UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA
ENGENHARIA DE TELECOMUNICAÇÕES

O USO DE REDES NEURAIS PARA PREDIÇÃO DE FEIXES

Bruno Marvin Rodrigues do Nascimento
Daniel Victor Carvalho de Oliveira
Ezequiel Alves Cardoso
Gefferson Mateus da Rocha Simão
Janathan Junior Planas Pena

Fortaleza-CE
2023

CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE
TELEINFORMÁTICA
ENGENHARIA DE TELECOMUNICAÇÕES

O USO DE REDES NEURAIS PARA PREDIÇÃO DE FEIXES

Bruno Marvin Rodrigues do Nascimento
Daniel Victor Carvalho de Oliveira
Ezequiel Alves Cardoso
Gefferson Mateus da Rocha Simão
Janathan Junior Planas Pena

Relatório de Ações Integradas de Ciência e Tecnologia apresentado ao prof. Yuri Silva como requisito avaliativo da disciplina.

Fortaleza-CE
2023

Sumário

Lista de ilustrações

Lista de tabelas

Nomenclature

$RTL-SDR$ Circuito Integrado RTL2832U

SDR Software Defined Radios ou Rádios Definidos por Software

$RMSE$ Métrica de avaliação de erro - Raiz da média do erro do quadrático

FFT Traduzido do inglês, Fast Fourier Transform, isto é, Transformada rápida de Fourier

KNN Método de Machine Learning K-Nearest Neighbors

MAE Métrica de avaliação de erro - Média do erro Absoluto

MSE Métrica de avaliação de erro - Média do erro do quadrático

Resumo

O objetivo deste trabalho é realizar a classificação de três tipos de dados no espectro: clear, LTE e WiFi, utilizando redes neurais do tipo CNN, ResNet, RNN e MLP. A classificação de dados no espectro é uma tarefa fundamental para a otimização e alocação eficiente dos recursos de comunicação sem fio.

Palavras-chave: Classificação; Wifi; Clear; LTE; Redes neurais.

Abstract

The objective of this work is to perform the classification of three types of data in the spectrum: clear, LTE and WiFi, using neural networks like CNN, ResNet, RNN, ResNet and MLP. The classification of data in the spectrum is a fundamental task for the optimization and efficient allocation of wireless communication resources.

Keywords: Classification; WiFi; Clear; LTE; Neural networks.

Estrutura do trabalho

O trabalho a seguir estará dividido em:

- Introdução: Contextualização do trabalho, motivação e objetivos;
- Metodologia: Onde iremos descrever o processo realizado para chegarmos aos resultados desejados, bem como a análise e descrição dos algorítmos abordados e discussão sobre os resultados;
- Conclusão e perpectivas;

1 Introdução

1.1 Contextualização

As redes neurais desempenham um papel fundamental no processo de classificação de dados em diversas áreas, desde a visão computacional até a análise de dados complexos. Com sua capacidade de aprendizado automático e capacidade de lidar com grande quantidade de informações, as redes neurais se tornaram uma ferramenta essencial para lidar com a crescente demanda por classificação precisa e eficiente.

A importância das redes neurais no processo de classificação de dados está intrinsecamente ligada à sua capacidade de simular o funcionamento do cérebro humano. Inspiradas na estrutura do sistema nervoso, as redes neurais são compostas por camadas de neurônios artificiais interconectados, que processam informações e realizam cálculos para identificar padrões e relações nos dados.

1.2 Motivação

Ao serem treinadas com conjuntos de dados de treinamento, as redes neurais são capazes de aprender a reconhecer padrões complexos e extrair características relevantes dos dados. Elas ajustam seus pesos e conexões internas de forma iterativa para minimizar o erro entre as saídas previstas e as saídas desejadas. Essa capacidade de aprendizado permite que as redes neurais se adaptem a diferentes tipos de dados e problemas de classificação, tornando-as extremamente versáteis.

Além disso, as redes neurais possuem uma capacidade inata de lidar com dados não-lineares e de alta dimensionalidade. Em comparação com outros métodos de classificação, como as abordagens estatísticas tradicionais, as redes neurais podem capturar relações complexas entre os atributos dos dados, o que é especialmente valioso em problemas de classificação mais desafiadores.

