,43
	F11 ↔ F12	236,93
Estrutural Tanque (100%,66%)	F13 ↔ F14	**122,48**
Estrutural V1 (30%, 50%, 84%, 100%)	F15 ↔ F16	226,43
	F15 ↔ F17	428,27
	F15 ↔ F18	436,50
	F16 ↔ F17	212,47
	F16 ↔ F18	219,92
	F17 ↔ F18	**50,41**
Estrutural V2 (25%, 50%, 75%)	F19 ↔ F20	**41,15**
	F19 ↔ F21	75,15
	F20 ↔ F21	83,50

da aleatoriedade ao gerar um modelo com determinada combinação de hiperparâmetros, além de analisar a influência e sensibilidade de cada hiperparâmetro no desempenho do modelo resultante. Por meio das medidas de acurácias obtidas no teste será possível observar de forma visual e numérica os componentes estatísticos que podem ser avaliados na escolha da arquitetura que será utilizada pelo modelo.

Os hiperparâmetros do modelo selecionados para serem variados foram o tamanho da rede, denominado de *hidden size* (*HS*), a quantidade de amostras em cada entrada, denominado de *J*, o *overlap* de amostras entre cada uma dessas entradas e a quantidade de camadas. De modo mais específico, cada um desses hiperparâmetros varia da seguinte forma:

- ***Hidden Size***: É o tamanho da camada da rede LSTM, onde seu valor irá variar em 20, 30, 40, 50, 60 neurônios.
- **Entrada** *J*: Se trata da quantidade de amostras sequenciais que irá compor uma entrada da rede. As variações das amostras serão 3, 5, 7 e 10. Essa entrada é composta pela amostra atual do processo e por $J-1$ amostras passadas.
- ***Overlap***: É a quantidade de amostras que se sobrepõem entre uma entrada e a entrada seguinte. O *overlap* varia do mínimo 0, até máximo $J-1$. Para uma janela de tamanho $J=5$, enquanto que o *overlap* varia de 0 até 4. Isso significa que para o *overlap* mínimo (zero), as entradas não compartilham nenhuma amostra. Para o caso máximo, a diferença entre uma entrada e outra é apenas de uma amostra do instante de tempo atual do processo.
- **Camadas**: Número de camadas LSTM para a rede. O máximo de duas camadas foi avaliado, com todas as combinações possíveis entres os *hidden sizes* anteriormente citados.

Devido à quantidade de amostras coletadas das variáveis do processo sob cada falha ser limitada em 600 amostras, ao se variar o *overlap* para valores menores do que o máximo $J-1$, se tem uma redução na quantidade de entradas possíveis para a rede LSTM. A equação (5.2) mostra como é feito o cálculo para a quantidade de entradas resultantes de acordo com a variação do tamanho da entrada *J* e do *overlap*.

$$Q_e = \left\lfloor \left(\frac{600 - J}{J - overlap} \right) + 1 \right\rfloor \tag{5.2}$$

Sendo que Q_e é a quantidade de entradas resultante e os símbolos $\lfloor \ \rfloor$ representam a função *floor* (pise), que arredonda o valor para o menor inteiro mais próximo.

5.4 Resultados dos Testes Exaustivos

Com a definição dos hiperparâmetros a serem variados, os testes foram executados seguindo os valores descritos na seção passada e *batchs* de tamanho 60. O artifício gráfico utilizado para representar a variação dos valores das acurácias obtidas no testes exaustivo, foi o *box-plot*. A Figura 5.5 mostra os *box-plots* das medidas de acurácias obtidas em relação à variação da entrada *J*. Nestes resultados estão unidas todas as acurácias para os

cinco tamanhos de rede HS com uma única camada. O objetivo do resultado ser descrito em *box-plots* é avaliar o comportamento médio dos valores de acurácia obtidos, levando em conta unicamente a alteração do tamanho da entrada da rede LSTM. Na Figura 5.5 foi inserida uma reta de cor verde no valor de acurácia $0,95$, que representa aqui, uma âncora visual para dar uma noção de escala em relação aos valores observados nos testes.

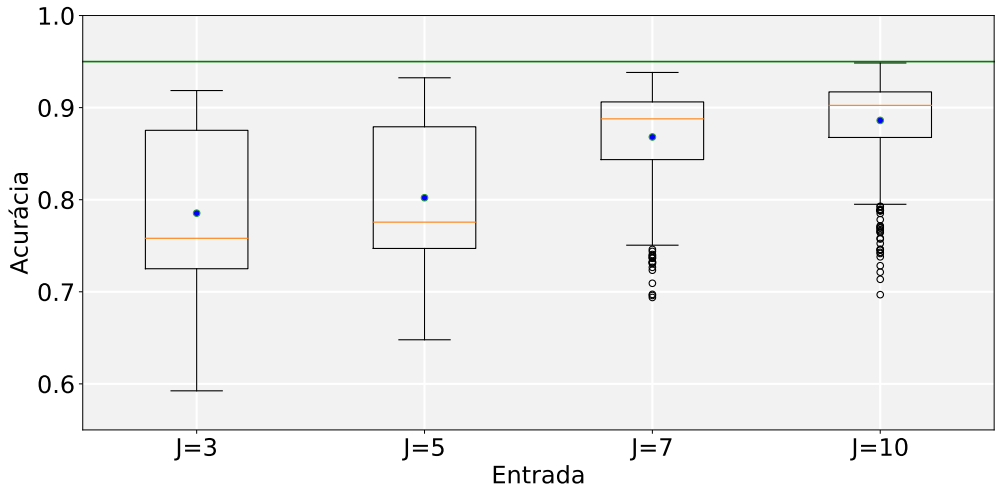


Figura 5.5: Valores das acurácias para a variação do tamanho da entrada J. O gráfico *box-plot* possui as seguintes componentes: i) retas horizontais nas extremidades são os valores máximos e mínimos respectivamente; ii) A linha superior do *box* delimita o terceiro quartil e a inferior o primeiro, compondo a amplitude interquartílica da distribuição; iii) A reta horizontal dento da caixa é a mediana; iv) O ponto preenchido contém a informação da média; v) pontos vazados representam os valores *outliers*.

Baseando-se nos resultados expostos na Figura 5.5, observa-se que os resultados que apresentaram os melhores desempenhos foram os com entradas da rede de tamanho $J = 7$ e $J = 10$, pois esses cenários demonstram mais consistência em seus resultados, com valores de acurácia média, mediana e máxima mais altos. Outro ponto a ser observado é uma distância interquartil mais curta, indicando uma menor variância nos seus resultados, ou ainda, menos sensível a essa variação paramétrica. Além disso, essas configurações de valores de entrada da rede possuem valores *outliers* sob o *box*, indicando que baixas acurácias não são comuns, o que é uma característica desejável.

De forma similar à análise anterior, a Figura 5.6 exibe os gráficos em *box-plot* dos resultados obtidos para a variação do tamanho da rede LSTM (HS). Neste caso, para cada valor de HS as distribuição agrupam as acurácias coletadas para todos os valores da entrada J. Para a variação do hiperparâmetro *hidden size*, o cenário com $HS = 20$ apresentou uma distribuição com aspectos mais favoráveis, de acordo com a média, mediana, variância, máximo e *outliers* observados. De modo geral, se espera que redes maiores produzam melhores resultados, o que não foi o caso para este experimento. Os modelos com redes de tamanho $HS = 20$ apresentaram uma distribuição com valores de acurácia mais próximos e mais elevados.

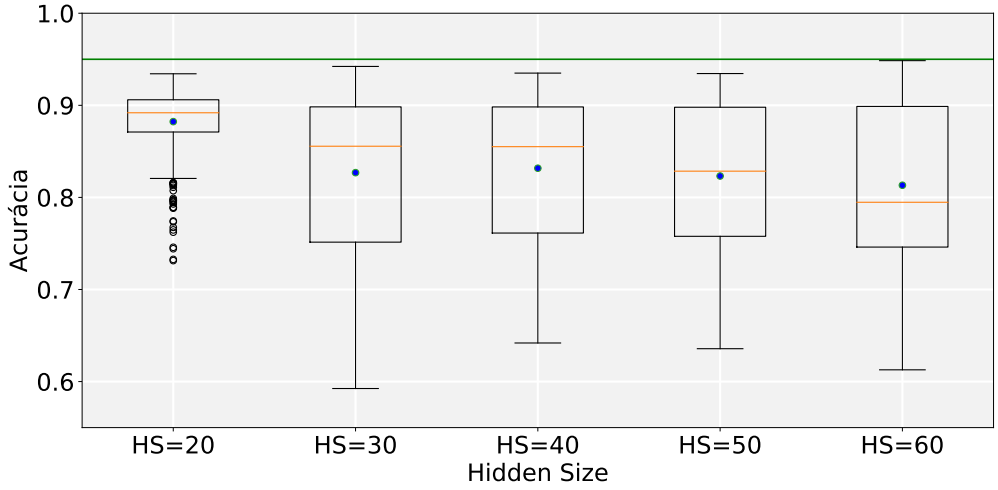


Figura 5.6: Resultados obtidos de acurácias para a variação do *hidden size*.

