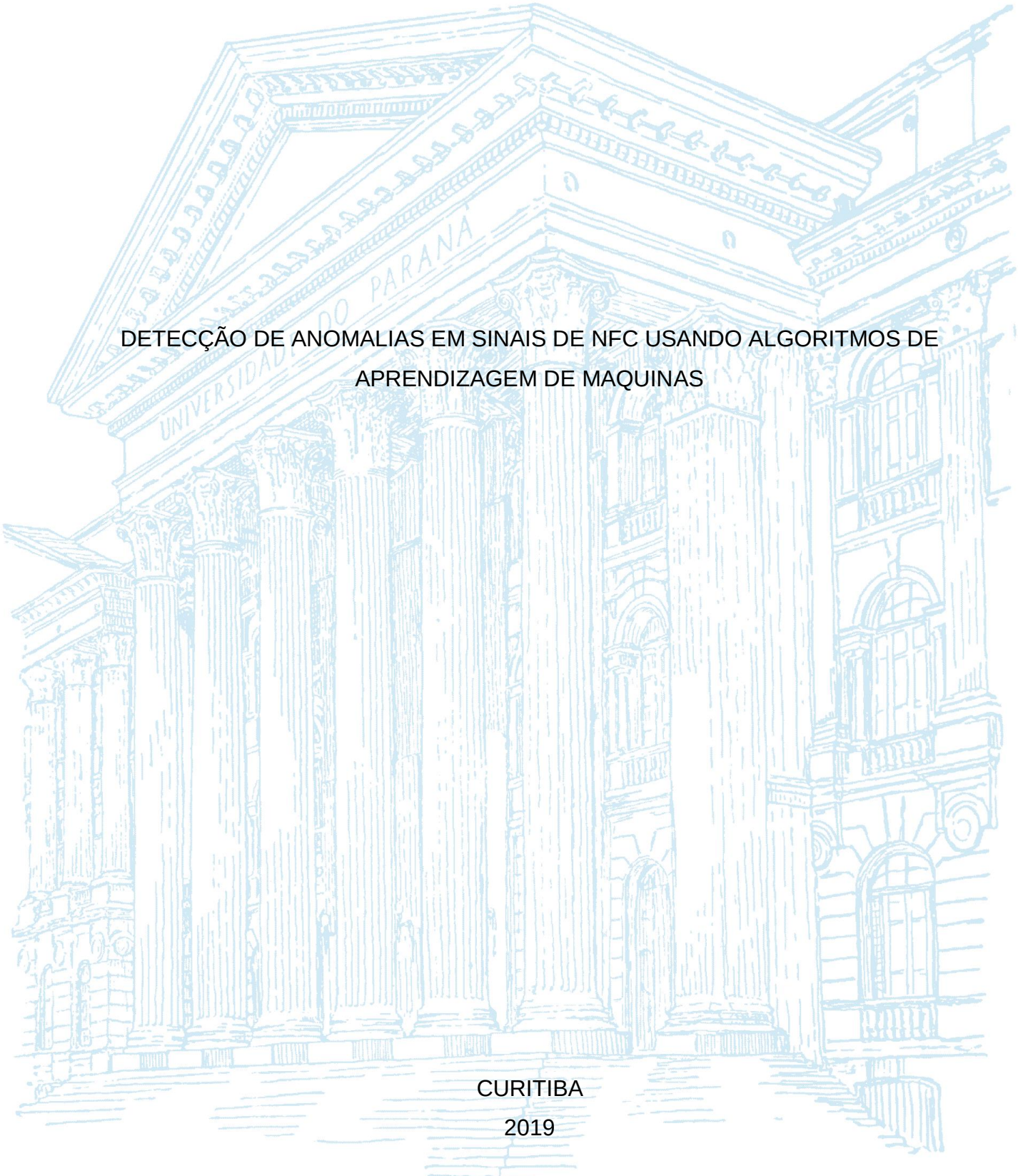
UNIVERSIDADE FEDERAL DO PARANÁ

WALTER BASTOS PFEFFER



DETECÇÃO DE ANOMALIAS EM SINAIS DE NFC USANDO ALGORITMOS DE APRENDIZAGEM DE MAQUINAS

CURITIBA

2019

WALTER BASTOS PFEFFER

DETECÇÃO DE ANOMALIAS EM SINAIS DE NFC USANDO ALGORITMOS DE APRENDIZAGEM DE MAQUINAS

TCC apresentado ao curso de Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Bacharel em Engenharia Elétrica.

Orientador:
Prof. Bernardo Rego Barros de Almeida Leite.

CURITIBA

2019

AGRADECIMENTOS

Sou grato a Renaud Chevillotte, gerente de P&D de software da NXP e a Alexandre Gailly, meu supervisor no campus de Caen. Gostaria também de agradecer a Remy Abdelwahab e André Lepine, que me ajudaram bastante durante todo o período do projeto com seus conselhos, indicações e orientações no acompanhamento do projeto.

Meus sinceros agradecimentos ao meu professor tutor Bernardo Leite. Gostaria também de agradecer a todos os meus professores da UFPR, especialmente ao professor André Mariano e o professor Eduardo Lima.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), o departamento do governo brasileiro responsável pela minha bolsa de estudos e o motivo pelo qual pude continuar meus estudos na França.

Agradeço à minha namorada Maële, que sempre me animou quando eu precisava, e aos meus amigos brasileiros Filipe Lima, João Marcos, Ricardo Barbedo, Gabriel Borba, Gabriel Bertão e Rafael Zacarias que me apoiaram durante toda a duração do projeto.

Finalmente, agradeço à minha família, meus pais e minha irmã.

RESUMO

Dada a variedade de testes realizados e a combinação de diferentes características de um sinal de radiofrequência NFC, um algoritmo de processamento de sinal padrão não é suficiente para detectar as anomalias que uma nova atualização de *firmware* pode introduzir. O objetivo deste projeto é a aplicação de um algoritmo de aprendizado de máquina capaz de detectar anomalias no sinal de radiofrequência que possam comprometer o desempenho de uma comunicação NFC após uma atualização de *firmware*.

A escolha do algoritmo de aprendizado de maquinas foi baseado nos requerimentos fornecidos para o projeto e foi realizado após o estudo de diferentes tipos de algoritmos de detecção de anomalias. Uma etapa de extração de características foi implementada para otimizar o funcionamento do algoritmo escolhido, bem como uma etapa de prova de conceito, aonde o protótipo do *software* foi testado com sinais alterados artificialmente.

O *software* final de testes foi desenvolvido usando o algoritmo K-vizinhos mais próximos e alcançou com sucesso as expectativas desejadas para este projeto, sendo capaz de detectar com grande precisão a presença de diversos tipos de anomalias adicionadas nos sinais NFC, como erros de amplitude, frequências parasitas, *overshoot* e *undershoot*.

Palavras-chave: NFC, Aprendizado de máquinas, Detecção de Anomalias

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS OU SIGLAS

NFC	- Comunicação por campo de proximidade
RFID	- Identificação por radiofrequência
RF	- Radiofrequência
IA	- Inteligência Artificial
MLPNN	- Rede neural de perceptron de multicamadas
AE	- Auto Codificador
SVM	- Máquina de vetores de suporte
KNN	- K-vizinhos mais próximos
RNA	- Rede neural artificial
PICC	- Cartão de acoplamento indutivo de proximidade
PCD	- Dispositivos de acoplamento de proximidade

SUMÁRIO

Introdução

NFC é uma especificação que permite a comunicação sem fio entre dois dispositivos ao aproximá-los. O princípio é simples: um dos dispositivos desempenha o papel de iniciador, responsável pela tarefa de iniciar a comunicação e controlar a troca de informações, o outro desempenha o papel de alvo, respondendo às demandas do iniciador. A NFC é, em certa medida, baseada na tecnologia RFID [1], que permite a existência de dispositivos de identificação que utilizam radiofrequência como meio de comunicação.

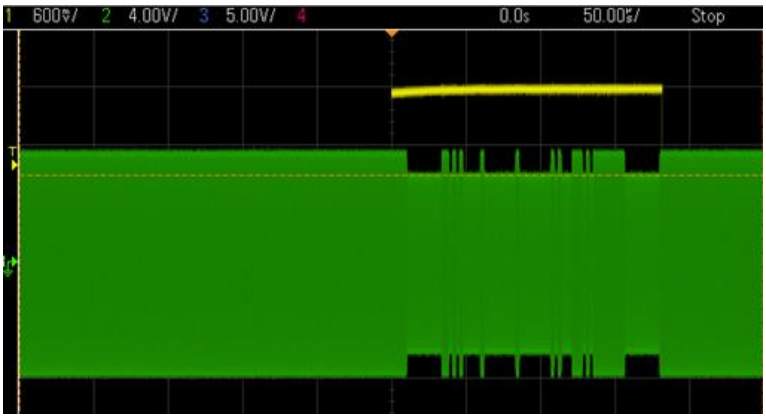


Figura 1 - Exemplo de sinal NFC, adquirido com um osciloscópio.
Fonte: O autor (2019)

Para garantir o bom funcionamento de um dispositivo NFC, testes de não regressão para cada alteração de *firmware* dos chips NFC são necessários. Um teste de não regressão serve para verificar se o desempenho operacional de um dispositivo NFC não foi impactado negativamente por uma alteração de *firmware*.

Este trabalho de conclusão de curso foi realizado em conjunto com a empresa NXP *semiconductors*, onde realizei um estágio de seis meses. Mais precisamente, meu estágio foi localizado na cidade de Caen/França, onde a NXP possui mais de 300 empregados, sendo que 70% são dedicados à pesquisa e desenvolvimento.

O foco principal da NXP de Caen é o desenvolvimento e melhoramento da tecnologia NFC [2], tecnologia da qual eles foram os inventores junto com a empresa Sony e são atualmente os líderes mundiais em vendas e produção.

Devido a pedidos de diversos clientes e também de órgãos reguladores da tecnologia NFC, a equipe com a qual trabalhei deseja melhorar esses testes de

controle de qualidade e, para isso, uma das principais etapas é baseada nos sinais RF.

Atualmente, os testes realizados pela equipe da NXP em Caen analisam somente o formato gráfico de um sinal NFC utilizando um osciloscópio com testes de máscara porém, para melhorar a performance desses testes, é necessário que a equipe de testes tenha aplicativos capazes de interpretar os sinais RF transmitidos e recebidos pelos dispositivos NFC e detectar possíveis erros.

Baseado no problema apresentado, o objetivo deste trabalho é a criação de um algoritmo capaz de analisar uma nova amostra de um sinal de RF NFC e detectar se há alguma anomalia que possa ser considerada como uma falha na geração do sinal pelo dispositivo NFC. Este procedimento é conhecido como detecção de anomalia [3], onde uma nova amostra de um sinal é comparada com um banco de dados já existente para classificá-la como **normal** ou **anômala**.

A principal necessidade do programa de testes é que ele possa se adaptar aos diferentes tipos de padrões e normas que os sinais de RF NFC devem respeitar, logo, o uso de um algoritmo de aprendizado de máquina parece ser adaptado para este caso.

No início de 2019, em um projeto que realizei anteriormente com a NXP junto com engenheiros com um maior conhecimento do assunto [4], foi realizado um estudo de diferentes tipos de algoritmos de detecção de anomalias, variando de uma análise simples de processamento de sinal, métodos de agrupamento e muitos tipos diferentes de redes neurais [5], que poderiam ser aplicados para detectar erros nos sinais de RF NFC. Este estudo será utilizado como base para escolher o algoritmo final a ser utilizado neste projeto.

Para garantir que o programa final atendesse a todos os requisitos necessários, trabalhei em conjunto com os engenheiros da NXP Remy Abdelwahab e André Lepine para a validação dos módulos da aplicação de detecção de anomalias e também de seu desempenho.

1.1 Objetivos

1.1.1 Objetivo geral

O objetivo geral desse trabalho de conclusão de curso é desenvolver uma aplicação para computadores baseada em inteligência artificial (IA) para detecção de anomalias que deve ser:

- De fácil manutenção: Caso seja necessário mudar algo no código fonte da aplicação, um engenheiro da NXP deve ser capaz de realizar essas mudanças o mais rápido possível. Para isso, além de uma documentação completa do aplicativo, o programa final deve ser o mais simples possível.
- Capaz de ser expandida para acomodar novas especificações da equipe de engenheiros da NXP: Como as especificações utilizadas pelas equipes da NXP para validar o funcionamento de um chip NFC são incrementadas constantemente para aumentar o controle de qualidade, o programa final deve ser capaz de ter as características do sinal NFC que ele analisa expandidas indefinidamente.
- Suficientemente genérica para que qualquer tipo de sinal de RF NFC possa ser classificado por ela: Existem diversos tipos diferentes de sinais RF NFC que serão definidos no próximo capitulo. O programa final deve ser capaz de analisar o comportamento de todos esses diferentes tipos.

A aplicação também deve utilizar a linguagem Python [6] e, se necessário, pode usar bibliotecas de aprendizado de máquina para Python, como o scikit-learn [7] e o TensorFlow [8].

1.1.2 Objetivos específicos

Como objetivos específicos desse trabalho tem-se:

a) Escolha do algoritmo de detecção de anomalia baseado em IA mais adaptado para o problema apresentado.

b) Compressão de informações presentes em uma única amostra de um sinal RF NFC, usando uma aplicação de extração de características [9].
c) Validação de que o algoritmo de IA escolhido pode ser usado para detectar erros em um sinal de RF NFC.
d) Criação de um banco de dados para todos os testes de sinais de RF NFC que estão sendo usados pela NXP.
e) Com esse banco de dados, criar um aplicativo de IA que pode ser adicionado no final da sequência do banco de testes, para validar que um chip NFC que está sendo testado funciona de acordo com as especificações NFC.