A aplicação de redes neurais na classificação de dados tem tido um impacto significativo em várias áreas, como reconhecimento de padrões, processamento de imagens, processamento de linguagem natural, medicina, finanças e muito mais. Elas têm sido utilizadas para identificar objetos em imagens, detectar fraudes em transações financeiras, analisar sentimentos em textos e prever diagnósticos médicos com alta precisão.

Motivados por esse contexto, nota-se que é muito valioso a utilização das redes neurais para classificarmos diferentes tipos de dados, e no nosso contexto classificarmos dados de espectro e diferenciar os tipos de sinais (ruido, rede WiFi e rede LTE).

1.3 Objetivos

A meta deste trabalho é trazer diversas formas e ferramentas de machine learning através da utilização de redes neurais e algoritmos de treinamento, além de claro mostrar situações práticas.

Uma vez que as redes neurais possuem fácil implementação e baixo custo, e os métodos de treinamento abordados são puramente executados em softaware, não há custos exarcerbados para que organizações empresariais que necessitem de serviços ou tecnologias que utilizam uma ou mais frequências sem que haja interferência, possam operar com segurança na transmissão de dados.

2 Metodologia e Resultados

Neste tópico abordaremos:

- Processo de extração de dados;
- Caracterização de cada método utilizado para treinamento dos conjuntos de dados;
- Aspectos de implementação dos métodos;
- Discussão dos resultados gráficos e métricos obtidos.

2.1 MLP - Multilayer Perceptron

2.1.1 Definição

Podemos definir o perceptron de múltiplas camadas (MLP) como sendo uma rede neural contendo uma ou mais camadas ocultas, com uma quantidade indeterminada de neurônios. Suas principais características são o mapeamento não linear entre entrada e saída, e seu principal algoritmo de treinamento é o algoritmo de retropropagação (backpropagation).

2.1.2 Estrutura

O MLP pode ser segregado em 3 principais partes.

1. Camada de Entrada (input layer);
    - Na camada de entrada estão os valores dos dados de entrada que serão utilizados.
2. Camadas Ocultas (hidden layers);
    - Nas camadas ocultas os dados serão processados de forma a encontrarmos pesos que minimizem o erro na camada de saída. O número de camadas ocultas, bem como o número de neurônios em cada camada oculta deve ser estipulado pelo analista que está a frente do treinamento da rede neural.
3. Camada de Saída (output layer);
    - Na camada de saída teremos uma quantidade de neuronios que serão determinados a partir do resultado almejado do problema. Para problemas de classificação são utilizados apenas um neurônio na camada de saída, a fim de assumir os valores de classificação.

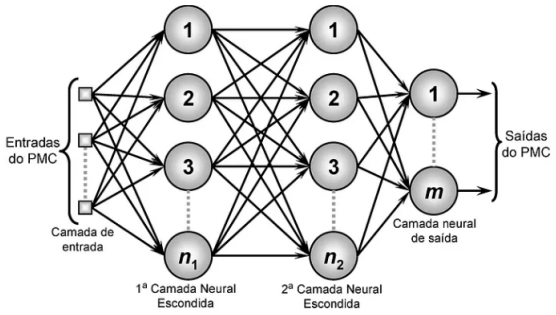


Figura 1 – Perceptron de múltiplas camadas

2.1.3 Algoritmo de backpropagation

O algoritimo de retropropagação (backpropagation) é a técnica de treinamento mais utilizada nas redes MLP. O algoritmo consiste na minimização do erro médio quadrado, encontrando pesos otimizados em cada neurônio a partir do fluxo de ida (foward) e volta (backward) dos dados pelos neurônios e consequentemente pelas funções de ativação.

Uma vez que os dados de entrada entram na primeira camada oculta em cada um dos neurônios, estes geram saídas individuais que a posteriore será entrada nos neurônios da camada oculta seguinte, e assim por diante até chegar a camada de saída, onde será calculado o erro. Este processo torna o MLP uma rede com alto grau de conectividade entre camadas, portanto, cada ajuste em parâmetros pode causar mudanças significativas na rede.