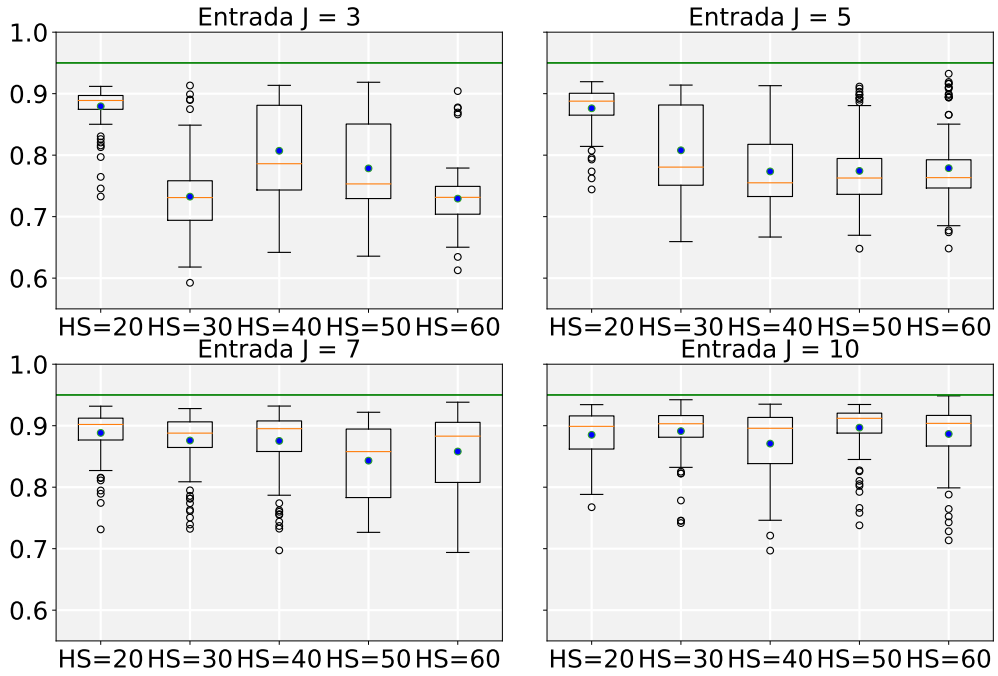


Figura 5.7: Resultados obtidos de acurácias para as combinações de HS e J.

A Figura 5.7 mostra os resultados de acurácias obtidos para cada combinação dos hiperparâmetros J e HS de forma isolada. De forma similar ao observado nos testes anteriores, a partir da Figura 5.7 é notado que as distribuições com melhores comportamentos acontecem quando $HS = 20$ e as entradas têm tamanho $J = 7$ e $J = 10$. Em destaque, está a arquitetura onde o $HS = 20$ está combinado com $J = 3$, na qual apresenta uma variância visivelmente menor que as demais. Observando as outras características dos resultados dispostos em *box-plots*, quando o $HS = 50$ é combinado com o $J = 10$ os valores para média e mediana são os mais elevados. Os resultados obtidos para os valores de $J = 3$ e $J = 5$, de modo geral, apresentaram um comportamento oposto ao considerado adequado, pois produziram uma maior variância, baixos valores para média e mediana, além de *outliers* acima do *box*, demonstrando que valores altos de acurácia não são comuns. Apenas para o valor de $HS = 20$, os resultados apresentaram um comportamento desejado quando a entrada tem o tamanho $J = 3$ e $J = 5$, como já discutido.

As Figuras 5.8, 5.9, 5.10, 5.11 e 5.12 mostram os resultados obtidos a partir da variação das combinações dos tamanhos de HS em redes com duas camadas. Para esses casos, cada uma das figuras representa um valor de HS para a primeira camada e as possíveis variações para o valor da entrada J. A Figura 5.8 mostra resultados obtidos com o $HS = 20$ na primeira camada do modelo e todos os demais valores de HS para a segunda camada.

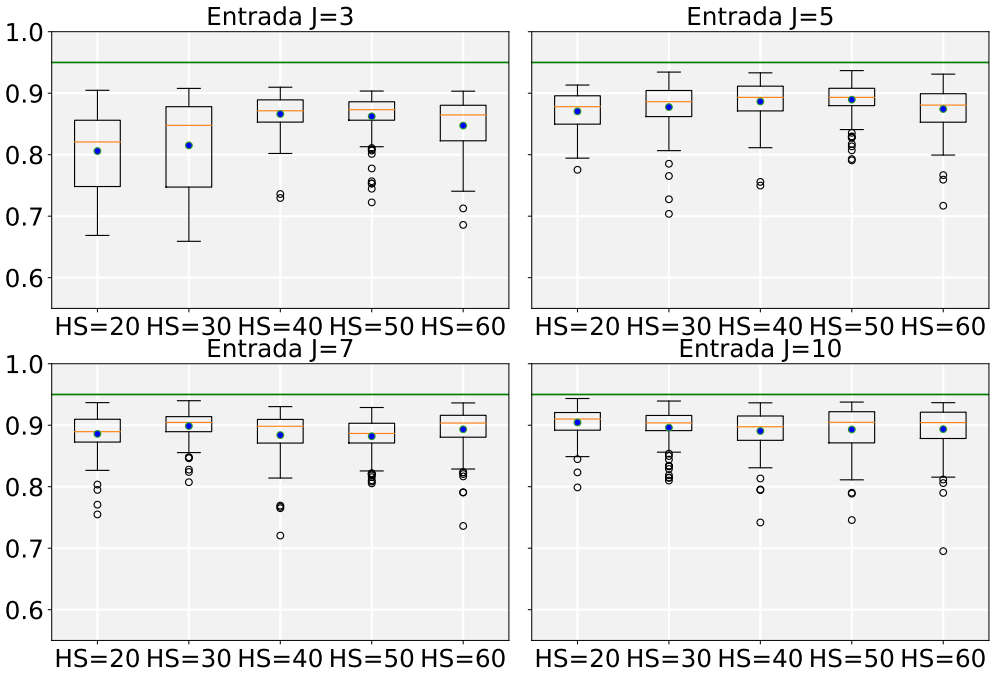


Figura 5.8: Resultados obtidos para a primeira camada $HS = 20$ e todos os tamanhos de entrada J.

Os destaques da Figura 5.8 são os resultados obtidos com os valores de $J = 7$ e $J = 10$, onde se apresentou valores de máxima próximos a referência de $0,95$. Com exceção dos resultados obtidos a partir dos valores de $HS = 50$ e $HS = 60$, a entrada de tamanho $J = 5$ demonstrou um comportamento um pouco abaixo da média. De modo geral, os resultados

obtidos a partir de $J = 3$ tiveram os piores comportamentos observados. A Figura 5.9 mostra os resultados obtidos referentes às combinações possíveis para a primeira camada da rede com $HS = 30$.

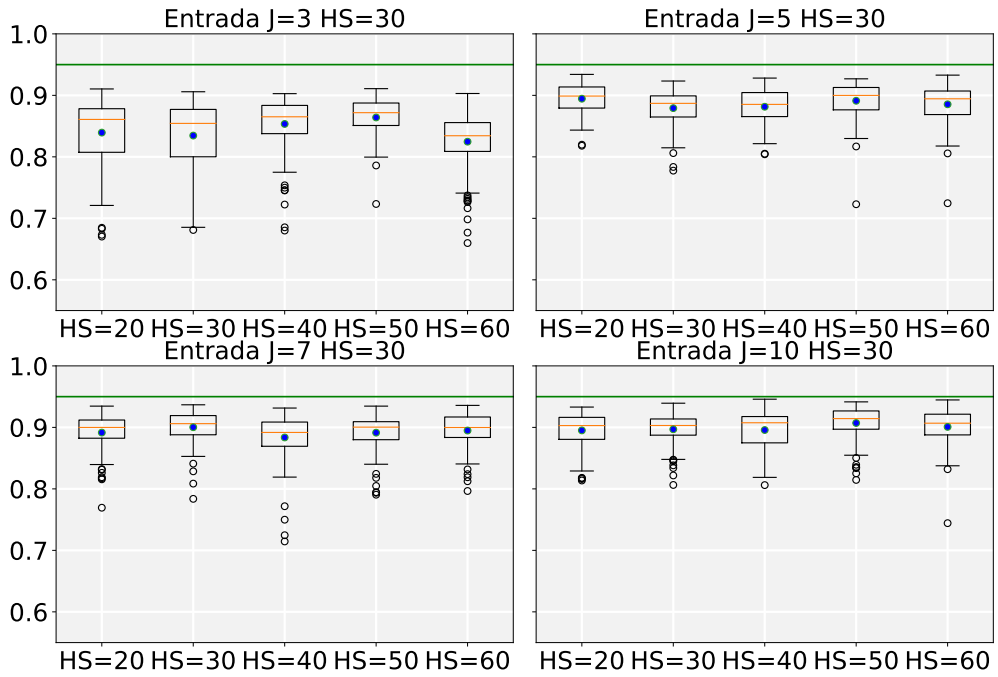


Figura 5.9: Resultados obtidos para a primeira camada $HS = 30$ e todos os tamanhos de entrada J.

Os resultados obtidos para as combinações de valores de hiperparâmetros mostrados na Figura 5.9 possuem um comportamento semelhante aos exibidos na Figura 5.8, onde os resultados obtidos para os valores de $J = 7$ e $J = 10$ apresentaram os seus melhores resultados próximos à referência de $0,95$, como também baixas variâncias. Porém, é possível notar uma leve melhora no comportamento nos resultados a partir de $J = 5$. As Figuras 5.10, 5.11 e 5.12 mostram o comportamento estatístico dos resultados com valores de $HS = 40$, $HS = 50$ e $HS = 60$ na primeira camada da rede.