2 Algoritmo de IA para detecção de anomalias

Neste capítulo, todos os métodos de detecção de anomalias estudados serão descritos em detalhes e o mais adaptado ao nosso problema será escolhido com base na simplicidade final do aplicativo e nas limitações do banco de dados disponível.

Os requisitos para a escolha do algoritmo definidos pela NXP podem ser simplificados na seguinte lista:

1. O algoritmo deve ser o mais simples possível: qualquer outro engenheiro da NXP deve ser capaz de usar facilmente o aplicativo desenvolvido e, se necessário, ser capaz de entender seu funcionamento e o modificar o mais rapidamente possível. A transição entre usos do aplicativo deve ser o mais fácil possível.
2. O único banco de dados necessário deve ser os sinais normais de NFC RF capturados: É difícil para a NXP criar um banco de dados de casos anômalos por dois motivos principais: a frequência de aparecimento de um sinal anômalo é muito inferior à de um sinal normal; e mesmo se pudermos gerar o caso anômalo mais frequente, não é possível ter amostras de todo tipo de anomalia.

De fato, o nome mais adaptado para o problema deste projeto é a detecção de novidades. A principal diferença entre detecção de novidades e detecção de anomalias é que o primeiro utiliza um banco de dados composto apenas por casos normais, onde o último precisa de informações sobre casos normais e anômalos. Mas continuarei a usar os termos anomalia pelo resto deste documento, visto que o termo anomalia é bem mais difundido no campo de aprendizado de máquina.

Antes de começar a descrição dos diferentes algoritmos estudados para este projeto, a próxima parte do capítulo se concentrará em uma explicação dos diferentes tipos de sinais RF NFC e as suas características principais. Essas informações serão muito importantes para a escolha da topologia final de detecção de anomalias nesses sinais.

2.1 Especificações de sinais NFC RF

Com base na especificação da norma ISO / IEC 14443 [10], cada sinal NFC RF tem características estritas que devem ser seguidas, como tempo de queda da

modulação, tempo de subida, frequência da portadora, coeficiente de modulação, frequência de transferência de dados, valores de *overshoot* e *undershoot*.

Embora existam muitas especificações que um sinal NFC RF deve respeitar, os valores dessas especificações podem ser definidos pela classificação do sinal com base em três características principais: modo, tipo e taxa de bits. O aplicativo final deste projeto deve ser capaz de executar corretamente a extração de características de um sinal para qualquer permutação possível dessas características:

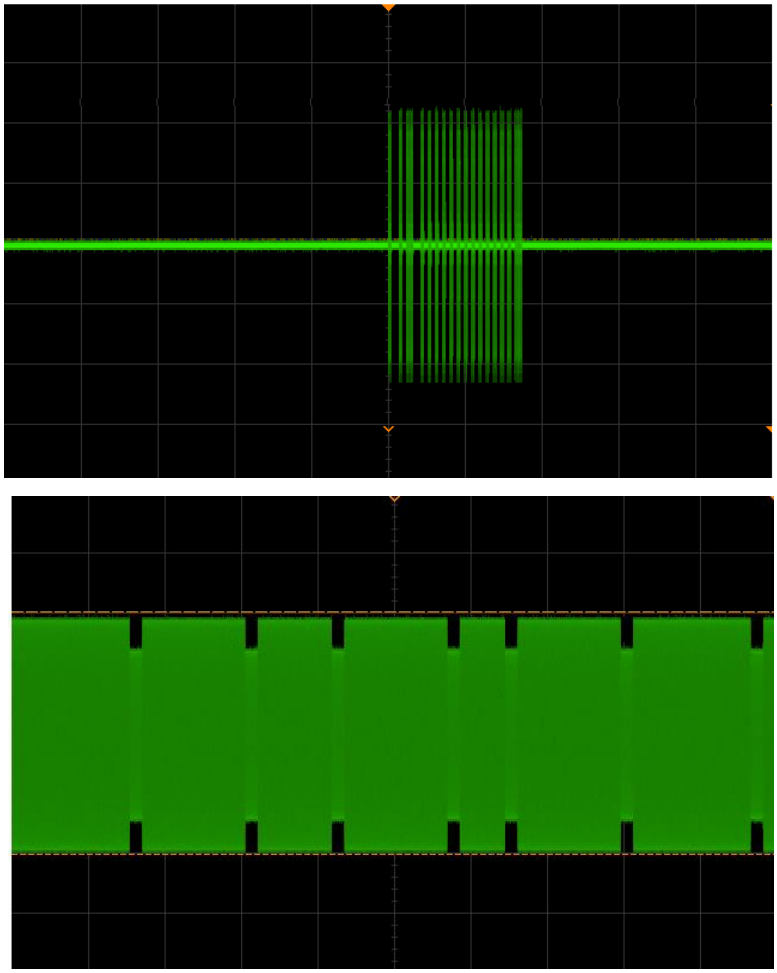


Figura 2 – Sinal modo cartão (alto) e sinal modo *leitor* (baixo)
Fonte: O autor (2019)

- Modo: O sinal NFC RF possui dois modos: PICC (cartão de acoplamento indutivo de proximidade) → PCD (dispositivos de acoplamento de proximidade), que chamaremos de sinal do *cartão*; e PCD → PICC, que chamaremos de sinal do *leitor* (Figura 2).

- Um sinal de *cartão* é o resultado da modulação do sinal da portadora, gerado pelo leitor RF, através do aumento e diminuição de sua amplitude pelo leitor RF.
- Um sinal de *leitor* é o resultado da modulação do sinal da portadora, gerado pelo leitor RF, através do uso de comutação de impedância no cartão NFC.

- Tipo: Existem 3 tipos de NFC e todos eles usam modulação de amplitude: NFC-A, com base na ISO / IEC 14443A, com codificação Miller e um coeficiente de modulação AM de 100%; NFC-B, com base na ISO / IEC 14443B, com codificação NRZ-L e um coeficiente de modulação AM de 10%; e NFC-F, baseado em FeliCA JIS X6319-4 [11] e que também possui um coeficiente de modulação AM de 10% porém utiliza a codificação Manchester.

- Taxa de bits: existem quatro taxas de bits possíveis para transferência de dados: 106 kbps, 212 kbps, 424 kbps e 848 kbps. NFC-A e NFC-B são adaptados a todas as taxas de bits, e o NFC-F usa apenas taxas de bits de 212 kbps e 424 kbps.

2.2 Algoritmos de IA para detecção de anomalias

Para o algoritmo de IA, quatro métodos foram examinados, dois baseados em uma topologia de rede neural: perceptron de multicamadas (MLPNN) [12] e auto codificador (AE) [13]; e outros dois baseados em classificação / regressão: máquina de vetores de suporte (SVM) [14] e K-vizinhos mais próximos (KNN) [15]. Esses métodos podem ser usados para detecção de anomalias em um banco de dados.

2.2.1 Algoritmos de rede de neurônios

Uma rede neural artificial (RNA) é um sistema matemático projetado para replicar o funcionamento básico dos neurônios biológicos e a conexão entre eles, como mostra a Figura 3.

A maioria das redes neurais são sistemas supervisionados de aprendizado de máquina, ou seja, a saída esperada deve ser expressa previamente para que o sistema possa ser treinado.

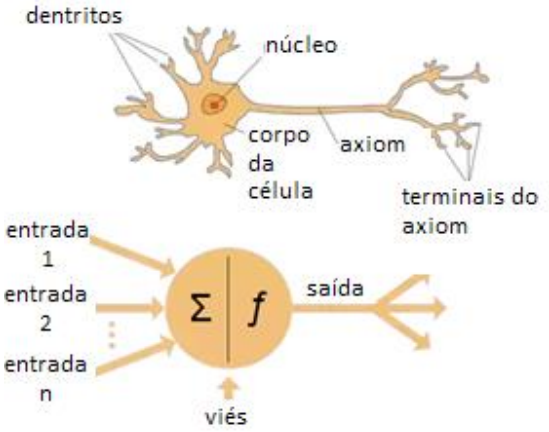


Figura 3 – Comparação entre um neurônio biológico e um neurônio artificial
Adaptado de: https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7, acesso em 21/10/2019

2.2.1.1 Perceptron de multicamadas

A arquitetura MLPNN é um dos primeiros modelos de redes neurais, projetado na década de 1980 [12]. A rede contém três partes principais: a camada de entrada, a camada de saída e um conjunto de camadas ocultas, que pode conter várias camadas vinculadas entre elas. O uso de várias camadas ocultas permite que a rede aprenda padrões e dependências complexas entre os dados de entrada e saída.

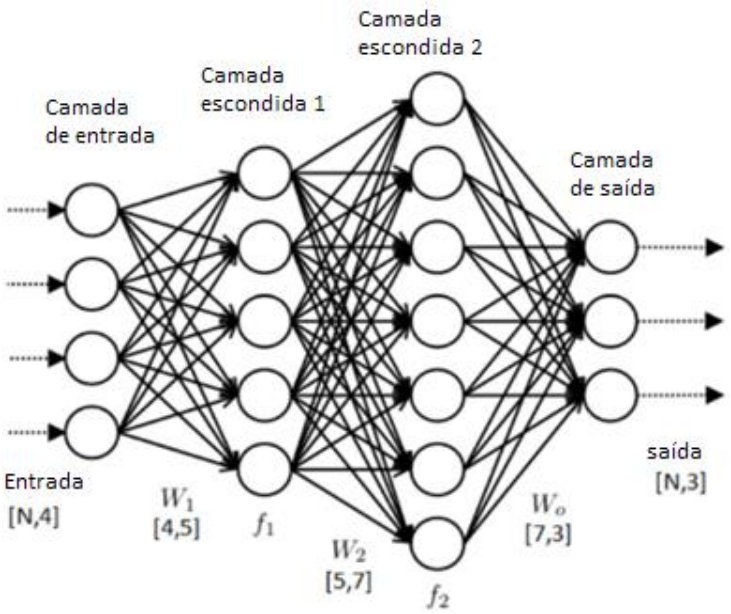


Figura 4 – Arquitetura básica de uma MLPNN
Adaptado de: https://www.datasciencecentral.com/profiles/blogs/how-to-configure-the-number-of-layers-and-nodes-in-a-neural, acesso em 21/10/2019

Mas é também importante notar que a adição de várias camadas ocultas conectadas pode aumentar bastante a complexidade do modelo final da MLPNN. Essa complexidade se traduzirá em um custo computacional maior e a necessidade de uma maior base de dados para treinar a enorme quantidade de parâmetros de ligação entre neurônios que precisam ser calculados.

Todas as camadas são ligadas por parâmetros chamados ‘pesos’, representados pelas linhas na Figura 4, e para cada neurônio, representado pelos círculos na Figura 4, existe uma função de ativação f que decide se o estado do neurônio mais próximo estará ligado (estado alto) ou desligado (estado baixo).

No caso de redes MLP, essa função de ativação é geralmente um sigmoide, mostrado na Figura 5.

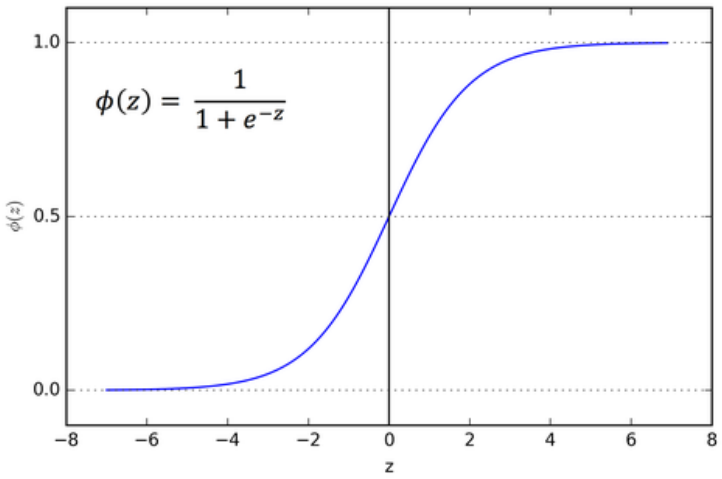


Figura 5 – Função sigmoide
Fonte: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, acesso em 21/10/2019

As funções de ativação baseadas em sigmoides são muito úteis para a classificação e a escolha, pois limitam o resultado entre os valores 0 (desativado) e 1 (ativado).

O objetivo dessa rede neural é prever os mesmos valores de saída esperados usando os valores de entrada. Isso é feito alterando os parâmetros de peso que vinculam suas camadas. Para isso, uma etapa de aprendizado é necessária, em que esses parâmetros serão otimizados de uma maneira controlada para obter o melhor resultado no final.

Se a otimização desses parâmetros for possível, a rede aprenderá os padrões e características do banco de dados que lhe foi fornecido. Em seguida, é possível

usar essa rede para classificar tipos de dados por padrões entre eles que podem ser invisíveis à primeira vista.

Esse tipo de rede neural é um dos primeiros a ser capaz de realizar processamento de imagens, como, por exemplo, para o reconhecimento da escrita manual em fotos [16].

Para o problema de detecção de anomalias, se um banco de dados de ambos sinais considerados normais e anômalos for fornecido, uma MLPNN poderá aprender a classificar um novo sinal entre normal e anômalo com base nas informações fornecidas anteriormente.

2.2.1.2 Auto codificador

Um AE é uma rede neural capaz de aprendizado não supervisionado, o que significa que ela não precisa saber o resultado esperado para melhorar a si mesma a cada iteração de seu treinamento. Ela é normalmente usada para aprender uma representação compactada (codificação) de um conjunto de dados.

Seu algoritmo é baseado na compressão e descompressão do sinal de entrada, para que possa ser reconstruído como um sinal de saída, conforme mostrado na Figura 6. É muito importante que a camada de código tenha menos variáveis que as camadas de entrada e saída, para forçar a rede a compactar as informações que caracterizam o sinal.