Tendo em vista estes aspectos abordados acima, percebe-se que a manipulação da rede deve ser feita de forma cautelosa. Normalmente não há especificações para estipular os valores da rede, sendo estes valores estimados empiricamente, através da experiencia do especialista que está supervisionando a rede. Um fator importante, por exemplo, é o número de camadas ocultas e de neurônios em cada camada, uma vez que a quantidade de neurônios influência diretamente na capacidade da rede resolver problemas mais complexos. Caso o número de camadas e neurônios seja aumentado a rede terá uma maior capacidade de extração de dados e conhecimentos do problema que está sendo abordado, porém caso o número de neurônios supere muito uma quantidade ótima, a rede pode não ser capaz de aprender/prever novos padrões, ocorrendo assim um caso de overfitting. Por outro lado uma rede com um número insuficiente de neurônios não é capaz de modelar precisamente o problema.

O algoritmo de aprendizado de retropropagação da MLP pode ser dividido em quatro partes:

1. Inicialização do algoritmo
    - Determinar valores para os limites e pesos;
    - Determinar os parâmetros iniciais da rede (inicialmente adotar valores aleatórios para teste);
2. Camada de ativação
    - Determinar a quantidade de neurônios na camada oculta;
    - Determinar a quantidade de neurônios na camada de saída;
3. Treinamento
    - Calcular os erros dos neurônios nas camadas citadas:
    - Calcular a correção dos pesos dos neurônios;
    - Atualizar os pesos dos neurônios nas camadas;

4. Repetição

    - Repetir todo o processo a partir do segundo passo até o algoritmo convergir.

2.1.4 Aplicação da rede ao problema

Como dito em parágrafos anteriores a problemática consistia em construir uma rede neural que fosse capaz de separar dados no espectro de três tipos diferentes, Clear, Wifi e LTE.

Com a ajuda da biblioteca **SCIKITLEARN** em python fomos capazes de utilizar a MLP-classifier para este caso.

2.1.4.1 Etapas da implementação

- Inicialmente carregamos os datasets com os dados do LTE, WIFI e Clear;
- Separamos as partes reais e imaginárias de cada sinal;
- Associamos a cada tipo de sinal um feature (uma classificação), como mostrado na tabela a seguir:

CLEAR	0
LTE	1
WIFI	2

Tabela 1 – Classificação dos tipos de dados

- Juntamos todos os dados e em seguida fizemos a permutação das linhas, de modo que os tipos de sinais ficassem misturados em linhas;
- Armazenamos os dados em formato de matriz de modo que a matriz de entrada (X), contivesse duas colunas, sendo uma a parte real do sinal e a outra a parte imaginária, e uma matriz de saída (Y) que contivesse apenas uma coluna com os valores das classificações de acordo com a tabela anteriormente mostrada;
- Com as matrizes de entrada e saída prontas, fomos capazes de separar os dados de teste e treino;
- Após a normalização dos dados, os datasets estavam prontos para serem aplicados a rede MLPClassifier;
- Em seguida, definimos alguns parâmetros - chave para o processamento da rede, sendo esses parâmetros descritos a seguir.

2.1.4.2 Parâmetros da rede MLPClassifier

A rede MLPClassifier possui diversos parâmetros que podemos ser alterados de forma a reduzir a otimizar a classificação dos dados, sendo alguns deles citados a seguir:

- **hidden layer size**: Número de camadas ocultas e quantidade de neurônios em cada camada;
- **activation**: Função de ativação utilizada na rede. A função de ativação esta presente em cada neurônio das camadas ocultas ou camada de saída, trazendo uma não-lineareadade para as saída dos dados, caso não houvesse uma função de ativação, os dados de saída seriam apenas uma combinação linear dos dados de entrada, limitando a capacidade de aprendizado da rede;
- **solver**: Responsável pela otimização dos pesos sinapticos dos neurônios;
- **alpha**: Taxa de aprendizado. Determina a que passo o otimizador irá ajustar os pesos da rede;
- **learning rate**: No MLPClassifier, o learning rate diz respeito a forma que a taxa de aprendizado irá se comportar de acordo com a resposta de aprendizado da rede, podendo ser constante ou adaptável;
- **early stopping**: Quando ativado faz com que a rede pare de executar iterações quando as dez iterações anteriores não tiverem mudado significativamente a perda.