Nas Figuras 5.10, 5.11 e 5.12 pode-se observar que a medida em que o valor de HS para a primeira camada aumenta há uma diminuição nas variâncias e aumento da acurácia para os valores de J mais altos. Também é notável uma redução nas variâncias das acurácias a partir do $J = 3$. Esses comportamentos levam a acreditar que o aumento da quantidade de neurônios na primeira camada da rede pode trazer mais estabilidade para os valores de acurácia resultantes das arquiteturas que possuem um tamanho de entrada J menor.

Após a avaliação do comportamento do modelo para a variação das camadas da rede LSTM, resta observar o comportamento das distribuições obtidas a partir da variação do *overlap*. Esta variação foi feita apenas para as redes com uma única camada. A medida que o valor do *overlap* diminui, a quantidade de amostras para treinamento e validação dos modelos também se reduz. Para cada valor de *overlap* diferente, a quantidade de

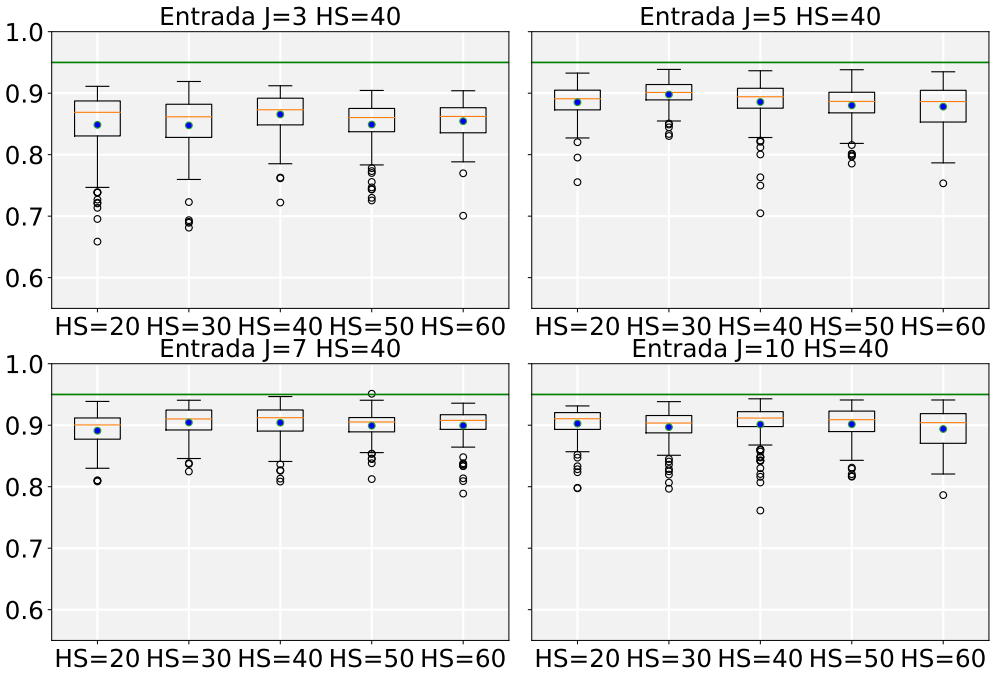


Figura 5.10: Resultados obtidos para a primeira camada $HS = 40$ e todos os tamanhos de entrada J.

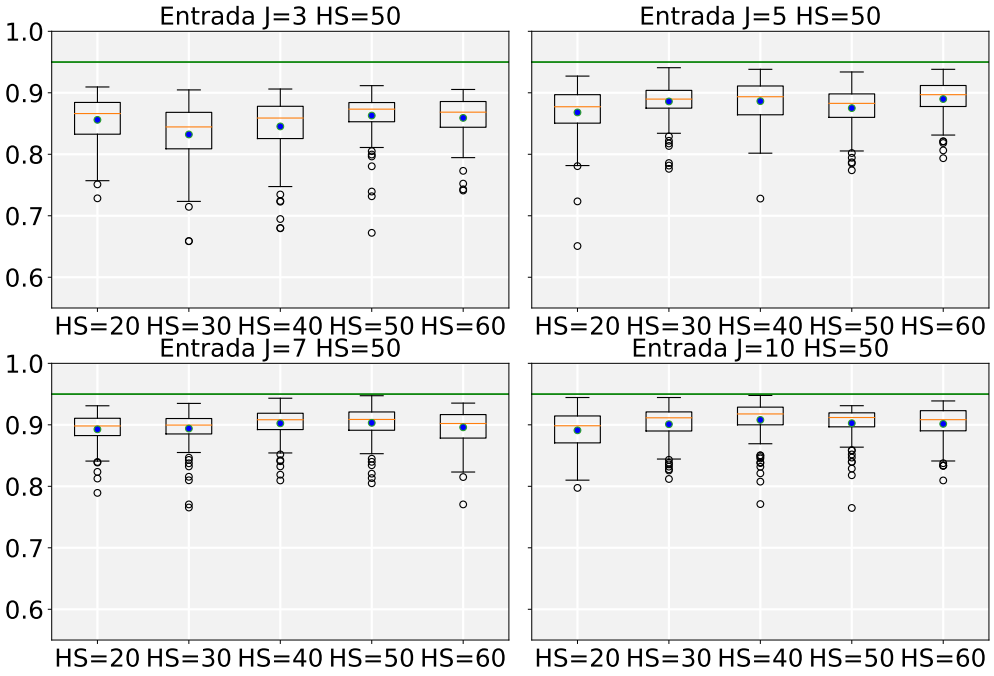


Figura 5.11: Resultados obtidos para a primeira camada $HS = 50$ e todos os tamanhos de entrada J.

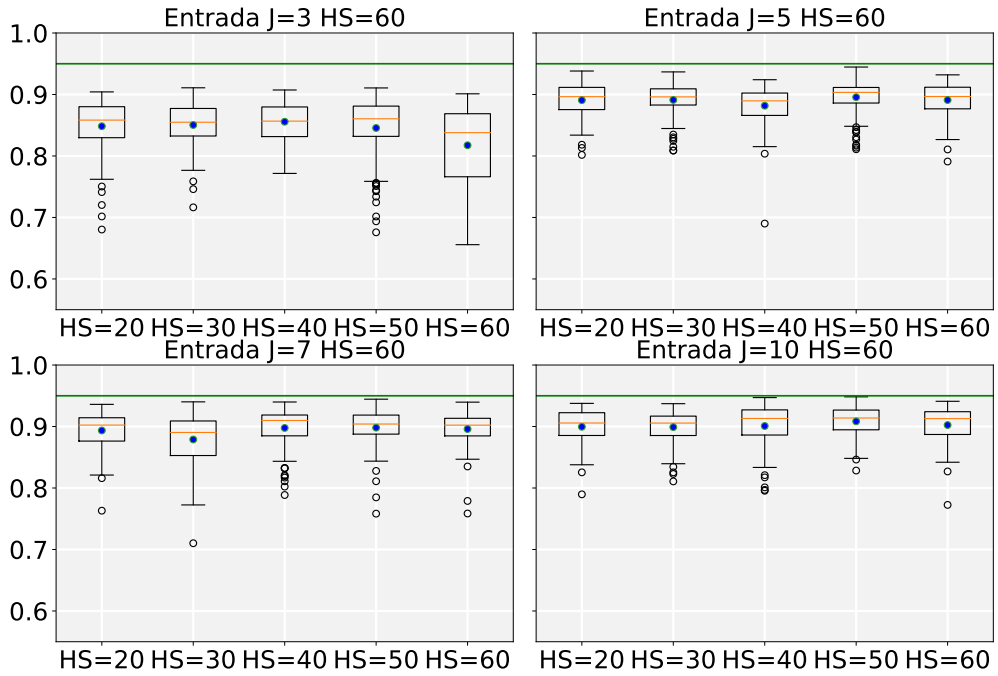


Figura 5.12: Resultados obtidos para a primeira camada $HS = 60$ e todos os tamanhos de entrada J.

amostras obedece à equação (5.2). Nos testes mostrados até agora, o valor de *overlap* foi de $J-1$. Para as entradas de tamanho $J = 3$, $J = 5$, $J = 7$ e $J = 10$, a quantidade de entradas para o modelo se tornou em 598, 596, 594 e 591, respectivamente. Como já mencionado, neste experimento o valor d *overlap* irá variar de 0 até o valor de $J-1$ o que faz a quantidade de entradas possíveis diminuir ainda mais, chegando ao mínimo de 60 entradas quando o $J = 10$ e não há *overlap* nas amostras.

As Figuras 5.13, 5.14, 5.15, 5.16 e 5.17 exibem os resultados obtidos para a influência do hiperparâmetro *overlap* sobre a acurácia do modelo a partir da variação do tamanho da rede. A Figura 5.13 mostra os resultados referentes a $HS = 20$, com as variações do *overlap* para cada um dos tamanhos da entrada J.

O destaque da Figura 5.13 são os resultados obtidos para o caso onde não há *overlap*, nos quais se mostram comportamentos tão bons ou até melhores que os obtidos com valores de *overlap* elevados. A Figura 5.14 mostra os resultados obtidos com a variação dos valores de *overlap* para o tamanho de rede $HS = 30$.