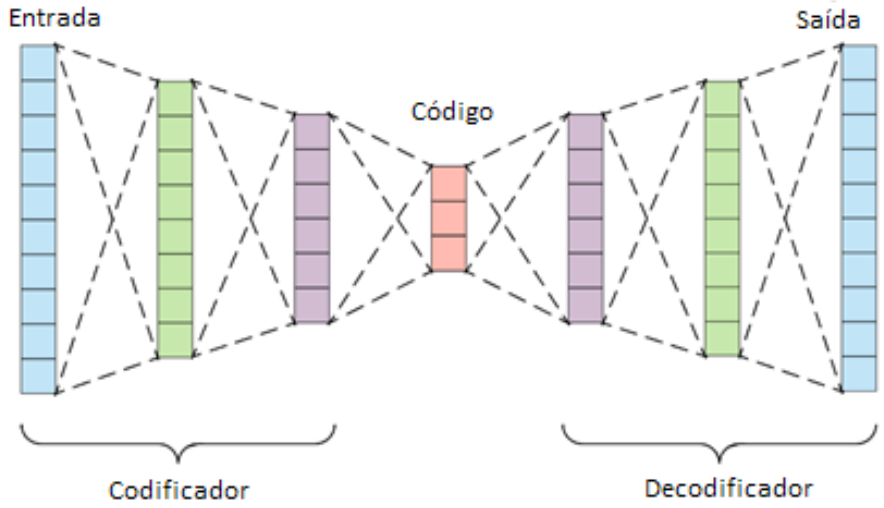


Figura 6 – Arquitetura básica de uma rede AE
Adaptado de: https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368, acesso em 21/10/2019

A função que caracteriza o aprendizado de uma rede neural AE é a diferença entre o sinal de entrada e o sinal de saída. Para cada iteração do algoritmo de aprendizado, o AE tentará convergir para ter o mesmo valor para cada neurônio de saída que os neurônios de entrada. Após a conclusão do estágio de aprendizado, a rede deverá ser capaz de reconstruir um sinal na etapa do decodificador usando as características da etapa do codificador.

No caso em que um sinal com anomalias for usado como entrada, a etapa do codificador criará características desconhecidas para o decodificador, que o converterá em um sinal de saída diferente do sinal de entrada. A partir dessa diferença, o sistema pode concluir que o sinal de entrada possuía uma anomalia.

Para nossa detecção de anomalias em NFC, a comparação do sinal de entrada original e do sinal de saída reconstruído por um AE já treinado com sinais NFC normais pode ser usada para detectar anomalias. Se a diferença entre os sinais for maior que um determinado valor limite empírico, poderíamos inferir que o sinal de entrada é anômalo.

2.2.2 Algoritmos de classificação/regressão

Os algoritmos de classificação / regressão são métodos capazes de agrupar um conjunto de um banco de dados em um mesmo subconjunto com base nas semelhanças e nos padrões comportamentais que uma parte dos dados tem em comum, como mostra a Figura 7 abaixo.

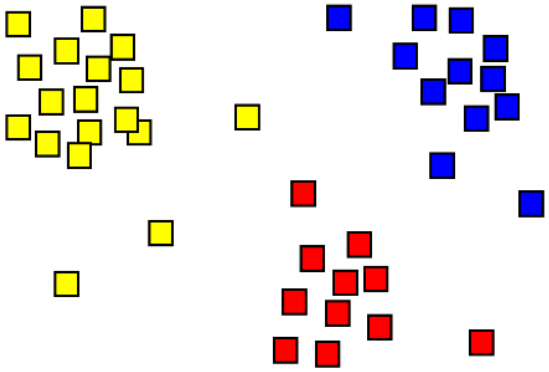


Figura 7 – Classificação de um base de dados 2D em 3 diferentes grupos
Fonte: https://en.wikipedia.org/wiki/Cluster_analysis, acesso em 21/10/2019

Esses algoritmos podem ser usados para muitas aplicações, como reconhecimento de padrões, classificação, compactação de dados, análise de dados e, é claro, detecção de anomalias. Nas próximas seções, detalharei dois algoritmos de classificação que podem ser usados neste projeto.

2.2.2.1 Máquina de vetores de suporte

As SVMs são métodos supervisionados de classificação criados para classificação binária, o que os torna bons candidatos ao problema de classificação NFC. Seu funcionamento baseia-se na criação de um hiperplano que tem a maior distância do ponto de dados de treinamento mais próximo. Após a otimização do hiperplano, o conjunto de dados em estudo será dividido em dois grupos diferentes, dependendo de qual lado do hiperplano ele está localizado.

A função a ser otimizada durante o treinamento de uma SVM integra um termo de qualidade de previsão e um termo de complexidade do modelo. A busca pelos hiperplanos é introduzida usando um núcleo que codifica os dados com uma transformação não linear, como mostra a Figura 8. Essa transformação procura distorcer o espaço definido da base de dados para permitir a separação de classes por um único hiperplano. Numericamente, todas as equações são definidas como produtos escalares do núcleo e certos pontos do banco de dados, esses pontos são chamados de vetores de suporte.

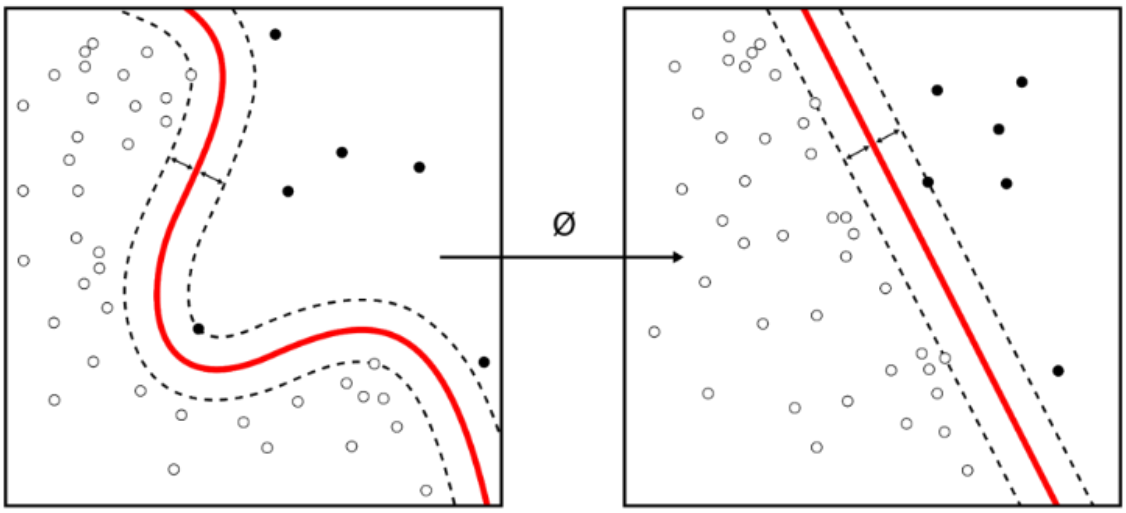


Figura 8 - Transformação de núcleo (Φ) de dados 2D, que permite que um hiperplano 1D separe os dados em dois grupos distintos.
Fonte: https://en.wikipedia.org/wiki/Support-vector_machine, acesso em 21/10/2019

As vantagens da SVM são que elas são bem adaptadas para a classificação de dados de grandes dimensões e para o problema de identificação de grupo. As desvantagens deste método são: a necessidade de escolher o melhor modelo de núcleo a ser usado; e o fato de que é um método que possui um treinamento supervisionado.

Este último ponto é considerado uma desvantagem visto que nos requerimentos apresentados no início do capitulo podemos utilizar apenas sinais normais como banco de dados. Um treinamento supervisionado significa que este método também requer informações sobre sinais anômalos, algo que não teremos acesso com o banco de dados final.

2.2.2.2 K-vizinhos mais próximos

O princípio do método KNN é encontrar um número predefinido de pontos de treinamento K, chamados vizinhos, que são os mais próximos de um novo ponto de amostra que será classificado. A classificação deste novo ponto será definida com base na quantidade e ao grupo ao qual pertencem esses K vizinhos que o rodeiam.

Por exemplo, na Figura 9 abaixo, a nova amostra (círculo verde) deve ser classificada ou na classe de quadrados azuis ou na classe de triângulos vermelhos.

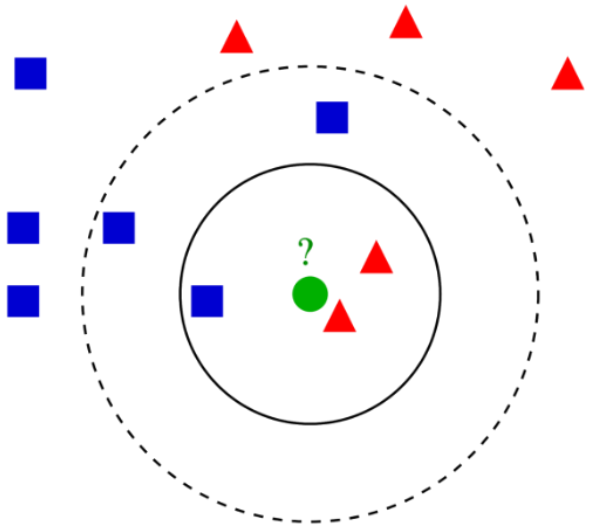


Figura 9 – Escolha baseada no conjunto mais próximo de vizinhos em um banco de dados 2D
Fonte: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm, acesso em 21/10/2019

Se o número de vizinhos selecionados K = 3 (círculo sólido), a nova amostra será atribuída à classe de triângulos vermelhos, porque existem 2 triângulos e somente 1 quadrado dentro do círculo interno. Se K = 5 (círculo tracejado), o novo

ponto será atribuído à classe de quadrados azuis, uma vez que existem 3 quadrados contra somente 2 triângulos dentro do círculo externo.

O uso da distância como parâmetro principal para o cálculo de agrupamento pode ser considerado uma desvantagem, uma vez que quanto maior o número da dimensão, menor a relevância relativa de uma coordenada do conjunto de dados. Isso é conhecido como a maldição da dimensionalidade [17], e pode diminuir a precisão do algoritmo. Um dos métodos mais utilizados para estimar a distância entre o novo ponto a ser classificado e seus vizinhos é a distância euclidiana [18].

O KNN é um método simples de treinar e adaptado aos problemas de classificação, por isso pode ser um bom candidato ao problema de detecção de anomalias NFC.

2.3 Escolha do algoritmo

Com base nos requisitos apresentados no início do capitulo, podemos classificar os métodos apresentados para encontrar os mais adaptados às nossas necessidades:

1. Rede Neural de perceptron de multicamadas: MLPNN é o algoritmo menos adaptado para o nosso caso, uma vez que eles precisam de um banco de dados de sinais normais e anômalos e são relativamente complexos quando comparados aos outros métodos apresentados, tento que variar o número de neurônios em cada camada, o número de camadas, a função de ativação e outros parâmetros que impossibilitam alguém sem experiência no assunto de modificar rapidamente a aplicação final.
2. Auto codificador: assim como o MLPNN, os AE são mais complexos em comparação com os métodos de agrupamento, portanto, mesmo sendo capaz de realizar a detecção de anomalias sem um banco de dados anterior de sinais anômalos, não é o método mais adaptado para este projeto.
3. Máquina de vetores de suporte: as SVMs já são muito mais simples que as soluções de redes neurais, mas ainda precisam de informações sobre as anomalias na etapa de treinamento do algoritmo, logo, não podem ser aplicadas neste projeto.

4. K-vizinhos mais próximos: o KNN é o método mais adaptado para o nosso caso, uma vez que precisa apenas de informações sobre sinais normais e tem apenas um parâmetro para modificar (o número de k vizinhos), o que o torna um dos métodos mais simples para ser modificado caso necessário.

É importante ressaltar que apesar de ter escolhido o KNN como método mais adaptado, realizei testes iniciais com todos os métodos apresentados, obtendo resultados parecidos em todos os casos.

Por exemplo, a utilização do AE para detecção de frequências parasitas dentro de um sinal NFC RF foi capaz de detectar anomalias com 100% de precisão quando o sinal parasita possuía um valor mínimo de 2,5% do sinal original. Já a utilização do método MLPNN conseguiu detectar problemas nos sinais com um acoplamento de antenas a uma distância de 10 cm, o que é um resultado interessante, visto que essa é a distância máxima para a comunicação NFC, o que se traduz em um efeito de acoplamento mínimo entre as antenas.

Porém esses testes só foram possíveis pois usei um banco de testes fornecido pela NXP com casos normais e anômalos, o que não é algo que poderei utilizar no treinamento da aplicação final.

2.3.1 Modificações necessárias

Mesmo que o KNN seja o algoritmo mais adaptado dos quatro apresentados, ele não pode ser aplicado diretamente ao nosso problema, uma vez que a aplicação padrão de um método KNN é para o agrupamento de um banco de dados em diferentes classes. Se quisermos usá-lo para detecção de anomalias, serão necessárias algumas modificações.

Normalmente, o método KNN encontra a distância de uma nova amostra aos seus vizinhos K mais próximos. Porém, se K for definido como 1, para cada nova amostra, o algoritmo retornará apenas a distância ao ponto mais próximo da base de dados que foi utilizada para o treinamento.

Portanto, se separarmos o banco de dados inicial em dois conjuntos: o conjunto de treinamento, que é uma grande porcentagem de todo o conjunto, e um conjunto

‘estimador de distância’, podemos avaliar a distância máxima entre um caso normal e seu vizinho mais próximo, fazendo o seguinte:

1. Treinar o algoritmo KNN usando o subconjunto de treinamento de 80% da base de dados total escolhidos aleatoriamente.
2. Aplicar a esse modelo treinado a outra parte do banco de dados. Para cada amostra deste segundo subconjunto, o modelo retornará um valor de distância.
3. Escolher o maior valor de distância, chamado β. Isso significa que esse valor de distância β é o maior que ainda é aceitável para um sinal normal. Qualquer novo sinal que tenha um valor de distância maior que aquele poderá ser considerado anômalo.