Foram utilizados os seguintes parâmetros na rede (os que não forem citados, suponha que esteja no default da configuração da rede):

Números e tamanho das camadas ocultas	400,200,100,20
Função de ativação	ReLu
Otimizador	SGD
Taxa de aprendizagem	Default = 0.001
Número de iterações	100
Parada antecipada	True
Verbose	True
Tipo de taxa de aprendizagem	Adaptive
Shuffle	True

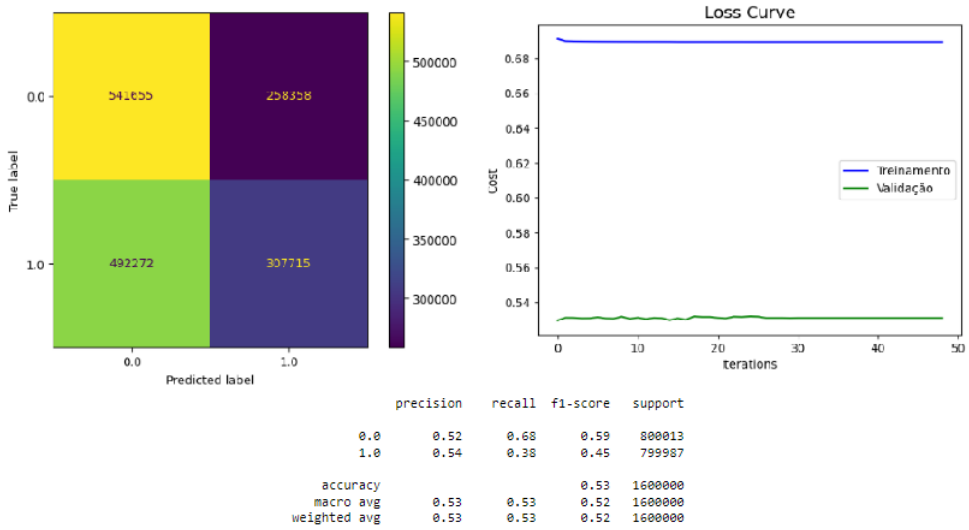
Tabela 2 – Configuração dos parâmetros da rede

2.1.4.3 Resultados e discussões

Realizamos o treinamento dos datasets de duas formas: comparando de dois em dois os datasets (CLEAR X WIFI, CLEAR X LTE e WIFI X LTE) e comparando os três datasets diretamente (CLEAR X WIFI X LTE). Para os dois modos obtemos resultados diferentes, além de o tempo para execução do script ser bem maior no modo direto. Os resultados estão registrados a seguir:

Comparando de dois em dois:

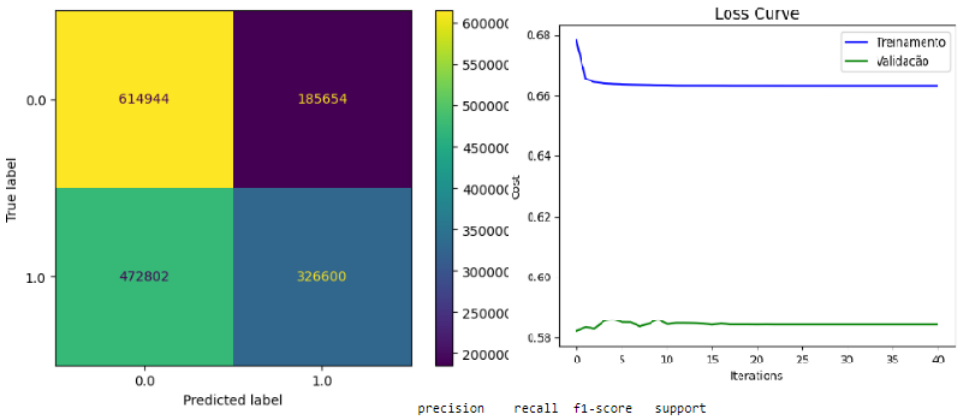
CLEAR X LTE



	precision	recall	f1-score	support
0.0	0.52	0.68	0.59	800013
1.0	0.54	0.38	0.45	799987
accuracy			0.53	1600000
macro avg	0.53	0.53	0.52	1600000
weighted avg	0.53	0.53	0.52	1600000