A partir da Figura 5.14 é possível observar que o comportamento para $HS = 30$ não segue o mesmo padrão do que para o $HS = 20$. Analisando-se esses resultados não se nota um comportamento tão melhor para os valores de *overlaps* menores que o máximo $J-1$, exceto para entrada de tamanho $J = 10$ e em uma ou caso para as entradas $J = 5$ e $J = 5$, o que não demonstra uma tendência. A partir da Figura 5.15 é possível ver o comportamento dos resultados obtidos com a variação do valor do hiperparâmetro *overlap* para o valor de $HS = 40$ na primeira camada.

Uma particularidade observada nos resultados exibidos na Figura 5.15 é a quantidade

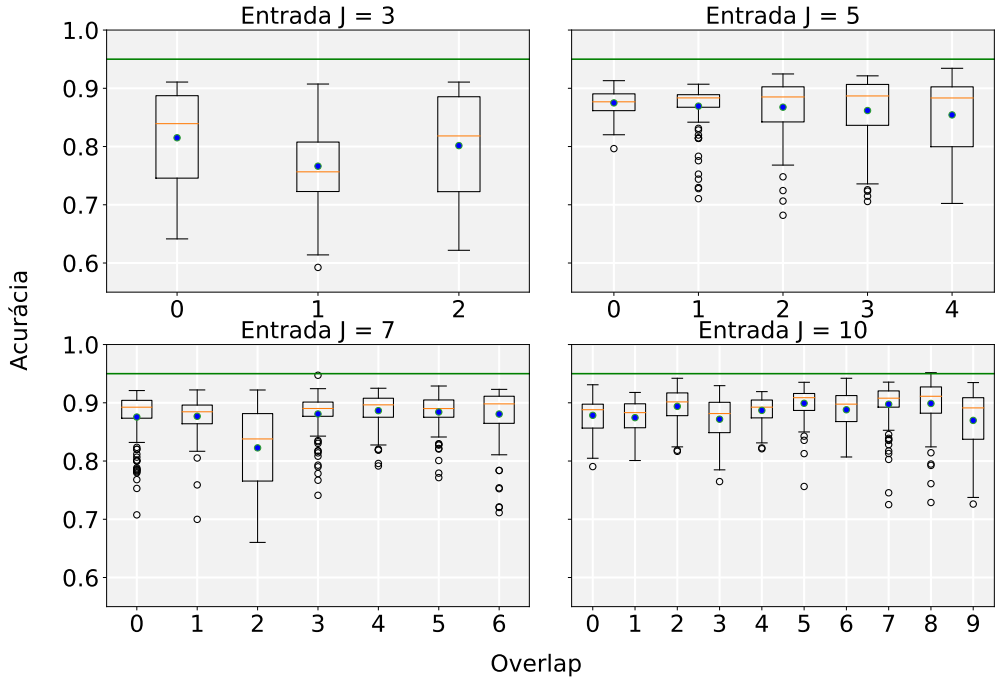


Figura 5.13: Resultados obtidos de acurácia do modelo para variação do *overlap* com $HS = 20$.

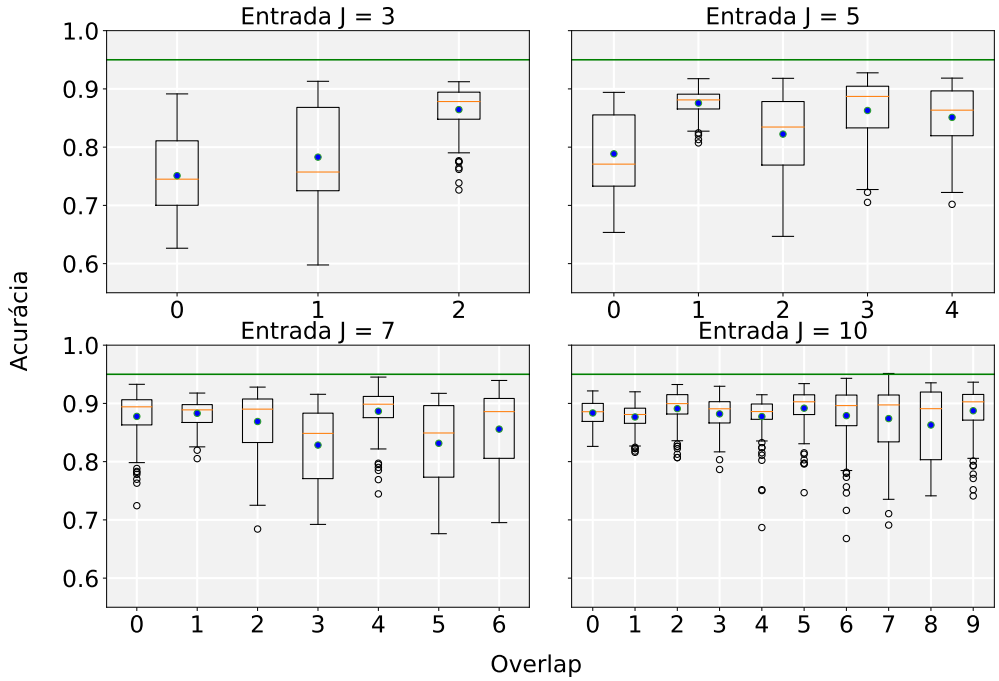


Figura 5.14: Resultados obtidos de acurácia do modelo para variação do *overlap* com $HS = 30$.

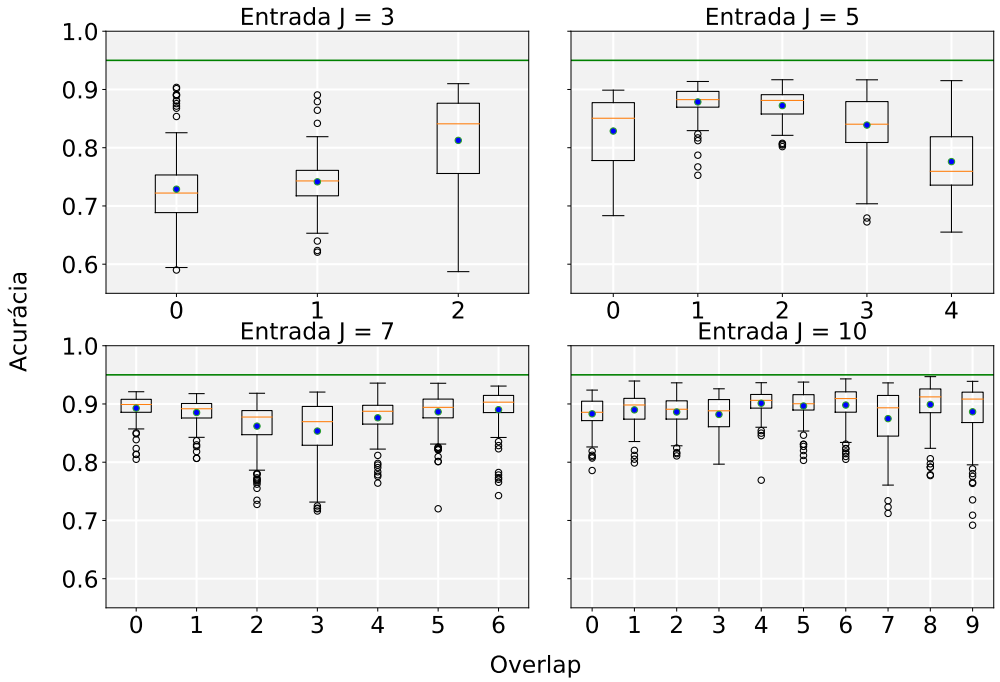


Figura 5.15: Resultados obtidos de acurácia do modelo para variação do valor do hiperparâmetro *overlap* com $HS = 40$.

de *outliers* acima do *box* para valores de entrada da rede $J = 3$. Este tipo de comportamento não havia sido notado para outros resultados de variação do valor do *overlap*. No restante dos resultados obtidos para as demais combinações deste valor de HS, não se obteve nenhum comportamento consideravelmente diferente dos já observados. As Figuras 5.16 e 5.17 mostram os resultados obtidos para $HS = 50$ e $HS = 60$.

Para as Figuras 5.16 e 5.17 não se obteve nenhum tipo de comportamento diferente dos anteriormente observados. De modo geral, os resultados obtidos para os $J = 7$ e $J = 10$ continuaram demonstrando melhor comportamento das acurácias de validação colhidas, com média, mediana, variância e máximos se mantendo de forma regular.