A Figura 10 mostra a ideia básica desta modificação para detecção de anomalias, usada para classificar dois novos sinais (verde e vermelho) em um modelo treinado (quadrados azuis). β é a maior distância entre todos os quadrados azuis. Este exemplo é para um banco de dados de 2 dimensões.

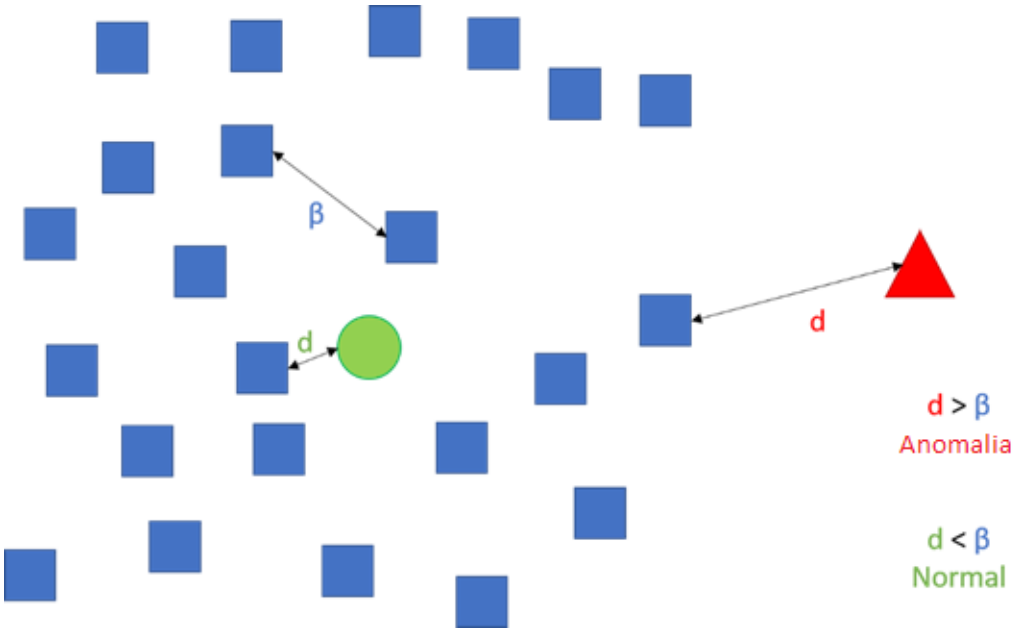


Figura 10 – Detecção de anomalias utilizando KNN com K=1
Fonte: O Autor (2019)

Opcionalmente, um valor de *tolerância do modelo* pode ser usado para alterar o limite da decisão de detecção de anomalias. Por exemplo, para um valor de *tolerância de modelo* de 1,5, isso significa que qualquer novo sinal que tenha uma distância que seja no máximo 1,5 vezes β ainda será considerado normal.

Esse valor de *tolerância do modelo* é o único parâmetro que precisa ser escolhido para o treinamento do modelo quando um banco de dados grande o suficiente é usado. Este valor deve ser escolhido de acordo com o banco de dados e ser testado durante a validação do modelo.

Se o banco de dados não for grande o suficiente, um segundo parâmetro, chamado *tolerância de banco de dados* deverá ser adicionado. Isso será explicado em profundidade no capítulo 3.

2.4 Extração de características

Nesta seção, apresentarei o conceito de extração de características, uma etapa importante na implementação deste algoritmo KNN modificado.

Uma única amostra de um sinal de RF NFC extraído com um osciloscópio pode ter mais de 100.000 pontos de dados, dado que é um sinal modulado em amplitude com uma frequência portadora de 13,56 MHz, amostrada a mais de 200 MHz por um tempo superior a 500 µs. Ainda, a frequência de amostragem pode ir até 2GHz e o tempo de amostragem pode ir até 5ms. O uso de uma frequência de amostragem 10 vezes maior, ou até mais, do que a frequência portadora pode parecer um exagero, dado o teorema de Nyquist para amostragem de sinais, mas é necessário, como será explicado nos parágrafos seguintes.

Nosso foco neste projeto é analisar anomalias no envelope do sinal de RF, logo, um método de extração de envelope é necessário. O método escolhido é aplicar uma transformada de Hilbert ao sinal RF NFC original e salvar o valor absoluto da transformada. Apesar de existirem outros métodos de extração de envelopes de sinais AM, a transformada de Hilbert foi o método que apresentou o melhor compromisso entre quantidade de ruído presente envelope extraído e o tempo de cálculo necessário para a extração.

Esse método é simples e rápido de implementar na linguagem Python, mas a quantidade de ruído presente no envelope extraído é inversamente proporcional à frequência de amostragem do sinal original, como mostra a Figura 11, onde um mesmo sinal é amostrado com frequências diferentes (33 MHz e 1 GHz) e a diferença entre o ruído entre os dois casos é visível.

Esses níveis de ruído são obtidos após a filtragem do sinal. Para esta filtragem, um filtro FIR (resposta finita ao impulso) passa-baixa foi projetado. Este filtro possui uma frequência de corte de 3,4 MHz e uma atenuação de 80 dB após a região de transição de uma década. Essa frequência é alta o suficiente para não perdermos nenhuma informação importante, pois a componente de frequência máxima que nos interessa é de 848 kHz (a maior taxa de bits possível para a comunicação NFC). Considerando os níveis de ruído apresentados, existe um interesse real em ter a maior frequência de amostragem possível para ajudar a contrabalancear este ruído.

Porém, devido à *maldição da dimensionalidade* apresentada no capítulo 2.2.2.2, o aumento do número de pontos de dados devido à maior frequência de amostragem resultará em uma grande perda de desempenho, portanto, um estágio de extração de características precisa ser adicionado ao projeto.

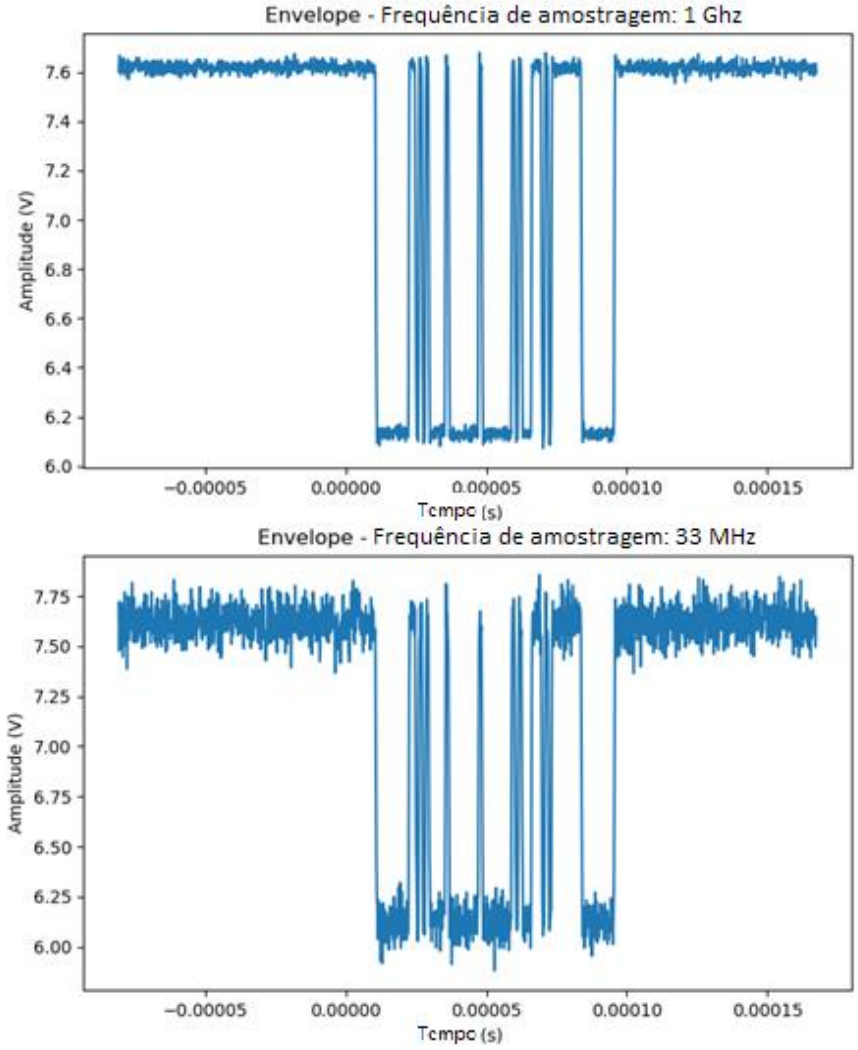


Figura 11 - Amplitude de ruído com base na frequência de amostragem, para o envelope de um sinal RF NFC-B com uma taxa de transferência de 106 kbps
Fonte: O autor (2019)

Nos algoritmos de aprendizado de máquina, a extração de características é conhecida como a redução da quantidade de dados necessários para descrever uma única amostra de um banco de dados enquanto tenta-se minimizar a perda de informações. Normalmente este processo é realizado com o uso do processamento de sinais.

Para a extração, dois tipos principais de características foram definidas: características diretamente vinculadas a um sinal NFC RF, como a duração de um bit, o coeficiente de modulação de amplitude e a frequência principal do envelope; e características que geralmente são usadas na extração de características para qualquer tipo de sinal temporal, como o valor médio do sinal, seu desvio padrão, e seus valores máximos e mínimos.

2.4.1 Processo de extração de características

O primeiro passo para a extração de características é o pré-processamento do sinal do envelope. Esta etapa serve para definir as partes do sinal do envelope que contêm informações importantes, como as bordas ascendentes e descendentes de uma modulação de bit e que devem ser analisadas mais rigorosamente. Isso é feito com um código em Python criado por mim e que depende de três variáveis: limiar, expansão e corte. Esses três parâmetros serão explicados em profundidade a seguir.

- Limiar: é o parâmetro que serve para separar a parte de alto nível e a contraparte de baixo nível de um sinal modulado. Seu valor deve estar entre 0 e 1. Por exemplo, um valor de 0,3 significa que qualquer coisa abaixo de 30% da amplitude da borda descendente será considerado como o nível baixo de uma modulação.
- Expansão: É o parâmetro que serve para aumentar o tamanho do que será percebido como a área de modulação.
    - Por exemplo, se uma modulação tiver 1000 pontos de amostra como nível baixo, com um valor de expansão de 0,1, o número de pontos que serão considerados como nível baixo será transformado em 1200 pontos, expandindo a área de modulação. Isso se traduz em um aumento de 20%, pois adicionamos 10% à esquerda e 10% à direita do sinal.
    - Isso pode ser usado para aumentar artificialmente o número de pontos que serão considerados como nível baixo em sinais em que não se pode definir

um nível de limiar próximo ao início da borda descendente devido a um alto nível de ruído, como no caso do sinal amostrado a 33 MHz da Figura 11. Para esse sinal, um limiar muito próximo de 0 resultaria em várias falsas detecções de níveis baixos devido ao ruído. Para obter um falso limiar perto de 0, podemos então definir um limiar de 0,5 e utilizar o coeficiente de expansão para aumentar a área que será considerada como nível baixo.

- Corte: para sinais do tipo *leitor*, 'corte' representa uma porcentagem (por exemplo: corte: 0,1 = 10%) do início do sinal que será cortado (isso é importante para a detecção de modulação, dado que, para os sinais do tipo *leitor*, a função de extração do características usará os primeiros 5% do sinal como referência e, portanto, não deve haver modulação nessa parte do sinal). Para sinais do cartão, ‘corte’ representa uma porcentagem do final do sinal que será cortado (para sinais de cartão, a função de extração de recurso usa os últimos 5% do sinal como referência).

Após a escolha desses três parâmetros, o script de extração de características é aplicado nas partes do sinal que foram consideradas como contendo uma informação importante. Essas partes do sinal estão definidas na Figura 12.

Na Figura 12, a parte azul dos envelopes é a parte não interessante do sinal; as partes vermelhas dos envelopes são as partes consideradas como o nível baixo do sinal modulado; e as partes amarelas do sinal são consideradas como o nível alto do sinal modulado. As áreas vermelha e amarela são consideradas partes interessantes do sinal e serão usadas para a extração de características.

No momento em que este documento foi escrito, o número máximo de características extraídas de uma única amostra de um sinal é 42. Isso representa uma compactação de dados de aproximadamente 2380 vezes de uma de amostra de 100.000 pontos de dados. Uma lista com todas as características usadas, como elas foram calculadas e uma breve explicação de como cada uma se correlaciona com as especificações NFC podem ser encontradas no anexo A.

Essas 42 características são divididas em 22 características baseadas nas especificações NFC, e 10 características genéricas usadas em algoritmos de aprendizados de máquinas que são obtidas do sinal inteiro e também de seu gradiente temporal, o que totaliza mais 20 características.