Figura 2 – Resultados para CLEAR e LTE

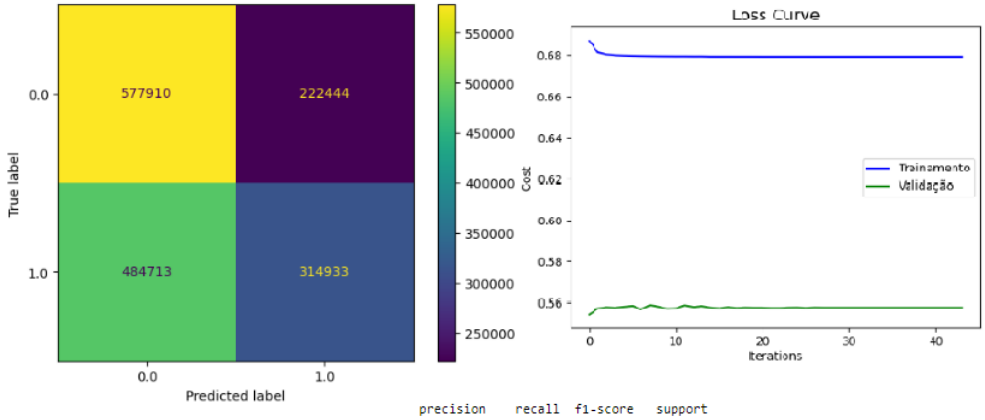
CLEAR X WIFI



	precision	recall	f1-score	support
0.0	0.57	0.77	0.65	800598
1.0	0.64	0.41	0.50	799402
accuracy			0.59	1600000
macro avg	0.60	0.59	0.57	1600000
weighted avg	0.60	0.59	0.57	1600000

Figura 3 – Resultados para CLEAR e WIFI

LTE X WIFI



	precision	recall	f1-score	support
0.0	0.54	0.72	0.62	800354
1.0	0.59	0.39	0.47	799646
accuracy			0.56	1600000
macro avg	0.56	0.56	0.55	1600000
weighted avg	0.56	0.56	0.55	1600000

Figura 4 – Resultados para LTE e WIFI

Comparando diretamente:

CLEAR X WIFI X LTE

ACURÁCIA

```
y_pred = mlp_clf.predict(testX_scaled)
print('Accuracy: {:.2f}'.format(accuracy_score(Y_test, y_pred)))
```

```
Accuracy: 0.59
              precision    recall  f1-score   support

         0.0       0.50      0.65      0.57    800205
         1.0       0.65      0.43      0.52    799939
         2.0       0.65      0.68      0.66    799856

    accuracy                           0.59   2400000
   macro avg       0.60      0.59      0.58   2400000
weighted avg       0.60      0.59      0.58   2400000
```

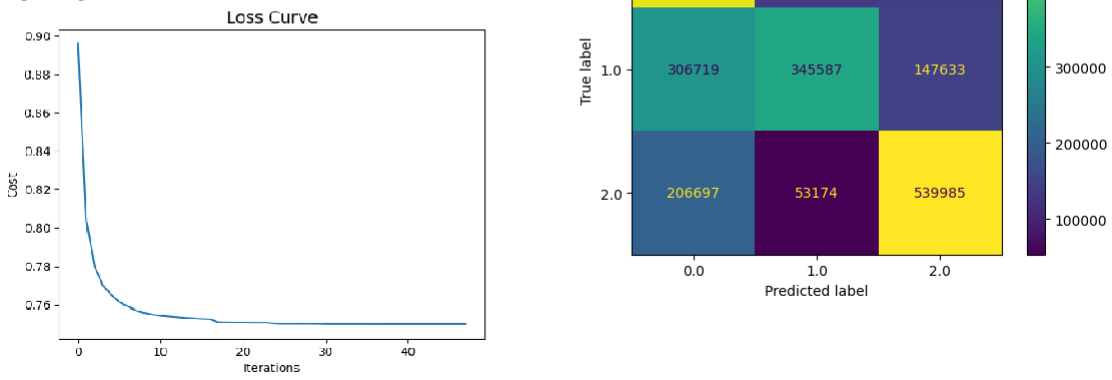


Figura 5 – Resultados para CLEAR, WIFI E LTE

Observações e comentários:

- Utilizamos um tamanho de 1.000.000 para cada dataset;
- A maior acurácia foi de 59%;
- O fato de não termos conseguido uma acurácia maior pode ser devido ao modelo escolhido (MLP) não ser adequado para nosso conjunto de dados;

- Tentamos implementar fft mas não obtivemos melhores resultados;
- A medida que testávamos os parâmetros da rede notamos que quanto maior era o número de camadas e neurônios, maior era a capacidade de aprendizagem da rede, aumentando a acurácia e diminuindo a perda. Porém, a medida que a complexidade das camadas aumentavam o tempo para a rede convergir ou executar as iterações estipuladas aumentavam muito. Sendo necessário um suporte muito melhor de hardware.
- Uma vez que a rede MLP chegou a uma acurácia de 0.59 com um número consideravelmente grande de amostras e camadas, e com um tempo de treinamento muito longo, chega-se a uma conclusão de inviabilidade parcial no uso da rede, onde a rede muito provavelmente seria capaz de predizer precisamente a classificação dos dados, porém a um custo muito grande, ou seja, um grande número de camadas e interações, exigindo um esforço computacional que muitas vezes pode estar indisponível.