5.5 Seleção dos Hiperparâmetros

Com os resultados dos testes exaustivos, uma seleção para a combinação dos hiperparâmetros pode ser realizada. Devido à alta quantidade de combinações nas possíveis configurações na arquitetura da rede obtidas a partir dos testes, é necessário realizar uma seleção dos melhores valores para os hiperparâmetros e então se avaliar de forma numérica as características dessas configurações paramétricas da rede. Ao analisar visualmente os resultados apresentados na seção anterior, de modo geral é possível observar um melhor comportamento dos mesmos com as entradas $J = 7$ e $J = 10$. Então, apenas esses dois valores de entrada serão analisados em mais detalhes. Um dos critérios estabelecidos como desejável da arquitetura da rede neural LSTM para classificação é um menor número de computações aliada a uma boa acurácia. Assim, as redes que possuem uma soma

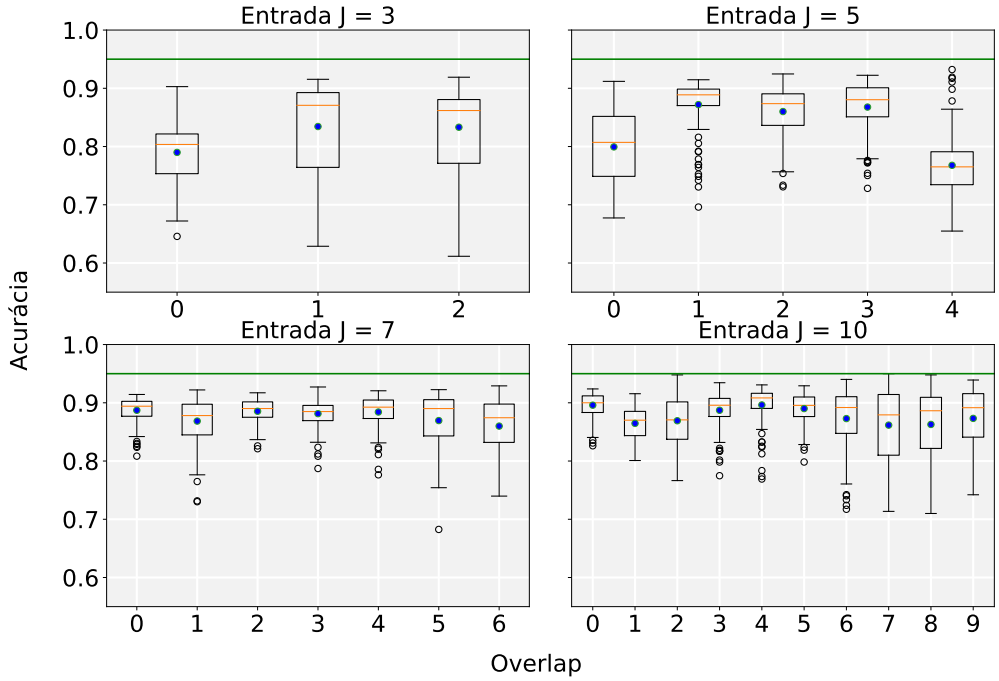


Figura 5.16: Resultados obtidos de acurácia do modelo para variação do valor do hiperparâmetro *overlap* com $HS = 50$.

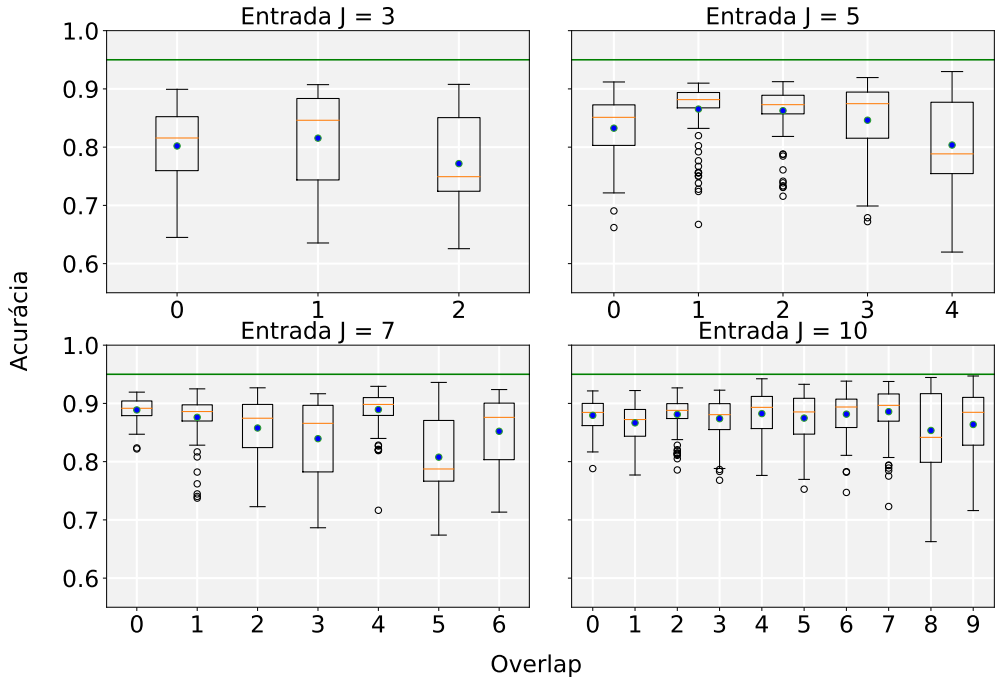


Figura 5.17: Resultados obtidos de acurácia do modelo para variação do valor do hiperparâmetro *overlap* com $HS = 60$.

total de neurônios acima de 60 (sessenta) não participarão da avaliação dessa avaliação mais detalhada. Em relação ao valor de *overlap*, ele não representa custo computacional de fato, por ser apenas uma forma de variação dos dados de entrada, então a escolha deste parâmetro será feita após a seleção dos demais hiperparâmetros. A Tabela 5.4 traz as arquiteturas que obtiveram os melhores valores de média, mediana, variância e máximo de acurácia para esses critérios.

Tabela 5.4: Arquiteturas selecionadas

Arquitetura	*HS* C1	*HS* C2	Ent. *J*	Média	Mediana	Máximo	Variância
#01	20	-	7	0,888	0,902	0,932	$1,33 \times 10^{-3}$
#02	30	-	7	0,876	0.888	0,928	$1,94 \times 10^{-3}$
#03	40	-	7	0,875	0.895	0,932	$2,55 \times 10^{-3}$
#04	50	-	7	0,843	0.858	0,922	$3,16 \times 10^{-3}$
#05	60	-	7	0,858	0,883	0,938	$3,71 \times 10^{-3}$
#06	20	-	10	0,885	0,899	0,934	$1,54 \times 10^{-3}$
#07	30	-	10	0,891	0,903	0,942	$1,65 \times 10^{-3}$
#08	40	-	10	0,871	0,896	0,935	$3,08 \times 10^{-3}$
#09	50	-	10	0,897	**0,912**	0,934	$1,57 \times 10^{-3}$
#10	60	-	10	0,887	0,904	**0,949**	$2,37 \times 10^{-3}$
#11	20	20	7	0,886	0,889	0,937	$1,12 \times 10^{-3}$
#12	20	30	7	0,899	0,905	0,940	$\mathbf{5,44 \times 10^{-4}}$
#13	20	40	7	0,884	0,898	0,930	$1,58 \times 10^{-3}$
#14	30	20	7	0,892	0,900	0,935	$9,41 \times 10^{-4}$
#15	30	30	7	0,900	0,906	0,937	$7,15 \times 10^{-4}$
#16	40	20	7	0,891	0,900	0,939	$8,76 \times 10^{-4}$
#17	20	20	10	**0,904**	0,910	0,943	$5,89 \times 10^{-4}$
#18	20	30	10	0,896	0,904	0,939	$8,65 \times 10^{-4}$
#19	20	40	10	0,891	0,898	0,936	$1,15 \times 10^{-3}$
#20	30	20	10	0,895	0,903	0,933	$9,03 \times 10^{-4}$
#21	30	30	10	0,897	0,903	0,939	$6,54 \times 10^{-4}$
#22	40	20	10	0,903	0,911	0,931	$7,14 \times 10^{-4}$

Na Tabela 5.4 se tem a coluna denominada Arquitetura, que atribui um número a determinada combinação dos hiperparâmetros. As colunas HS C1 e HS C2 representam o *hidden size* para as camadas um e dois da rede, respectivamente. A coluna Entrada *J* (Ent. J) mostra a informação a respeito do tamanho da entrada em que foi utilizada naquela arquitetura. As colunas Média, Mediana, Máximo e Variância exibem os valores destas métricas obtidos nos experimentos. Na tabela, estão em destaque os melhores valores para cada uma das métricas extraídas dos resultados obtidos. Para a média das acurácias, o melhor valor obtido foi de $0,904$ na arquitetura #17, referente a uma rede com duas camadas de *hidden sizes* 20 e 20 respectivamente, e uma entrada de tamanho $J = 10$. O melhor valor da mediana obtido foi de $0,912$ na arquitetura #09, na qual possui uma única

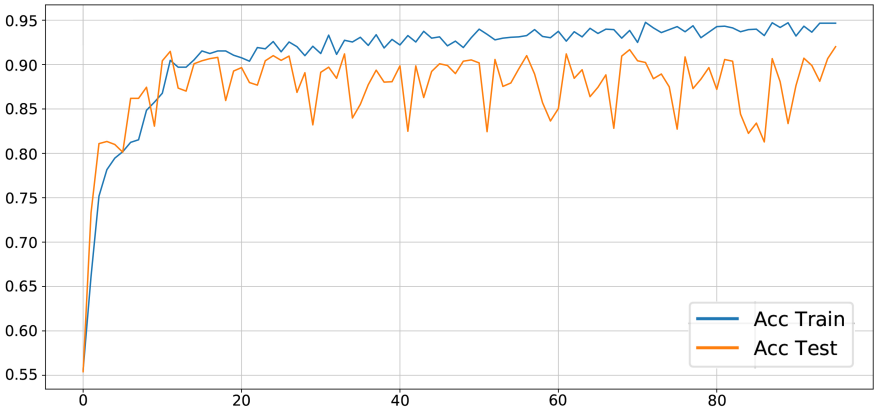
camada com $HS = 50$ e entrada de tamanho $J = 10$. O valor máximo de acurácia registado nos resultados selecionados foi de $0,949$, gerado pela arquitetura #10, que também tem uma só camada com $HS = 60$ e entrada $J = 10$. A menor variância apresentada é de aproximadamente $0,0005$ ($5,44 \times 10^{-4}$), que correspondente à arquitetura #12, que possui entrada $J = 7$ e duas camadas na rede, uma com $HS = 20$ e outra com $HS = 30$.