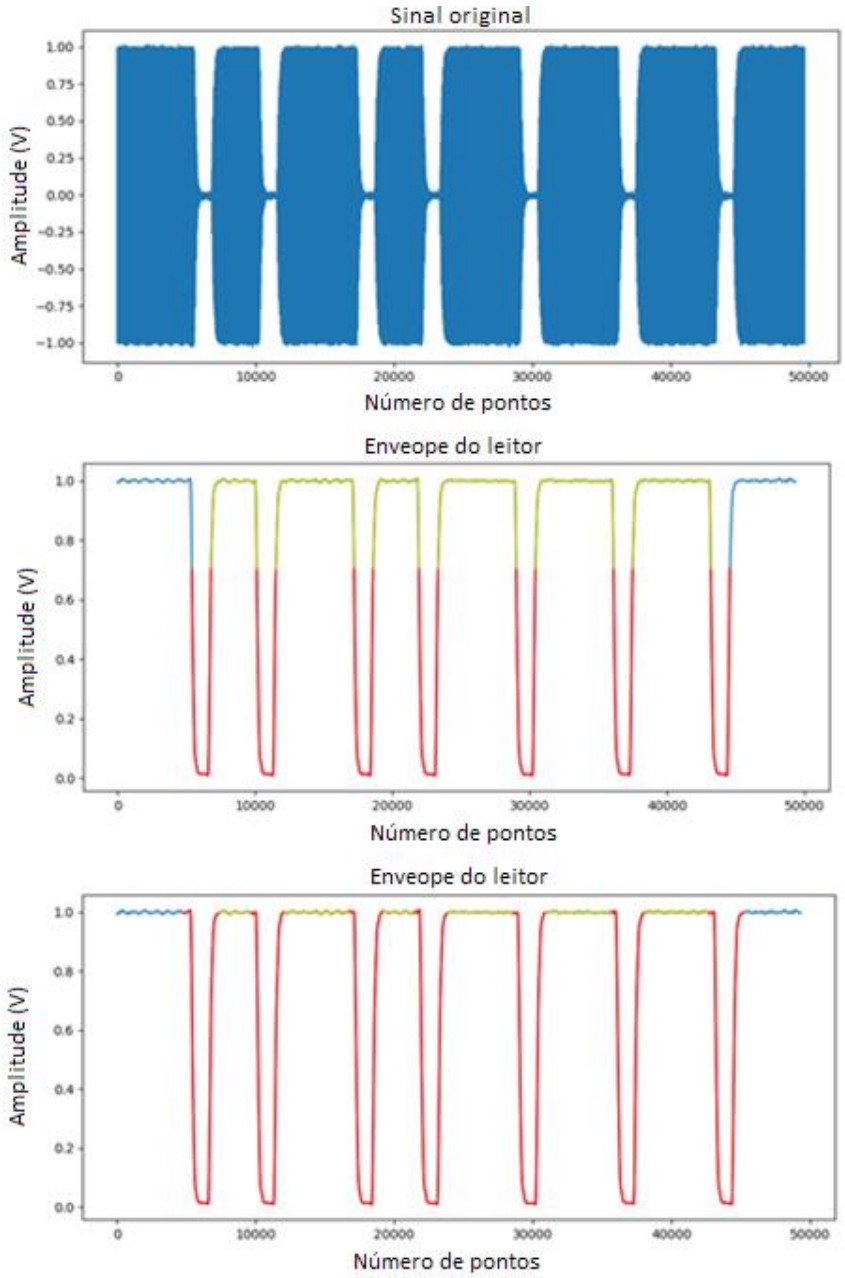


Figura 12 - Primeira imagem: Sinal original de 106 kbps do NFC-A. Segunda imagem: Envelope com valor limite de 0,3 e expansão de 0. Terceira imagem: Envelope com valor limite de 0,3 e expansão de 0,4.
Fonte: O autor (2019)

Essas ultimas 20 podem ser adicionadas e removidas do programa final de acordo com a necessidade do usuário. Através de testes foi percebido que a adição dessas características genéricas aumenta consideravelmente a sensibilidade do sistema à ruído.

Caso o nível de ruído seja de interesse para o usuário, a adição dessas características corresponde a uma melhora na performance do sistema, porém, caso

o nível de ruído não seja importante para o usuário, recomenda-se não utilizar esse conjunto de características, ou ao menos não utilizar as características geradas pela análise do gradiente do sinal, pois essas são as mais sensíveis ao ruído.

3 Prova de conceito

Este capítulo detalha uma primeira aplicação de teste do algoritmo de agrupamento KNN com o uso das características extraídas apresentadas no capítulo 2 e no anexo A. O objetivo desta seção é validar o uso desta solução, com a criação de um banco de dados de um sinal RF NFC e com a adição de erros artificiais no sinal, a fim de testar os limites do algoritmo de detecção de anomalias.

3.1 Tolerância de banco de dados

Antes de iniciar a criação do modelo de aprendizado de máquina, há outro parâmetro a ser criado, a *tolerância do banco de dados*. Esse parâmetro serve como um gerador de aumento de dados, ou seja, serve para aumentar artificialmente a quantidade de amostras que usaremos como nosso banco de dados, como exemplificado na Figura 13.

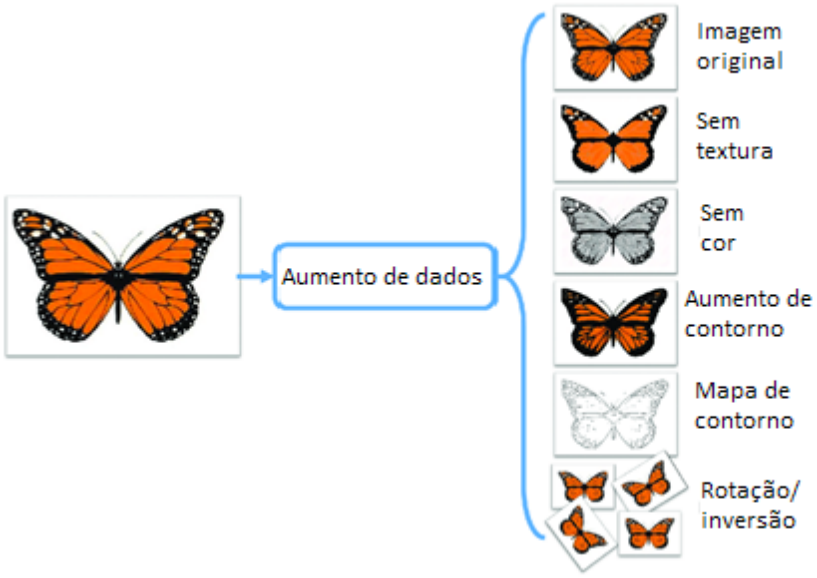


Figura 13 - Exemplo de aumento de dados para imagens: criação de versões diferentes de uma mesma foto, para multiplicar o tamanho do banco de dados
Adaptado de: https://www.researchgate.net/figure/Data-augmentation-using-semantic-preserving-transformation-for-SBIR_fig2_319413978, acesso em 25/10/2019

Esta etapa é importante principalmente para bancos de dados pequenos que não possuem amostras suficientes para permitir que nosso modelo treinado se adapte a todas as especificações da NFC. Por exemplo, se a especificação NFC permitir um

período de modulação entre 2 µs e 4 µs, mas tivermos apenas exemplos de sinal de 2,7 µs a 3,3 µs, o modelo classificará sinais que estão entre 2 µs e 2,7 µs e entre 3,3 µs e 4 µs como anomalias. E isso seria um falso negativo, um erro.

A solução proposta é introduzir uma nova variável no sistema, chamada *tolerância ao banco de dados*, que pode modificar levemente as características extraídas de uma amostra. Por exemplo, para um valor *de tolerância de banco de dados* de 0,2, multiplicaríamos cada característica de uma amostra com um número aleatório entre 0,8 e 1,2. Faríamos isso N vezes para cada amostra, aumentando artificialmente o tamanho do banco de dados por um fator de N.

Isso significa que uma amostra do banco de dados que possui um período de modulação de 3 µs será transformado em N amostras que teriam um período de modulação entre 3 * (1 ± 0,2) µs. Devido a essa transformação, todos os períodos de modulação entre 2,4 µs e 3,6 µs agora são representados no modelo. Este aumento de dados pode ser aplicado a todos os recursos definidos no anexo A.

O valor da *tolerância do banco de dados* pode ser definido como o mesmo para todos as características extraídas, ou cuidadosamente escolhido para cada uma, para garantir que toda a especificação NFC para um determinado tipo de sinal esteja contida no modelo KNN final.

Embora o aumento de dados seja amplamente usado em algoritmos de aprendizado de máquina, o caso ideal é ter um banco de dados de sinais reais que contenha todos os exemplos possíveis de sinais que vão do limite inferior ao superior das especificações da NFC. Infelizmente, a criação de um banco de dados desse tamanho exigiria um tempo muito longo.

3.2 Teste

Para validar a hipótese do algoritmo de aprendizado de máquinas apresentado no capítulo anterior e na seção 3.1, criamos um conjunto de dados com 500 amostras de um sinal de leitor NFC-A a 106 kbps, gerado por uma placa NXP *Chameleon* (Figura 14). Em seguida, criamos modelos KNN diferentes para diferentes níveis de *tolerância do banco de dados* e tentamos detectar anomalias com eles.

Figura 14 - Na ordem da esquerda para a direita: NFC FORUM POLLER-0 que será usada para introduzir frequências parasitas no sinal original; Placa de demonstração NXP PN547 que será usada para introduzir anomalias de acoplamento de antena; Placa NXP Chameleon, *usada para gerar os sinais do leitor.*
Fonte: O autor

O sinal base, apresentado na Figura 15, tem um pico de tensão de 1,1 V, mas foi normalizado por *software* para ter um pico de tensão de 1 V, em um esforço para reduzir o impacto de diferentes amplitudes de sinal no sistema. É um sinal de leitor NFC-A com 8 bits de informação, amostrado a 200 MHz.

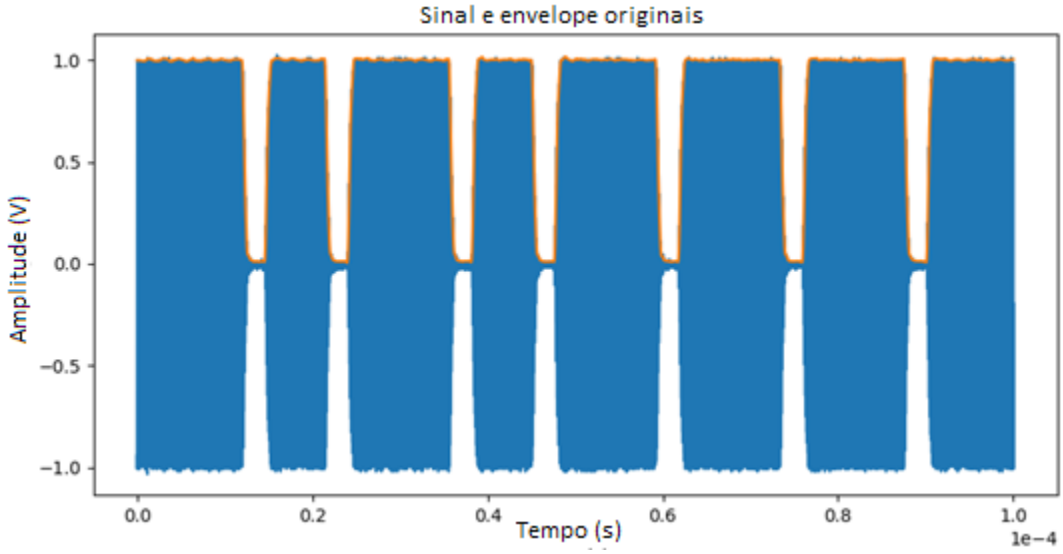


Figura 15 - Sinal de leitor NFC-A a 106kpbs
Fonte: O autor (2019)

Como se trata de um teste de controle, precisamos saber previamente o tipo de anomalia que está contida nos sinais anômalos. Para isso, adicionamos 3 tipos diferentes de anomalias geradas artificialmente:

1. Alterar a tensão de pico do sinal RF NFC para: 0,33 V, 0,65 V, 0,875 V, 1,04 V, 1,14 V e 1,23 V.
2. Usando uma placa NXP FORUM POLLER-0 (Figura 14), adicionamos uma frequência parasita de 11 MHz ou 12 MHz ao sinal original, com uma tensão pico a pico de 15 mV, 30 mV ou 45 mV
3. Adicionar outra antena, usando a placa de demonstração NXP PN547 (Figura 14), dentro do volume de trabalho da antena do gerador NFC para criar uma perturbação no sinal original.

O uso desses três tipos de anomalias foi escolhido baseado na simplicidade em gera-los, visto que a NXP já possuía o material necessário para adicionar esses tipos de anomalias e também existiam casos documentados de erros de comunicação causados pelos 3 tipos de sinais anômalos apresentados.

Para cada anomalia, amostramos 20 sinais e classificamos os resultados em 3 categorias, com base na classificação do modelo KNN treinado com os 500 sinais normais da base de dados:

- Tudo normal - se os 20 sinais foram detectados como um sinal normal
- Somente anomalias - Se os 20 sinais foram detectados como um sinal anômalo
- x/20 - Se o modelo detectou sinais normais e anômalos no banco de dados, onde x representa o número de sinais anômalos detectados.

Para todos os testes, esperamos que as anomalias de tensão próximas à amplitude inicial de 1,1 V, com uma variação máxima de 10% do valor inicial, não sejam consideradas como anomalias, já que estamos apenas alterando ligeiramente a amplitude do sinal e isso que não interfere no desempenho de um dispositivo NXP.

As anomalias criadas pela adição de outra frequência ou outra antena ao volume de trabalho devem ser detectadas como anomalias reais. Os resultados podem ser encontrados na Tabela 1.

Tabela 1 – Resultados da prova de conceito

	Tolerância do banco de dados			
Anomalias adicionadas	**0**	**0.05**	**0.1**	**0.2**
0,33 Vp	Somente anomalias	Somente anomalias	Somente anomalias	Somente anomalias
0,65 Vp	Somente anomalias	Somente anomalias	Somente anomalias	Somente anomalias
0,88 Vp	Somente anomalias	Somente anomalias	Tudo normal	Tudo normal
1,04 Vp	Somente anomalias	Tudo normal	Tudo normal	Tudo normal
1,14 Vp	Somente anomalias	Tudo normal	Tudo normal	Tudo normal
1,23 Vp	Somente anomalias	Tudo normal	Tudo normal	Tudo normal
11 MHz 15 mVpp	Somente anomalias	Somente anomalias	3/20	2/20
11 MHz 30 mVpp	Somente anomalias	Somente anomalias	11/20	6/20
11 MHz 45 mVpp	Somente anomalias	Somente anomalias	Somente anomalias	16/20
12 MHz 15 mVpp	Somente anomalias	Somente anomalias	5/20	4/20
12 MHz 30 mVpp	Somente anomalias	Somente anomalias	14/20	9/20
12 MHz 45 mVpp	Somente anomalias	Somente anomalias	Somente anomalias	19/20
Perturbação por outra antena	Somente anomalias	Somente anomalias	Somente anomalias	Somente anomalias

Ao analisar os resultados, descobrimos que os melhores resultados foram obtidos usando um valor de *tolerância de banco de dados* de 0,05 para o sinal do leitor NFC-A a 106 kbps. Com ele, o modelo detecta a variação de amplitudes entre

1,04Vpp e 1,23Vpp como normal enquanto todas as outras anomalias artificiais foram detectadas corretamente como anomalia.