2.2 CNN - Convolution Neural Network

2.2.1 Blockage-Prediction-Using-Wireless-Signatures

A classificação de espectro de sinal é uma tarefa essencial em várias áreas, como processamento de sinais, telecomunicações e sensoriamento remoto. A utilização de redes neurais convolucionais (CNNs) para essa tarefa tem se mostrado bastante promissora devido à capacidade das CNNs em capturar características espaciais relevantes em sinais de natureza temporal, como os espectros de sinal. As CNNs são capazes de extrair automaticamente características discriminativas do espectro de sinal, como padrões de frequência e modulações, por meio de camadas convolucionais e de pooling, seguidas por camadas totalmente conectadas para classificação.(GOODFELLOW et al., 2016)

Ao utilizar CNNs para a classificação de espectro de sinal, é possível obter resultados mais precisos e robustos em comparação com abordagens tradicionais. A capacidade das CNNs de aprender e adaptar-se aos padrões presentes nos espectros de sinal, combinada com a sua habilidade em lidar com dados complexos e de alta dimensionalidade, torna essa abordagem particularmente adequada para aplicações de classificação de espectro. Além disso, a utilização de CNNs oferece a vantagem de ser capaz de lidar com espectros de sinal de diferentes tamanhos e resoluções, proporcionando flexibilidade no processamento de diferentes tipos de sinais.(RASCHKA, 2015)

2.2.2 Breve Introdução ao CNN

As redes neurais convolucionais (CNNs) são uma classe de redes neurais artificiais que são particularmente bem-sucedidas no processamento de dados de imagens. Elas são capazes de aprender padrões complexos em dados, mesmo quando os dados são ruidosos ou incompletos.

As CNNs são usadas em uma ampla gama de aplicações, incluindo reconhecimento de imagens, detecção de objetos, processamento de linguagem natural e visão computacional.

Funcionamento

As CNNs são compostas por uma série de camadas, cada uma das quais realiza uma operação diferente. As camadas mais comuns em uma CNN são as seguintes:

- Camada de entrada: Esta camada recebe os dados de entrada.
- Camada de convolução: Esta camada aplica uma função de convolução aos dados de entrada. A função de convolução é uma operação matemática que extrai características locais dos dados.
- Camada de pooling: Esta camada reduz o tamanho dos dados de saída da camada de convolução. Isso ajuda a reduzir o número de parâmetros da rede e a acelerar o treinamento.
- Camada de ativação: Esta camada aplica uma função de ativação aos dados de saída da camada de pooling. A função de ativação é uma operação matemática que ajuda a melhorar a aprendizagem da rede.
- Camada de classificação: Esta camada classifica os dados de saída da rede.

Valor

As CNNs são valiosas para uma ampla gama de aplicações porque são capazes de aprender padrões complexos em dados, mesmo quando os dados são ruidosos ou incompletos. Isso as torna particularmente adequadas para aplicações em que os dados são difíceis de obter ou processar.

Blockage-Prediction-Using-Wireless-Signatures

As CNNs podem ser usadas para prever bloqueios em redes sem fio usando assinaturas sem fio. As assinaturas sem fio são características estatísticas que podem ser extraídas de sinais sem fio. Elas podem ser usadas para identificar diferentes tipos de tráfego, bem como para detectar a presença de obstáculos.

Uma abordagem típica para usar CNNs para prever bloqueios em redes sem fio envolve os seguintes passos:

1. Coleta de dados: Os dados de treinamento são coletados de uma rede sem fio real. Os dados incluem assinaturas sem fio capturadas em diferentes condições de tráfego e com diferentes tipos de obstáculos.
2. Pré-processamento dos dados: Os dados de treinamento são pré-processados para remover ruído e normalizar os dados.
3. Treinamento da CNN: A CNN é treinada usando os dados de treinamento pré-processados.