Dentre as arquiteturas apresentadas na Tabela 5.4, a que possui a menor complexidade computacional é a arquitetura #01. Esta arquitetura tem uma única camada de $HS = 20$ e uma entrada de tamanho $J = 7$. Em relação aos melhores valores obtidos para as métricas estabelecidas, os valores de média mediana, máximo e variância obtidos a partir desta arquitetura não são muito inferiores. A arquitetura #01 apresentou os valores de $0,888$, $0,902$, $0,932$ e $1,33 \times 10^{-3}$ para média, mediana, máximo e variância, respectivamente. Devido ao *trade-off* entre bons valores para as métricas avaliadas e a complexidade da rede, a arquitetura #01 foi a escolhida para observar de forma mais detalhada o comportamento do modelo na classificação das falhas. Após a seleção da arquitetura, resta escolher qual o *overlap* que será utilizado nas entradas. Observando os resultados exibidos na Figura 5.13, tem-se os resultados obtidos com a variação do valor da hiperparâmetro *overlap* para a entrada de tamanho $J = 7$ e tamanho da rede $HS = 20$, que foi a arquitetura escolhida. O destaque desse resultado são os obtidos com valores de *overlaps* de 4 e 5 amostras, que visualmente apresentam média, mediana e variância semelhantes. Além disso, também é possível notar um valor de máxima ligeiramente maior para o valor de *overlap* de 5 amostras, diferença que servirá como critério para escolha deste valor como o *overlap* a ser utilizado.

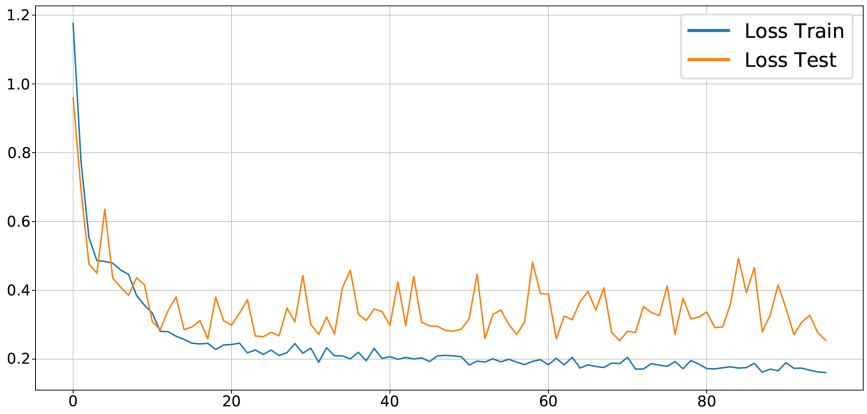
Após a definição de todos os hiperparâmetros, o modelo foi treinado e validado. A Figura 5.18 mostra o valores para acurácia e *loss* ao longo do treinamento do modelo em 97 épocas, utilizando-se os valores de hiperparâmetros selecionados.

A partir da Figura 5.18 se nota a velocidade de convergência dos valores de acurácia e de *loss* para o modelo. Também é possível observar que a acurácia de treino em maior parte das épocas é maior que a de validação. De forma análoga, este mesmo comportamento acontece para o *loss*, porém o valor do *loss* de treinamento se mantém menor que o de validação. Após observar o comportamento do modelo durante o treinamento, será realizada uma análise da classificação das falhas de validação com o modelo já treinado. A Figura 5.19 mostra a matriz de confusão M_c, referente aos resultados obtidos para classificação de falhas realizada pelo modelo com as configurações escolhidas.

As informações contidas na Figura 5.19 apresentam as acurácias individuais de classificação para cada uma das falhas de validação. Como anteriormente citado, a quantidade de entradas para o modelo é obtida a partir da equação (5.2), que para a configuração selecionada resulta em 297 entradas possíveis. A partir da matriz de confusão é possível observar esses valores destacados na sua diagonal principal, seguidos abaixo pela sua porcentagem em proporção ao total de entradas. É importante ressaltar que a matriz de confusão $M_c(i, j)$ exibe quantas vezes o modelo classificou a falha i como sendo a falha j. Assim, pode-se concluir que as falhas melhor classificadas foram as falhas $F08$ e $F11$, pois o modelo acertou 100% das vezes ao classificá-las. Esses dois resultados obtidos para essas falhas mostram que as falhas pertencentes ao grupo Sensor, vide Tabela 5.1, tem um comportamento mais parecido e melhor interpretado pelo modelo. A falha



(a) Comportamento da acurácia para dados de treino e validação durante o treinamento do modelo.



(b) Comportamento do *loss* para dados de treino e validação durante o treinamento do modelo

Figura 5.18: Acurácia e *Loss*.

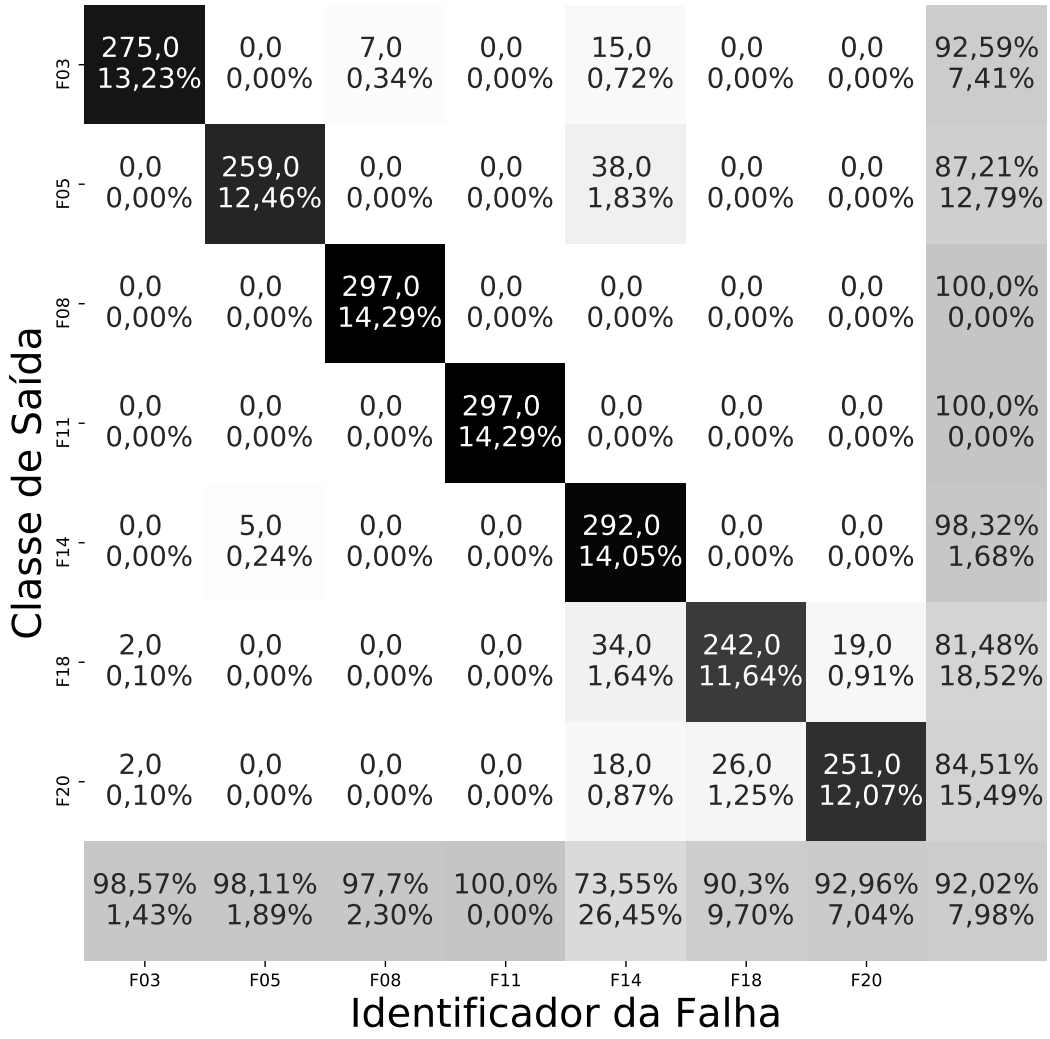


Figura 5.19: Matriz de confusão das acurácias obtidas através do modelo avaliado para a classificação das falhas de validação.

$F14$ também obteve um ótimo nível de classificação de $98,32\%$, o que revela um bom resultado para classificar falhas do tipo Estrutural, quando as falhas ocorrem na forma de vazamento do tanque. A falha $F03$ obteve $92,59\%$ de acurácia, o que está na média de classificação geral do modelo. Este resultado referente à falha $F03$ demonstra que esta configuração do modelo é eficiente para classificar falhas do grupo Atuador, com *off-set* positivo, mas não tão eficiente assim para classificar falhas com *off-set* negativo, já que a falha $F05$ obteve uma acurácia de classificação de $87,21\%$. As duas falhas restantes, $F18$ e $F20$, tiveram um desempenho consideravelmente abaixo da média, com $81,48\%$ e $84,51\%$, respectivamente. Isso sugere, que esta configuração do modelo tem dificuldade em classificar falhas do tipo Estrutural, quando estas ocorrem na válvula $V1$ e na válvula $V2$. O modelo obteve a acurácia geral de $92,02\%$ para classificar falhas do conjunto de validação.