Para ajudar na visualização da precisão do modelo e como ele pode detectar variações que normalmente não podemos perceber à primeira vista, as figuras do anexo B comparam um sinal normal e um sinal que foi considerado anômalo.

3.3 Resultados da prova de conceito

Os resultados obtidos mostram que a combinação do algoritmo KNN e a extração de características pode ser usada para detecção de anomalias. De fato, ele pode detectar anomalias em um sinal NFC RF que não podem ser percebidas a olho nu, como a contaminação do sinal original com uma onda senoidal de 11 MHz 15 mVpp, conforme apresentado na terceira figura do anexo B.

Após todos os testes apresentados em 3.2, reproduzimos o mesmo procedimento de teste para as demais permutações de sinal da especificação NFC:

- NFC-A - *leitor*; com taxas de bits: 106 kbps; 212 kbps; 424 kbps e 848 kbps
- NFC-B - *leitor*; com taxas de bits: 106 kbps; 212 kbps; 424 kbps e 848 kbps
- NFC-F - *leitor*; com taxas de bits: 212 kbps e 424 kbps
- NFC-A - *cartão*; com taxas de bits: 106 kbps; 212 kbps; 424 kbps e 848 kbps
- NFC-B - *cartão*; com taxas de bits: 106 kbps; 212 kbps; 424 kbps e 848 kbps
- NFC-F - *cartão*; com taxas de bits: 212 kbps e 424 kbps

Os resultados para todas as outras permutações de NFC acima seguem a mesma tendência dos apresentados para o NFC-A - *leitor* a 106 kbps, ou seja, conseguimos detectar as anomalias artificiais definidas anteriormente em todas elas. É importante notar que os dois parâmetros (*tolerância do banco de dados* e *tolerância do modelo*) devem ser adaptados para cada sinal diferente. Para todos os casos estudados, um valor de *tolerância de modelo* de 1,5 parece ser um bom compromisso e apenas a *tolerância do banco de dados* teve que ser adaptada empiricamente para cada caso.

Com esses resultados, podemos concluir que a arquitetura apresentada de detecção de anomalias funcionará para as aplicações necessárias deste projeto.

4 Aplicação final

Após a validação do método KNN com a prova de conceito, o último objetivo deste projeto é a criação da aplicação final, que testará por anomalias nos sinais NFC RF de todos os testes de não regressão de um dispositivo NXP NFC.

A criação deste aplicativo pode ser dividida em duas partes principais: A criação de um banco de dados, que conterá todas as informações necessárias para a criação de um modelo KNN; e a criação do próprio modelo KNN. Depois disso, a única coisa que resta a fazer é criar um script Python que execute a extração de características de um novo sinal RF NFC e aplicá-las ao seu modelo KNN correspondente. O resultado desta operação será usado para classificar o novo teste de não regressão como **aprovado** (normal) ou **reprovado** (anômalo).

4.1 Criação da base de dados

Existem aproximadamente 600 casos de teste diferentes para o teste de não regressão de um único chip NXP NFC. Essa enorme quantidade é necessária para garantir que o sinal gerado pelo chip sendo testado respeite todas as especificações da NFC. Portanto, geralmente, um teste corresponde à validação de uma única especificação para uma única permutação das possíveis características de NFC definidas no capítulo 2.4.1.

Isso significa que devemos criar 600 modelos, porque as informações que o sinal contém influenciam o modelo KNN final. Por exemplo, um leitor NFC-A com sinal de RF de 106 kbps que transfere um fluxo de 8 bits '00010010' cria um modelo KNN diferente de um sinal NFC-A de leitor a 106 kbps de RF que transfere um fluxo de 8 bits '11010010'.

Devido ao enorme tamanho do banco de dados necessário para a validação de todos os testes, precisamos criar um script para extrair automaticamente as características de cada caso de teste e armazená-los, para serem usados posteriormente na criação de seu respectivo modelo KNN.

Após a conclusão de um teste, um arquivo .csv é criado com a forma de onda do sinal NFC que foi medido com um osciloscópio. Isso é feito para cada teste,

portanto, no final da validação de um chip, existem aproximadamente 600 arquivos .csv a serem analisados.

O script desenvolvido para a criação do banco de dados desses testes pode ser separado em duas partes: o script Python de extração de características principal que segue as instruções descritas no anexo A e um arquivo .json que contém os parâmetros para a extração de características de cada caso de teste

Um arquivo .json (JavaScript Object Notation) é um formato de troca de dados que é fácil para os humanos lerem e escreverem e também é fácil para as máquinas analisarem e gerarem. O fato de esse formato ser fácil de entender para os seres humanos é importante para a nossa aplicação, pois se um novo teste for criado, deve ser o mais fácil possível incluir esse novo teste no script de extração de características.

```
{
    "extract": [
        {
            "tag": "ANA_CARD_A_Host-VPP",
            "mode": "r",
            "threshold": 0.3,
            "expansion": 0.4,
            "save_image": true,
            "cut": 0,
            "type_e": 0
        },
        {
            "tag": "ANA_CARD_A_P73-VPP",
            "mode": "c",
            "threshold": 0.1,
            "expansion": 0.05,
            "save_image": true,
            "cut": 0.1,
            "type_e": 0
        }
    ]
}
```

Figura 16 - Exemplo de um arquivo .json, com os parâmetros para dois testes: ANA_CARD_A_Host-VPP e ANA_CARD_A_P73-VPP
Fonte: O autor (2019)

Esse arquivo .json contém 7 parâmetros para cada teste: `tag`, que é o nome do teste; `mode`, que pode ser ‘c’, para *cartão* ou ‘r’, para *leitor*; `threshold`; `expansion`; `cut`, que são as variáveis de extração de recursos definidas no capítulo 2.4.2 .; `save_image`, que pode ser definido como ‘verdadeiro’ ou ‘falso’, dependendo se o usuário quiser armazenar também uma imagem .png do sinal, como as figuras do anexo B; e `type_e`, que é usado para escolher um conjunto de características a serem salvas: ‘0’ para salvar todas as características, ‘1’ para todas as características exceto

as geradas a partir do gradiente do sinal e ‘2’ para salvar apenas as características relacionadas às especificações NFC. Um exemplo de arquivo .json usado para o script pode ser encontrado na Figura 16.

Para cada `tag` no arquivo .json, o script procurará um arquivo .csv com o mesmo nome. Se ele encontrar um arquivo correspondente, usará os parâmetros que foram definidos abaixo de `tag` para extrair as características. No final, ele criará um arquivo .npy que contém todos as características extraídas do sinal e, se `save_image` for definido como ‘verdadeiro’, uma imagem .png. Este arquivo .npy terá o mesmo nome que o arquivo .csv.

Como o arquivo .npy pode ser aberto apenas por um script Python, também é criado um arquivo de texto que contém uma breve descrição com o valor de todas as características, para que o usuário possa acessá-lo se desejar analisar as características extraídas.

Após a criação de um banco de dados grande o suficiente para cada teste, um novo script será usado para treinar todos os modelos KNN que serão usados para a validação de um chip NXP.

4.2 Criação dos modelos KNN

Para a criação de um modelo KNN, precisamos garantir que todos os sinais que serão usados no treinamento não apresentem anomalias. Embora seja possível, uma análise manual de cada sinal de caso de teste levaria muito tempo, portanto, uma das soluções é a análise da distribuição estatística das características das amostras de um teste.

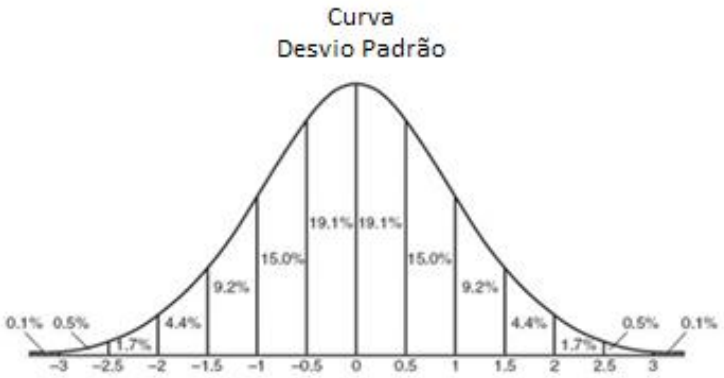


Figura 17 – Probabilidade de uma característica ser encontrada em uma curva gaussiana Adaptado de: https://www.mathplanet.com/education/algebra-2/quadratic-functions-and-inequalities/standard-deviation-and-normal-distribution, acesso em 26/10/2019

Ao ajustar a distribuição estatística de cada caso de teste a uma curva gaussiana, esperamos encontrar o desvio padrão de cada uma das características e usá-lo como fator de seleção. Por exemplo, se um determinado sinal de um caso de teste tiver uma característica que não esteja em um intervalo de duas vezes o seu desvio padrão, poderíamos estimar que ele possui alguma anomalia e não deve ser usado para treinar o modelo KNN. Como mostra a Figura 17, o intervalo de duas vezes o desvio padrão deve representar 95% das características dos sinais normais.

Esse método não é perfeito, pois existe a possibilidade de descartarmos sinais perfeitamente normais mas, para o treinamento do modelo, é melhor descartar um sinal normal do que incluir um sinal anômalo.

4.3 A aplicação de teste

Após a criação do modelo KNN, o script de teste o utilizará para classificar novos sinais em normal ou anômalo. Se uma anomalia for detectada, esse script também gera um arquivo de texto que contém uma lista com as principais características que causaram essa anomalia. Isso permite que o usuário final entenda por que o sinal foi considerado anômalo, uma vez que a anomalia pode ser invisível à primeira vista, como foi mostrado no capítulo 3.2. O funcionamento da aplicação de teste é ilustrado pelo fluxograma da Figura 18.

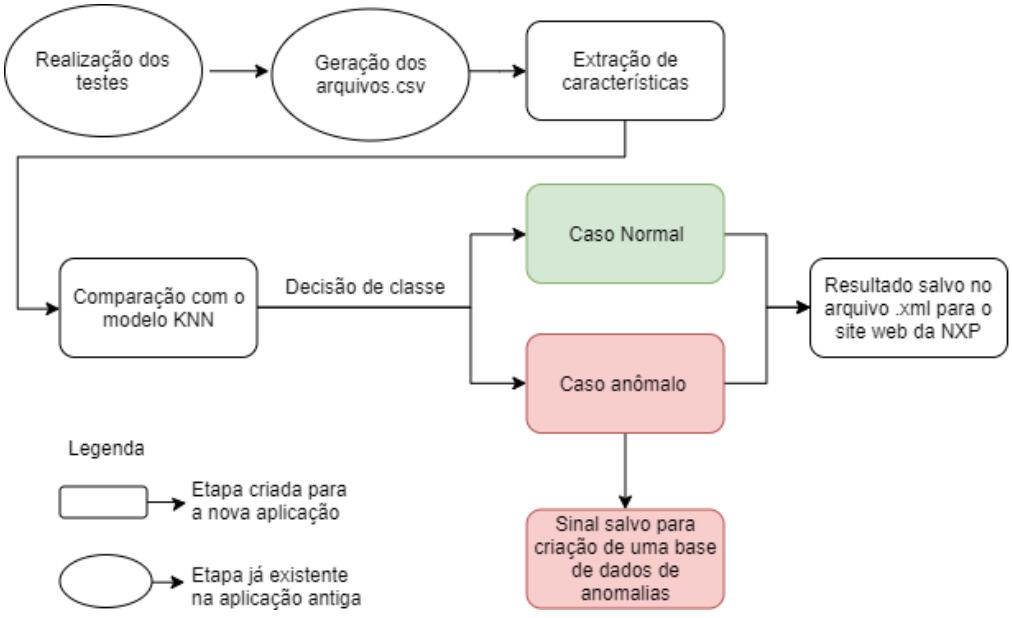


Figura 18 – Fluxograma do funcionamento da aplicação de testes
Fonte: O autor (2019)

Finalmente, existe também o conceito de que os modelos sempre se aprimoram, usando sinais normais recém-classificados para aumentar o tamanho do banco de dados de treinamento. Portanto, após uma classificação normal de uma amostra, o script principal transferirá o arquivo de características desse sinal para uma pasta diferente, onde ele será armazenado e posteriormente utilizado para treinar novamente o modelo KNN.

Esta etapa pode ajudar com o problema de generalização da especificação NFC e, no futuro, remover completamente a necessidade da variável de *tolerância do banco de dados*, uma vez que o banco de dados será grande o suficiente, conforme explicado no capítulo 3.1.

Depois de realizar todos os testes de não regressão para anomalias, o script adicionará os resultados a um arquivo .xml já existente, carregado no servidor da NXP. Este arquivo permite a geração de uma representação gráfica dos resultados de cada dia, como mostra a Figura 19.