4. Teste da CNN: A CNN é testada usando dados de teste coletados nas mesmas condições que os dados de treinamento.

Os resultados de experimentos mostram que as CNNs são capazes de prever bloqueios em redes sem fio com alta precisão. Isso pode ajudar a melhorar a confiabilidade e o desempenho das redes sem fio.

Outros valores

As CNNs podem ser usadas para uma ampla gama de outras aplicações, incluindo:

- Reconhecimento de imagens: As CNNs podem ser usadas para identificar objetos, faces e outros objetos em imagens.
- Detecção de objetos: As CNNs podem ser usadas para detectar objetos em imagens ou vídeos.
- Processamento de linguagem natural: As CNNs podem ser usadas para entender o significado de texto.
- Visão computacional: As CNNs podem ser usadas para realizar uma ampla gama de tarefas de visão computacional, como detecção de objetos, reconhecimento de faces e rastreamento de objetos.

As CNNs são uma ferramenta poderosa que pode ser usada para resolver uma ampla gama de problemas. Elas são especialmente valiosas para aplicações em que os dados são difíceis de obter ou processar.

2.2.3 Contextualizando Predição de Bloqueio usando sinais sem fio

O bloqueio de conexões de linha de visão (LOS) é um desafio: Comunicações de ondas milimétricas (mmWave) e sub-terahertz estão se tornando direções dominantes para redes sem fio modernas e futuras. Apesar de possuírem grande largura de banda para atender demandas de alta taxa de dados em aplicações como Realidade Virtual/Aumentada (VR/AR) e direção autônoma, essas bandas enfrentam desafios devido à sensibilidade de propagação do sinal a bloqueios. Manter conexões LOS entre estações base e usuários é crucial, pois bloqueios estáticos ou dinâmicos podem afetar significativamente a confiabilidade e latência dos sistemas mmWave/terahertz, dificultando o suporte a aplicações altamente móveis e sensíveis à latência.

Prever bloqueios com auxílio de sensores é uma solução promissora: Superar os desafios de bloqueio de conexão depende da capacidade de compreender o ambiente ao redor. Como os sistemas de comunicação mmWave/sub-THz dependem de conexões de linha de visão entre transmissor/receptor, ter consciência de suas localizações e do ambiente ao redor (geometria dos prédios, objetos em movimento, etc.) pode ajudar na previsão de futuros bloqueios. Por exemplo, dados coletados de câmeras RGB, LiDARs, radares, GPS e potência do sinal mmWave

recebido podem identificar transmissores e bloqueios prováveis no ambiente sem fio e entender seus padrões de mobilidade. Essas informações podem ser usadas pela rede sem fio para prever proativamente bloqueios iminentes e iniciar a transição de conexão antecipadamente. Chamamos esse método de previsão e transição auxiliadas por sensores. A previsão de bloqueio baseada em assinaturas sem fio é um caso especial em que a estação base tenta antecipar bloqueios futuros na conexão LOS usando informações da potência do sinal mmWave recebido.

A assinatura pré-bloqueio de mmWave refere-se a um padrão de sinal observado antes que uma obstrução interrompa o caminho de linha de visão (LOS) entre um transmissor e um receptor fixos em um determinado ambiente. Quando um objeto se move nesse ambiente e eventualmente obstrui o caminho de LOS, ele atua como um dispersor para o sinal que viaja do transmissor para o receptor. Durante esse movimento, o sinal recebido sofre interferência construtiva e destrutiva do raio LOS e do raio disperso pelo objeto em movimento.

Conforme o objeto se aproxima do link de LOS, a contribuição desse bloqueio ou dispersor em movimento altera o sinal recebido. Essa mudança no padrão do sinal recebido, antes que o bloqueio ocorra, caracteriza o comportamento do objeto obstrutivo. Esse padrão é chamado de assinatura pré-bloqueio wireless.

Essa assinatura wireless pré-bloqueio serve como indicador ou precursor da iminente obstrução do link de LOS, oferecendo informações valiosas sobre o comportamento e movimento de objetos que possam causar interrupções potenciais na comunicação sem fio.