Capítulo 6

Conclusão

Nesta dissertação foi proposto um estudo de caso para a aplicação de um método de FDI baseado em histórico de processos. Este método é composto por uma rede LSTM e uma técnica de reparametrização conhecida por *Batch Normalization*. No estudo de caso, foi testada a capacidade do método em classificar falhas obtidas a partir de um processo não simulado. O conjunto de falhas utilizado para este teste foi gerado a partir de uma planta piloto para controle de nível. A planta possui características semelhantes a de sistemas industriais reais, como a sua instrumentação e não-linearidade inerente ao sistema. Além de testar a aplicabilidade do método em um conjunto de dados mais próximo do real, foi testada a capacidade do modelo classificar falhas diferentes pertencentes ao mesmo grupo, no qual foram geradas a partir dos mesmos componentes. O segundo ponto proposto neste estudo, é a avaliação do comportamento do método a medida que alguns de seus hiperparâmetros são variados. O hiperparâmetros escolhidos foram o tamanho e a quantidade de camadas da rede LSTM, o tamanho em amostras da entrada do modelo e o *overlap* entre cada uma destas entradas.

A partir dos testes feitos neste trabalho, é possível constatar a possível aplicabilidade do método para FDI em um processo real. Pode-se destacar um dos pontos positivos, a capacidade de classificação de falhas que pertencem a mesma classe, como por exemplo as falhas $F01$ e $F03$, que são geradas pela variação positiva do *off-set* do atuador. Desta forma, o modelo foi treinado com a falha $F01$ e validado com a falha $F03$, obtendo exito em classificá-la. A partir dos testes exaustivos para avaliar o comportamento do modelo a variação dos hiperparâmetros, foi possível observar que alguns dos valores, no geral, demonstraram melhores comportamentos em suas distribuições. O tamanho da entrada J e tamanho da rede LSTM (HS) se demonstraram os hiperparâmetros mais impactantes na resultado final da classificação. As variações do *overlap* e quantidade de camadas não demonstraram alterar de forma tão significante o comportamento das distribuições, porém é importante ressaltar que esses hiperparâmetros em conjunto com os demais, geram alterações desejadas para uma escolha mais criteriosa da arquitetura, principalmente o *overlap*. Devido a quantidade limitada de amostras de processo que puderam ser avaliadas, a variação no valor do *overlap* reduzia a quantidade de entradas possíveis para o modelo, o que pode ter refletido no aprendizado do modelo, caso fosse necessário um número de épocas maior para convergência. Para a seleção dos hiperparâmetros neste estudo, foi adotado o critério de melhor relação entre uma baixa complexidade computacional e bom valor para acurácia. Esse critério foi adotado visando a possível aplicação do método para classifi-

cação de falhas em um sistema real. Foi necessário observar as combinações selecionadas de modo mais específico, não só de forma visual, se extraindo métricas estatísticas para as distribuições, com o objetivo de que o modelo atendesse as especificações requiridas. Com a seleção dos valores do hiperparâmteros, o modelo foi treinado e obteve uma acurácia final de $92,02\%$ para o conjunto de validação, alcançando 100% para algumas falhas específicas. Estes resultados apontam a validade do método utilizado para classificação de falhas em um conjunto de dados reais, como também a relevância da metodologia proposta para se avaliar a reação do modelo a variações em seus parâmetros.

6.1 Publicações e Trabalhos Futuros

Durante a pesquisa e a construção do trabalho apresentado nessa dissertação, foram desenvolvidos dois trabalhos:

- OLIVEIRA, Emerson Vilar; PINHEIRO, Yuri Thomas; GUEDES, Luiz Affonso. "Abordagem Baseada em Aprendizado de Máquina para Detecção de Alarmes Industriais". Em: *"ANAIS DO 14º SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE"*, 2019. D.O.I: 10.17648/sbai-2019-111453, disponível online em: https://proceedings.science/sbai-2019/papers/abordagem-baseada-em-aprendizado-de-maquina-para-deteccao-de-alarmes-industriais.
- OLIVEIRA, Emerson Vilar; SANTOS, Mailson Ribeiro; PINHEIRO, Yuri Thomas; GUEDES, Luiz Affonso. "Identificação de Falhas de uma Planta de Nível Piloto Baseada em Rede Neural LSTM". Em: *"23º CONGRESSO BRASILEIRO DE AUTOMÁTICA*, 2020 (aceito para publicação).

Além dos estudos apresentados aqui, outros tipos de experimentos podem ser feitos utilizando a metodologia de avaliação de hiperparâmetros, como a variação de parâmetros referentes ao treinamento, como alterações na taxa de aprendizado (*learning rate*), utilização de diferentes tipos de algoritmos de otimização, como o Adagrad, Adadelta e Adamax que são variações do Adam, além do SGD. Outro estudo que pode vir a ser feito posteriormente é um aprofundamento na rede LSTM, para observar como ocorre a transferência de informação entre seus neurônios e camadas. O objetivo, é tentar tornar mais próximo do entendimento humano como a rede toma decisões, utilizando técnicas de uma área conhecida como *Explainable Artificial Intelligence* (XAI), o que poderia aumentar a confiabilidade do modelo para um aplicação real.

Referências Bibliográficas

Ayoubi, M (1994), Nonlinear dynamic systems identification with dynamic neural networks for fault diagnosis in technical processes, *em* 'Proceedings of IEEE International Conference on Systems, Man and Cybernetics', Vol. 3, IEEE, pp. 2120–2125.

Bartyś, Michał, Ron Patton, Michał Syfert, Salvador [de las Heras] & Joseba Quevedo (2006), 'Introduction to the damadics actuator fdi benchmark study', *Control Engineering Practice* **14**(6), 577 – 596. A Benchmark Study of Fault Diagnosis for an Industrial Actuator.
URL: *http://www.sciencedirect.com/science/article/pii/S0967066105001796*

Basha, Nour, M Ziyan Sheriff, Costas Kravaris, Hazem Nounou & Mohamed Nounou (2020), 'Multiclass data classification using fault-detection-based techniques', *Computers & Chemical Engineering* p. 106786.

Bezerra, Clauber Gomes, Bruno Sielly Jales Costa, Luiz Affonso Guedes & Plamen Parvanov Angelov (2016), 'An evolving approach to unsupervised and real-time fault detection in industrial processes', *Expert systems with applications* **63**, 134–144.

Chen, ZhiQiang, Chuan Li & René-Vinicio Sanchez (2015), 'Gearbox fault identification and classification with convolutional neural networks', *Shock and Vibration* **2015**.

Chester, Daniel, David Lamb & Prasad Dhurjati (1984), Rule-based computer alarm analysis in chemical process plants, *em* 'Proceedings of the Seventh Annual Conference on Computer Technology', MICRO-DELCON 84, IEEE, pp. 22–29.

Chiang, Leo H, Mark E Kotanchek & Arthur K Kordon (2004), 'Fault diagnosis based on fisher discriminant analysis and support vector machines', *Computers & chemical engineering* **28**(8), 1389–1401.

Chopra, Tarun & Jayashri Vajpai (2011), 'Classification of faults in damadics benchmark process control system using self organizing maps', *Int. J. Soft Comput. Eng* **1**(3), 85–90.

Coppin, B. (2004), *Artificial Intelligence Illuminated*, Jones and Bartlett illuminated series, Jones and Bartlett Publishers.
URL: *https://books.google.com.br/books?id=LcOLqodW28EC*

Costa, Bruno Sielly Jales (2014), Detecção e Diagnóstico de Falhas Não-supervisionados Baseados em Estimativa de Densidade Recursiva e Classificador Fuzzy Auto-evolutivo, Tese de doutorado, UFRN, Natal, RN.

De Bruin, Tim, Kim Verbert & Robert Babuška (2016), 'Railway track circuit fault diagnosis using recurrent neural networks', *IEEE transactions on neural networks and learning systems* **28**(3), 523–533.

Dey, D, B Chatterjee, S Dalai, S Munshi & S Chakravorti (2017), 'A deep learning framework using convolution neural network for classification of impulse fault patterns in transformers with increased accuracy', *IEEE Transactions on Dielectrics and Electrical Insulation* **24**(6), 3894–3897.

Downs, J.J. & E.F. Vogel (1993), 'A plant-wide industrial process control problem', *Computers & Chemical Engineering* **17**(3), 245 – 255. Industrial challenge problems in process control.
URL: *http://www.sciencedirect.com/science/article/pii/009813549380018I*

D'Angelo, Marcos FSV, Reinaldo M Palhares, Murilo CO Camargos Filho, Renato D Maia, João B Mendes & Petr Ya Ekel (2016), 'A new fault classification approach applied to tennessee eastman benchmark process', *Applied Soft Computing* **49**, 676–686.

Frank, PM & J Wunnenberg (1989), 'Robust fault diagnosis using unknown input observer schemes', *Fault diagnosis in dynamic systems* pp. 47–97.

Gajjar, Shriram & Ahmet Palazoglu (2016), 'A data-driven multidimensional visualization technique for process fault detection and diagnosis', *Chemometrics and Intelligent Laboratory Systems* **154**, 122–136.

Gao, Xin & Jian Hou (2016), 'An improved svm integrated gs-pca fault diagnosis approach of tennessee eastman process', *Neurocomputing* **174**, 906–911.