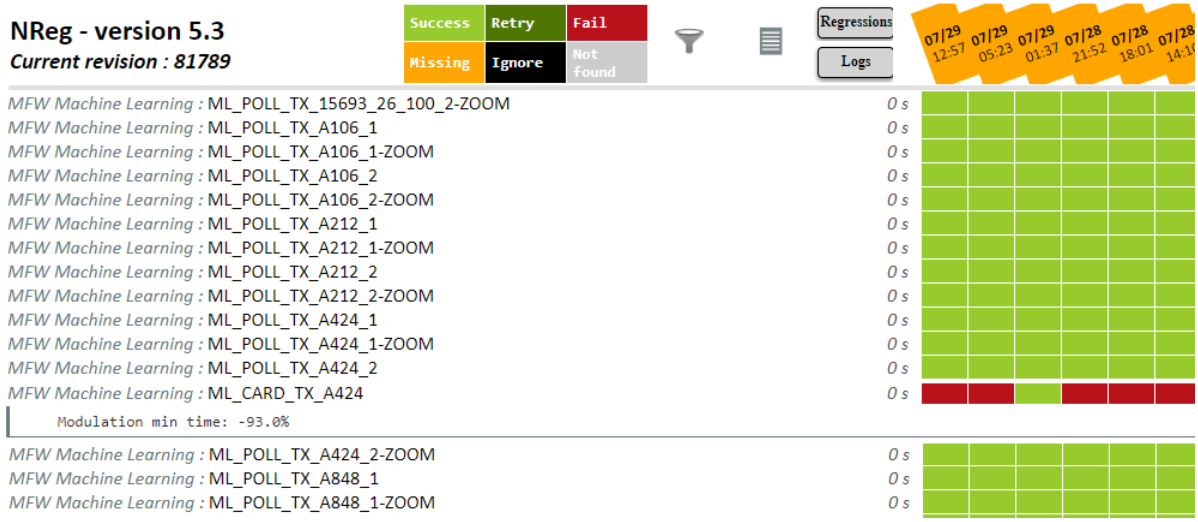


Figura 19 – Página da Web do servidor NXP com os resultados para testes de não regressão de vários dias. Retângulos verdes: Aprovado; Retângulos vermelhos: falha. As falhas da última execução têm uma descrição da principal característica que causou a falha.
Fonte: O autor (2019)

Sempre que uma anomalia (retângulo vermelho) for detectada, o nome do teste também terá uma breve descrição abaixo da causa principal da anomalia, ou seja, o recurso que teve maior variação, com base no valor médio extraído do modelo KNN.

4.3.1 Resultados obtidos

Com a finalização da aplicação, o antigo método utilizado pela equipe da NXP foi substituído. Esse método era baseado na análise gráfica entre um sinal antes e depois da atualização do *firmware*. Esta analise era obtida com a criação de uma máscara de testes em um osciloscópio.

Em linhas gerais, seu funcionamento começava pela criação de uma máscara em um osciloscópio usando um chip antes de atualizar seu firmware. Após a criação da máscara, o chip era atualizado e o mesmo sinal era gerado novamente. Caso o novo sinal não atravessasse os limiares definidos na máscara, ele era considerado normal, caso atravessasse, ele era considerado anômalo. Um exemplo de teste de máscara pode ser encontrado na Figura 20. Esta figura é só uma representação fornecida pelo fabricante, e não um teste real utilizado pela NXP.

Figura 20 – Exemplo de teste de máscara.
Fonte: https://triotest.com.au/keysight-streamline-usb-oscilloscopes/, acesso em 12/11/2019

Este método era pouco rigoroso, visto que era baseado em uma análise gráfica da forma do sinal e não continha nenhuma informação baseada nas especificações dos protocolos NFC, diferentemente do novo método proposto neste trabalho. Para a análise específica de pequenas partes de um sinal, este tipo de teste de máscara pode ser útil, mas para trechos de sinais RF com milhares de ciclos, como apresentado na Figura 15, esses testes não são robustos o suficiente.

Como comprovação de resultados, nos primeiros dias após a implementação do novo método, o antigo método continuou sendo utilizado, como forma de realizar um controle da nova performance do sistema.

Durante uma semana, houve por volta de 20 casos em que o novo sistema detectou erros em testes e o antigo sistema não os detectou. Esses erros foram causados por frequências parasitas, variações nos tempos de subida e descida do envelope e pequenos picos de tensão durante as rampas de subida e descida do sinal. Todos esses erros não foram detectados pelo sistema antigo, visto que em uma análise gráfica eles não eram perceptíveis.

Em comparação, todos os erros detectados pelo sistema antigo também foram detectados pelo novo método proposto.

Um dos erros mais detectados pelo novo sistema era a existência de níveis de ruído acima do esperado. Esses erros foram detectados através das características de gradiente existentes no sistema. Como esse erro não era importante, as características em gradiente foram abandonadas para grande maioria dos testes, reduzindo a quantidade de características de 42 para 32.

Este valor pode ser reduzido ainda mais se retirarmos as características ‘genéricas’ do sistema (reduzindo a quantidade para 22). Isto pode ser realizado caso as informações do sinal como um todo não sejam relevante para um teste, e somente as informações ligadas à especificação NFC estejam sendo avaliadas.

5 Conclusão e perspectivas

Este projeto tinha como objetivo a criação de um sistema de detecção de anomalias em sinais RF NFC que seria utilizado para melhorar o controle de qualidade dos sistemas de atualização de *firmware* de dispositivos NFC da empresa NXP *semiconductors*.

Para criar esse novo sistema, diversos métodos de detecção de anomalias que usam aprendizado de máquinas foram estudados e, de acordo com as especificações fornecidas pela NXP, o método dos K-vizinhos mais próximos foi escolhido como o mais adaptado.

Após a escolha do método de detecção de anomalias, uma etapa adicional foi necessária: a extração de características de um sinal RF NFC. O principal foco desta etapa era diminuir a quantidade de informações que o aplicativo final deveria fornecer ao algoritmo KNN. Para esta etapa, um estudo aprofundado das especificações dos protocolos NFC foi necessário.

A última etapa antes da criação do script final deste projeto foi a prova de conceito, aonde sinais RF NFC artificialmente anômalos foram utilizados para comprovar o bom funcionamento de todas as etapas anteriores, e também para validar a solução apresentada para a empresa.

Como uma análise geral da solução desenvolvida, o sistema pode detectar anomalias que antes seriam consideradas como um caso normal, como apresentado no capitulo 4.3.1, melhorando assim o controle de qualidade das atualizações de firmware do NXP. Com o uso diário do sistema, o próprio desempenho de detecção de anomalias deve se tornar cada vez mais robusto, devido ao aumento na quantidade de dados disponíveis para comparar uma nova amostra.

O *software* final foi implantado no sistema de teste de regressão NXP no mês de agosto. No momento em que este documento foi escrito, o *software* é responsável por verificar mais de 126 testes diferentes. Dado que os primeiros modelos KNN foram criados a partir de um pequeno banco de dados com aproximadamente 30 amostras de cada teste, espera-se que ocorram erros na detecção de anomalias (principalmente falso-negativos) que precisem ser corrigidos pelo usuário. Essas amostras devem ser adicionadas ao banco de dados para a próxima sessão de treinamento, a fim de corrigir a detecção desses falsos negativos.

5.1 Trabalhos futuros

Após a criação de um banco de dados grande o suficiente de sinais normais e anômalos, esse projeto pode ser continuado com o uso de algoritmos de aprendizado profundo para a criação de um sistema de detecção de anomalias ainda mais robusto, como as Redes Neurais de Longa Duração (LSTM), que podem ser usadas em dados de séries temporais para prever uma anomalia antes que ela aconteça, analisando as tendências ocultas no desempenho de um sistema.

Também é importante observar que, embora a implementação final baseada em KNN tenha sido usada para a detecção de anomalias em um sinal RF NFC, esse método pode ser generalizado para qualquer tipo de sinal e qualquer tipo de campo de aplicação. Um exemplo disto é o uso de uma abordagem semelhante, desenvolvida pelos engenheiros da NXP, para a detecção de anomalias usada no monitoramento das condições de máquinas na indústria [19].

Referências

[1] USMI JR. RFID Tag (Radio Frequency Identification Tag). Disponivel em: https://www.behance.net/gallery/67918923/RFID-Tag Acesso em 21 de outubro de 2019

[2] NFC FORUM. What is NFC?. Disponivel em: https://nfc-forum.org/what-is-nfc/ Acesso em 21 de outubro de 2019

[3] Ved, Mehul. Effective Outilier Detection techiniques in machine learning. Disponivel em: https://medium.com/@mehulved1503/effective-outlier-detection-techniques-in-machine-learning-ef609b6ade72 Acesso em 21 de outubro de 2019

[4] Detection d'anomalies dans une trame NFC par traitement de signal et machine learning, W. B. Pfeffer, O. Salim, Relatório de projeto industrial, ENSICAEN, 2019.

[5] Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain". Psychological Review. 65 (6): 386–408.

[6] Python. Disponivel em: https://www.python.org/ Acesso em 21 de outubro de 2019

[7] Scikit-learn: Machine learning in Python. Disponivel em: https://scikit-learn.org/stable/ Acesso em 21 de outubro de 2019

[8] Tensorflow: An end-to-end open source machine learning plataform. Disponivel em: https://www.tensorflow.org/ Acesso em 21 de outubro de 2019

[9] Ved, Mehul. Feature selection and feature extraction in machine learning: An overview. Disponivel em: https://medium.com/@mehulved1503/feature-selection-and-feature-extraction-in-machine-learning-an-overview-57891c595e96 Acesso em 23 de outubro de 2019

[10] International Organization for Standardization. ISO/IEC 14443 Identification cards -- Contactless integrated circuit cards -- Proximity cards. 2018

[11] RF Wireless World. NFC-A vs NFC-B vs NFC-F. Disponivel em: http://www.rfwireless-world.com/Terminology/NFC-A-vs-NFC-B-vs-NFC-F.html Acesso em 24 de outubro de 2019

[12] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In Parallel Distributed Processing. Vol 1: Foundations. MIT Press, Cambridge, MA, 1986.

[13] Rosenblatt, Frank (1957), The Perceptron--a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory

[14] GADAT, Sébastien. Algorithmes de Support Vector Machines : séance 12. Laboratoire de Statistique et Probabilités.

[15] Python Machine Learning Tutorial. Bernd Klein, Bodenseo, 2011 – 2018

[16] Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

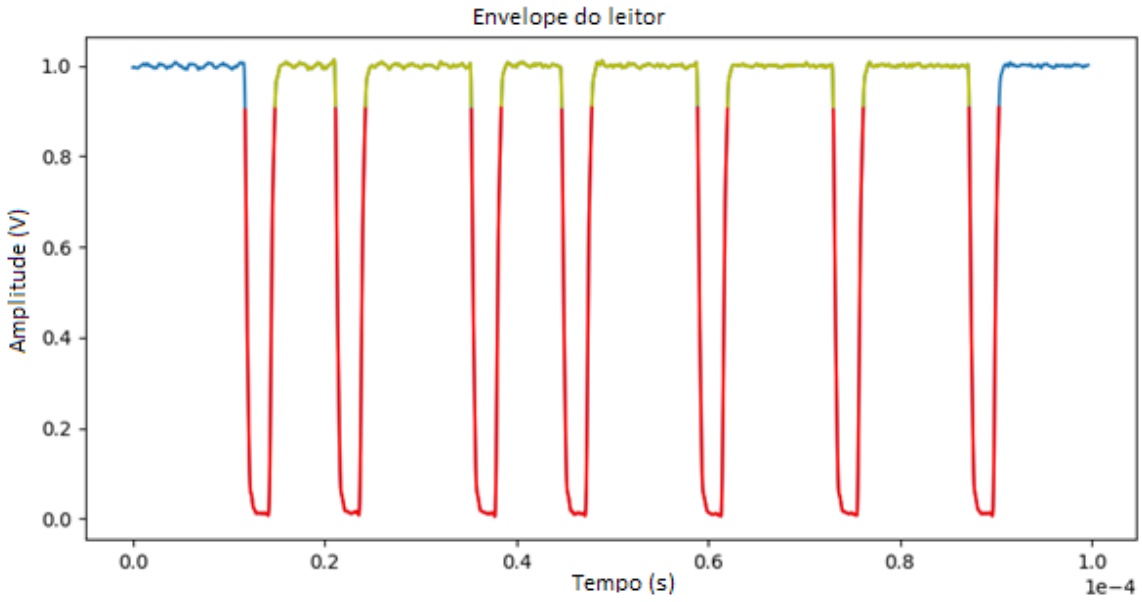
[17] Yiu, Tony. The curse of dimensionality. Disponivel em: https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e Acesso em 24 de outubro de 2019

[18] Anton, Howard (1994), *Elementary Linear Algebra* (7th ed.), John Wiley & Sons, pp. 170–171

[19] IoT Edge NXP. Anomaly detection.Disponivel em: https://iotedge.nxp.com/anomaly-detection/ Acesso em 24 de outubro de 2019

Anexo A – Descrição dos diferentes tipos de características utilizados para a extração de características

Para facilitar a visualização de como foi feita a extração de características de um sinal NFC RF, todas as explicações utilizaram como base a figura abaixo.



Esta figura contém o envelope de um sinal RF NFC-A de 106 kbps normalizado e já dividido em suas regiões de interesse: nível alto (amarelo) e nível baixo (vermelho), bem como a região do sinal que não será analisada (azul).

A extração de características pode ser dividida em dois tipos diferentes de características:

A. **Características que são baseadas em especificações de sinais RF NFC que são descritas pelas normas NFC regentes.**
B. **Características genéricas que são normalmente utilizadas em algoritmos de aprendizado de máquinas. Mais precisamente, todas essas características foram baseadas em aplicações já existentes criadas por diferentes equipes da NXP.'**

Para cada característica será fornecido uma breve explicação sobre ela, como ela é calculada e, se existente, a que parte da especificação NFC ela pode ser correlacionada.

A – Foram definidas 22 características baseadas nas especificações NFC:

- *Tempo médio de modulação* - O tempo médio que um nível baixo de um sinal leva do início ao fim. Calculado a partir do valor médio do número de pontos de cada segmento vermelho da figura, e depois dividindo esse valor pela frequência de amostragem. Correlaciona a: Taxa de bits, Tempo de subida e Tempo de queda.
- *Tempo máximo da modulação* - o tempo máximo que um nível baixo de um sinal leva do início ao fim. Calculado a partir do valor máximo do número de pontos de cada segmento vermelho da figura, e depois dividindo esse valor pela frequência de amostragem. Correlaciona a: Taxa de bits, Tempo de subida e Tempo de queda.
- *Tempo mínimo da modulação* - o tempo mínimo que um nível baixo de um sinal leva do início ao fim. Calculado a partir do valor mínimo do número de pontos de cada segmento vermelho da figura, e depois dividindo esse valor pela frequência de amostragem. Correlaciona a: Taxa de bits, Tempo de subida e Tempo de queda.
- *Tempo médio curto entre modulação* - O tempo médio entre dois níveis baixos consecutivas de curta duração (para ser considerada uma curta duração, ela deve ser no máximo 3 vezes maior que a duração mais curta entre dois estados de nível baixo). Calculado a partir do valor médio do número de pontos de cada segmento amarelo curto da figura, e depois dividindo esse valor pela frequência de amostragem. Correlaciona a: Taxa de bits.
- *Tempo máximo curto entre modulação* - O tempo máximo entre dois níveis baixos consecutivas de curta duração. Calculado a partir do valor máximo do número de pontos de cada segmento amarelo curto da figura, e depois dividindo esse valor pela frequência de amostragem. Correlaciona a: Taxa de bits.
- *Tempo mínimo curto entre modulação* - O tempo máximo entre dois níveis baixos consecutivas de curta duração. Calculado a partir do valor mínimo do número de pontos de cada segmento amarelo curto da figura, e depois dividindo esse valor pela frequência de amostragem. Correlaciona a: Taxa de bits.

- *Desvio padrão do tempo curto entre modulação* – O desvio padrão do tempo entre dois níveis baixos consecutivos de curta duração. Calculado a partir do desvio padrão do valor do número de pontos de cada segmento amarelo curto da figura divididos pela frequência de amostragem. Correlacionada com: informações binárias do sinal.
- *Tempo médio longo entre modulação* - O tempo médio entre dois níveis baixos consecutivas de longa duração (para ser considerada uma longa duração, ela deve ser no mínimo 3 vezes maior que a duração mais curta entre dois estados de nível baixo. É possível que não exista um tempo longo para um sinal RF NFC). Calculado a partir do valor médio do número de pontos de cada segmento amarelo longo da figura, e depois dividindo esse valor pela frequência de amostragem. Correlacionada com: taxa de bits.
- *Tempo máximo longo entre modulação* - O tempo máximo entre dois níveis baixos consecutivas de longa duração. Calculado a partir do valor máximo do número de pontos de cada segmento amarelo longo da figura, e depois dividindo esse valor pela frequência de amostragem. Correlaciona a: Taxa de bits.
- *Tempo mínimo longo entre modulação* - O tempo mínimo entre dois níveis baixos consecutivas de curta duração. Calculado a partir do valor mínimo do número de pontos de cada segmento amarelo da figura, e depois dividindo esse valor pela frequência de amostragem. Correlaciona a: Taxa de bits.
- *Desvio padrão do tempo longo entre modulação* – O desvio padrão do tempo entre dois níveis baixos consecutivos de longa duração. Calculado a partir do desvio padrão do valor do número de pontos de cada segmento amarelo longo da figura divididos pela frequência de amostragem. Correlaciona com: Informações binárias do sinal.
- *Média CC* - A tensão média do nível alto do sinal. Calculada a partir do valor médio de todos os pontos dos seguimentos amarelos da figura. Correlaciona a: Potência média do sinal.
- *Máximo CC* - A tensão máxima do nível alto do sinal. Calculada a partir do valor máximo de todos os pontos dos seguimentos amarelos da figura. Correlaciona a: *Overshoot* e *Undershoot*.

- *Mínimo CC* - A tensão mínima do nível alto do sinal. Calculada a partir do valor mínimo de todos os pontos dos seguimentos amarelos da figura. Correlaciona a: *Overshoot* e *Undershoot*.
- *Desvio padrão CC* – O desvio padrão da tensão do nível alto do sinal. Calculada a partir do desvio padrão do valor de todos os pontos dos seguimentos amarelos da figura. Correlaciona a: *Overshoot* e *Undershoot*.
- *Média Modulação* - A tensão média do nível baixo do sinal. Calculada a partir do valor médio de todos os pontos dos seguimentos vermelhos da figura. Correlaciona a: Forma da modulação, tempo de queda e tempo de subida.
- *Máximo Modulação* - A tensão máxima do nível baixo do sinal. Calculada a partir do valor máximo de todos os pontos dos seguimentos vermelhos da figura. Correlaciona a: coeficiente de modulação.
- *Mínimo Modulação* - A tensão mínima do nível baixo do sinal. Calculada a partir do valor mínimo de todos os pontos dos seguimentos vermelhos da figura. Correlaciona a: coeficiente de modulação.
- *Desvio padrão Modulação* – O desvio padrão da tensão do nível baixo do sinal. Calculada a partir do desvio padrão do valor de todos os pontos dos seguimentos vermelhos da figura. Correlaciona a: ruído e formações anômalas no sinal.
- *Tempo de comando* - O tempo total do sinal de informação do sinal NFC RF. Calculada a partir do tempo decorrido entre o primeiro e o ultimo seguimento vermelho. Correlaciona para: Taxa de bits e número de bits da mensagem.
- *Frequência de pico* - A frequência do pico principal da transformada de Fourier do envelope do sinal NFC. Corresponde a: Taxa de bits, Tempo de subida e Tempo de queda.
- *Valor de frequência de pico* - O valor de amplitude do pico principal da transformada de Fourier do envelope do sinal NFC. Corresponde a: Taxa de bits, Tempo de subida e Tempo de queda.

B – Foram definidas 10 características diferentes baseados em algoritmos clássicos de agrupamento. Essas características do tipo B não têm correlação direta com as especificações da NFC:

- *Média* – O valor médio entre todos os pontos dos seguimentos vermelhos e amarelos

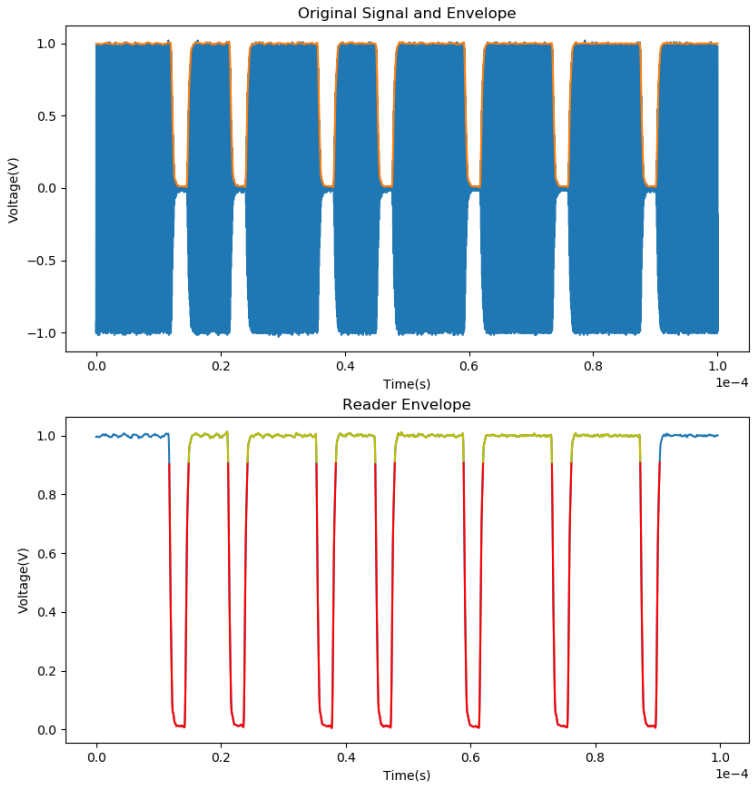
- *Variância* – A variância entre todos os pontos dos seguimentos vermelhos e amarelos
- *Desvio padrão* – O desvio padrão entre todos os pontos dos seguimentos vermelhos e amarelos.
- *Skewness* – Um valor que corresponde ao quanto a distribuição de probabilidade dos valores do envelope do sinal NFC se assemelha a uma distribuição gaussiana.
- *Kurtosis* – Uma medida da "cauda" da distribuição de probabilidade do envelope do sinal NFC.
- *Valor mínimo* - O valor mínimo do sinal (segmentos vermelhos e amarelos) após subtrair dele seu valor médio.
- *Valor máximo* - O valor máximo do sinal (segmentos vermelhos e amarelos) após subtrair dele seu valor médio.
- *Valor mínimo absoluto* - O valor mínimo absoluto do sinal (segmentos vermelhos e amarelos) após subtrair dele seu valor médio.
- *Valor máximo absoluto* - O valor máximo absoluto do sinal (segmentos vermelhos e amarelos) após subtrair dele seu valor médio.
- *Fator de crista* - Uma medida de quanto o sinal (segmentos vermelhos e amarelos) se assemelha a uma onda sinusoidal.

Aplicamos a extração de características do tipo B em todo o sinal do envelope e em seu gradiente.

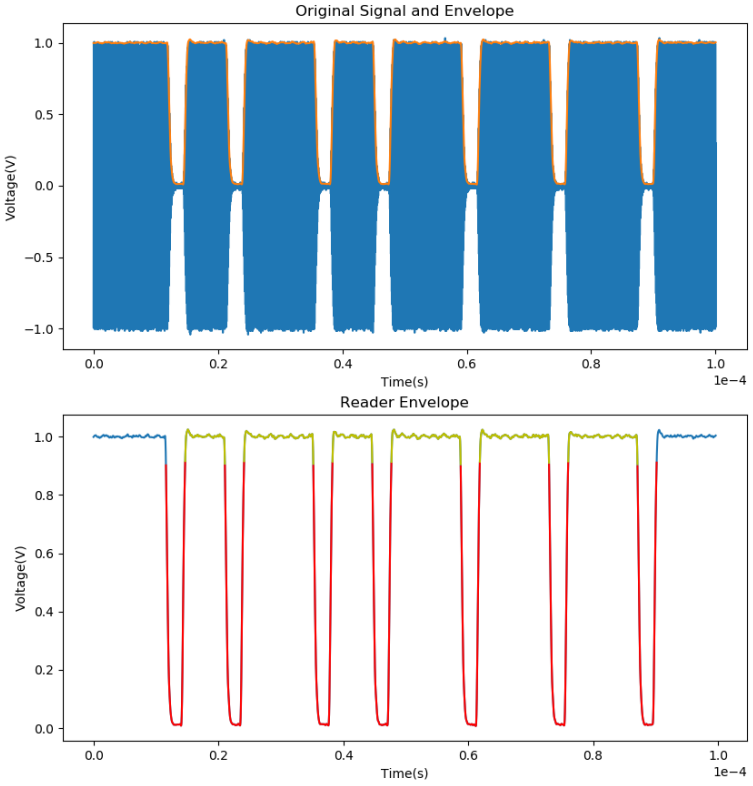
Esse gradiente é calculado a partir da derivada temporal do envelope do sinal NFC. O ponto interessante deste processo é que variações rápidas em tensão como picos anômalos de tensão ou fortes ruídos são postos em evidencia ao aplicar a derivada temporal, logo, este tipo de anomalia é mais facilmente detectada nas características que foram extraídas do gradiente do sinal.

Anexo B – Comparação dos resultados da prova de conceito

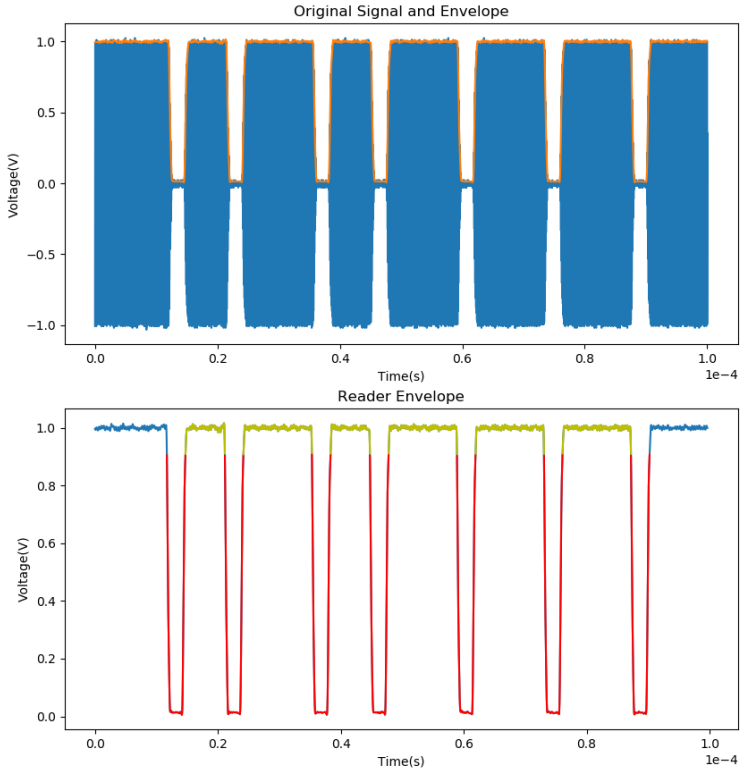
Sinal NFC-A *leitor* a 106kbps original



Sinal com tensão pico de 0,88 V, considerado como anômalo



Sinal que foi adicionado com uma onda senoidal parasita de frequência de 11 MHz e 15 mV de tensão pico a pico, também considerado como anômalo



Sinal após ser a acoplagem da antena da placa de demonstração PN547 (percebe-se a criação de um *overshoot* no sinal), também considerado como anômalo