Gertler, Janos (1991), 'Analytical redundancy methods in fault detection and isolation-survey and synthesis', *IFAC Proceedings Volumes* **24**(6), 9–21.

Graves, Alex & Jürgen Schmidhuber (2005), 'Framewise phoneme classification with bidirectional lstm and other neural network architectures', *Neural networks* **18**(5-6), 602–610.

Greff, Klaus, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink & Jürgen Schmidhuber (2016), 'Lstm: A search space odyssey', *IEEE transactions on neural networks and learning systems* **28**(10), 2222–2232.

Hadroug, Nadji, Ahmed Hafaifa & Attia Daoudi (2015), Valve actuator fault classification based on fuzzy system using the damadics model, *em* 'Proceedings of the 1st International Conference on Applied Automation and Industrial Diagnostics (ICAAID 2015), Djelfa on', pp. 29–30.

Haykin, Simon (2007), *Neural networks: a comprehensive foundation*, Prentice-Hall, Inc.

He, Yanfeng, Yali Liu, Shuai Shao, Xuhang Zhao, Guojun Liu, Xiangji Kong & Lu Liu (2019), 'Application of cnn-lstm in gradual changing fault diagnosis of rod pumping system', *Mathematical Problems in Engineering* **2019**.

Hemmer, Martin, Huynh Van Khang, Kjell G Robbersmyr, Tor I Waag & Thomas JJ Meyer (2018), 'Fault classification of axial and radial roller bearings using transfer learning through a pretrained convolutional neural network', *Designs* **2**(4), 56.

Heo, Seongmin & Jay H Lee (2018), 'Fault detection and classification using artificial neural networks', *IFAC-PapersOnLine* **51**(18), 470–475.

Himmelblau, David Mautner (1978), *Fault detection and diagnosis in chemical and petrochemical processes*, Vol. 8, Elsevier Science Ltd.

Hochreiter, Sepp & Jürgen Schmidhuber (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Ioffe, Sergey & Christian Szegedy (2015), 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', *arXiv preprint arXiv:1502.03167* .

Iri, Masao, Katsuaki Aoki, Eiji O'Shima & H Matsuyama (1979), 'An algorithm for diagnosis of system failures in the chemical process', *Computers & Chemical Engineering* **3**(1-4), 489–493.

Isermann, Rolf (1984), 'Process fault detection based on modeling and estimation methods—a survey', *automatica* **20**(4), 387–404.

Iwasaki, Yumi & Herbert A Simon (1986), 'Causality in device behavior', *Artificial intelligence* **29**(1), 3–32.

Jedliński, Łukasz & Józef Jonak (2015), 'Early fault detection in gearboxes based on support vector machines and multilayer perceptron with a continuous wavelet transform', *Applied Soft Computing* **30**, 636–641.

Kingma, Diederik P & Jimmy Ba (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Konstantinov, Konstantin B & Toshiomi Yoshida (1992), 'Real-time qualitative analysis of the temporal shapes of (bio) process variables', *AIChE Journal* **38**(11), 1703–1715.

Kourd, Yahia, Dimitri Lefebvre & Noureddine Guersi (2013), 'Fault diagnosis based on neural networks and decision trees: application to damadics', *International Journal of Innovative Computing, Information and Control* **9**(8), 3185–3196.

Lapp, Steven A & Gary J Powers (1977), 'Computer-aided synthesis of fault-trees', *IEEE Transactions on Reliability* **26**(1), 2–13.

LeCun, Yann et al. (2015), 'Lenet-5, convolutional neural networks', *URL: http://yann. lecun. com/exdb/lenet* **20**(5), 14.

Lee, Ki Bum, Sejune Cheon & Chang Ouk Kim (2017), 'A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes', *IEEE Transactions on Semiconductor Manufacturing* **30**(2), 135–142.

Lei, Jinhao, Chao Liu & Dongxiang Jiang (2019), 'Fault diagnosis of wind turbine based on long short-term memory networks', *Renewable energy* **133**, 422–432.

Leonard, James A & Mark A Kramer (1993), 'Diagnosing dynamic faults using modular neural nets', *IEEE Expert* **8**(2), 44–53.

Li, Shiwei, Yongping Zhao & Mingli Ding (2018), 'Mobile robot motor bearing fault detection and classification on discrete wavelet transform and lstm network', *Journal of Mechanics in Medicine and Biology* **18**(08), 1840034.

Lipnickas, Arūnas, J Sa da Costa & C Bocaniala (2004), Fdi based on two stage classifiers for fault diagnosis of valve actuators. application to damadics benchmark, *em* 'Proceeding of the 11th International Conference EPEPEMC', pp. 2–4.

Liu, Chang, Gang Cheng, Xihui Chen & Yusong Pang (2018), 'Planetary gears feature extraction and fault diagnosis method based on vmd and cnn', *Sensors* **18**(5), 1523.

Marciniak, A, CD Bocăială, R Louro, J Sa da Costa & J Korbicz (2003), 'Pattern recognition approach to fault diagnosis in the damadics benchmark flow control valve', *IFAC Proceedings Volumes* **36**(5), 861–866.

Marins, A (2009), *Continuous process workbench. Technical manual*, DeLorenzo Brazil, Brazil.

Ogata, Katsuhiko & Bernardo Severo (1998), *Engenharia de controle moderno*, Prentice Hall do Brasil.

Petti, Thomas F, Jim Klein & Prasad S Dhurjati (1990), 'Diagnostic model processor: using deep knowledge for process fault diagnosis', *AIChE Journal* **36**(4), 565–575.

Precup, Radu-Emil, Plamen Angelov, Bruno Sielly Jales Costa & Moamar Sayed-Mouchaweh (2015), 'An overview on fault diagnosis and nature-inspired optimal control of industrial process applications', *Computers in Industry* **74**, 75–94.

Raich, Anne & Ali Çinar (1997), 'Diagnosis of process disturbances by statistical distance and angle measures', *Computers & chemical engineering* **21**(6), 661–673.

Sorsa, Timo, Heikki N Koivo & Hannu Koivisto (1991), 'Neural networks in process fault diagnosis', *IEEE Transactions on systems, man, and cybernetics* **21**(4), 815–825.

Tian, Guanyu, Qun Zhou & Liang Du (2018), Deep convolutional neural networks for distribution system fault classification, *em* '2018 IEEE Power & Energy Society General Meeting (PESGM)', IEEE, pp. 1–5.

Udmale, Sandeep S & Sanjay Kumar Singh (2019), 'Application of spectral kurtosis and improved extreme learning machine for bearing fault classification', *IEEE Transactions on Instrumentation and Measurement* **68**(11), 4222–4233.

Venkatasubramanian, Venkat, Raghunathan Rengaswamy, Kewen Yin & Surya N Kavuri (2003), 'A review of process fault detection and diagnosis: Part i: Quantitative model-based methods', *Computers & chemical engineering* **27**(3), 293–311.

Venkatasubramanian, Venkat, Raghunathan Rengaswamy & Surya N Kavuri (2003), 'A review of process fault detection and diagnosis: Part ii: Qualitative models and search strategies', *Computers & chemical engineering* **27**(3), 313–326.

Venkatasubramanian, Venkat, Raghunathan Rengaswamy, Surya N Kavuri & Kewen Yin (2003), 'A review of process fault detection and diagnosis: Part iii: Process history based methods', *Computers & chemical engineering* **27**(3), 327–346.

Verron, Sylvain, Teodor Tiplica & Abdessamad Kobi (2006), Fault diagnosis with bayesian networks: Application to the tennessee eastman process, *em* '2006 IEEE International Conference on Industrial Technology', IEEE, pp. 98–103.

Wen, Long, Xinyu Li, Liang Gao & Yuyan Zhang (2017), 'A new convolutional neural network-based data-driven fault diagnosis method', *IEEE Transactions on Industrial Electronics* **65**(7), 5990–5998.

Wiesler, Simon & Hermann Ney (2011), A convergence analysis of log-linear training, *em* 'Advances in Neural Information Processing Systems', pp. 657–665.

Willsky, Alan & H Jones (1976), 'A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems', *IEEE Transactions on Automatic control* **21**(1), 108–112.

Yang, Rui, Mengjie Huang, Qidong Lu & Maiying Zhong (2018), 'Rotating machinery fault diagnosis using long-short-term memory recurrent neural network', *IFAC-PapersOnLine* **51**(24), 228–232.

Yin, Shen, Steven X Ding, Adel Haghani, Haiyang Hao & Ping Zhang (2012), 'A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark tennessee eastman process', *Journal of process control* **22**(9), 1567–1581.

Yu, Lu, Jianling Qu, Feng Gao & Yanping Tian (2019), 'A novel hierarchical algorithm for bearing fault diagnosis based on stacked lstm', *Shock and Vibration* **2019**.

Yuan, Mei, Yuting Wu & Li Lin (2016), Fault diagnosis and remaining useful life estimation of aero engine using lstm neural network, *em* '2016 IEEE International Conference on Aircraft Utility Systems (AUS)', IEEE, pp. 135–140.

Zhang, Senlin, Yixing Wang, Meiqin Liu & Zhejing Bao (2017), 'Data-based line trip fault prediction in power systems using lstm networks and svm', *IEEE Access* **6**, 7675–7686.

Zhao, Haitao, Shaoyuan Sun & Bo Jin (2018), 'Sequential fault diagnosis based on lstm neural network', *IEEE Access* **6**, 12929–12939.