Wireless Signature-based blockage prediction: Specific Task Description

2.3 Introdução

A previsão de bloqueio baseada em assinaturas sem fio na infraestrutura é a tarefa de prever proativamente os futuros bloqueios de ligação LOS (Line-of-Sight) utilizando um modelo de aprendizado de máquina e as assinaturas sem fio pré-bloqueio.

2.4 Objetivo da Tarefa de ML

Em qualquer momento t, dado uma sequência de 'r' vetores de potência mmWave recebidos anteriores e atuais t – r + 1, . . . , t, o objetivo principal desta tarefa é projetar um modelo de aprendizado de máquina que preveja os futuros bloqueios de ligação. Em geral, espera-se que o modelo de aprendizado de máquina retorne o estado de bloqueio nos próximos 'k' intervalos de tempo, ou seja, t + 1, . . . , t + k. Se houver algum bloqueio durante esses intervalos, o estado de bloqueio para os 'k' intervalos de tempo (futuro-k) é considerado como bloqueado. Para o desenvolvimento do modelo de ML, fornecemos um conjunto de dados rotulado consistindo de uma sequência ordenada de vetores de potência mmWave recebidos (entrada do modelo de ML) e o estado de bloqueio futuro verdadeiro.

2.5 Conclusão

A capacidade de prever bloqueios baseados em assinaturas sem fio desempenha um papel crucial na manutenção e otimização de redes de comunicação sem fio. Este trabalho propõe o uso de técnicas de aprendizado de máquina para realizar previsões proativas de bloqueio de ligação, proporcionando uma oportunidade de melhorar a confiabilidade e eficiência das comunicações sem fio.

Nesse estudo em específico usamos as seguintes configurações de layers e blocos]

$$\sum_{n=1}^{5,10} Future - n$$

Layer	Output Dim
Input	8 x 24
Conv(ReLU)	8 x 24
MaxPool1d()	8 x 16
Conv(ReLU)	8 x 16
MaxPool1d()	8 x 10
Softmax()	3

Tabela 3 – Configuração de Layers e Camadas

Essa tabela descreve uma arquitetura de rede neural convolucional (CNN) com diferentes camadas e as dimensões de saída após cada camada. Aqui está um resumo do que ocorre:

1. **Input**: A entrada inicial é um tensor de dimensão 8x24.

2. **Conv(ReLU)**: A primeira camada convolucional com ativação ReLU mantém as dimensões de saída como 8x24, indicando que não houve mudança no tamanho após a convolução.

3. **MaxPool1d()**: Em seguida, é aplicada uma camada de Max Pooling 1D que reduz a dimensão no eixo de largura para 8x16. Isso é comum em CNNs para reduzir o tamanho dos dados e extrair características mais importantes.

4. **Conv(ReLU)**: Outra camada convolucional é aplicada, mantendo as dimensões de saída como 8x16.

5. **MaxPool1d()**: Outra camada de Max Pooling 1D é aplicada, reduzindo as dimensões no eixo de largura para 8x10.

6. **Softmax()**: Finalmente, a camada de Softmax é usada para gerar as saídas finais. Ela converte a saída da camada anterior em uma distribuição de probabilidade sobre três classes diferentes, resultando em um tensor de saída de dimensão 3.

Resumindo, a rede começa com uma entrada de 8x24, passa por camadas convolucionais e de pooling, e termina com uma camada Softmax para classificar os dados em três categorias distintas.

	precision	**recall**	**f1-score**	**support**
Blockage	0.86	0.83	0.86	14698
No Blockage	0.82	0.85	0.86	14657
accuracy			0.85	44023
macro avg	0.85	0.85	0.85	44023
weighted avg	0.85	0.85	0.85	44023

Tabela 4 – Future -1

2.6 RNN - Rede Neural Recorrente

Uma rede neural recorrente (RNN) é uma classe de redes neurais que inclui conexões ponderadas dentro de uma camada, em comparação com as redes de feed-forward tradicionais, onde a conexão é feita apenas para camadas subsequentes. Essas conexões internas na RNN formam loops, permitindo que a rede armazene informações enquanto processa novas entradas. Essa capacidade de memória torna as RNNs ideais para tarefas de processamento em que entradas anteriores devem ser consideradas, como dados de séries temporais.

Ao lidar com dados de séries temporais, um único ponto de dados isolado não é suficientemente informativo, pois não captura os atributos importantes, como a tendência da série (crescente ou decrescente). Isso é semelhante ao processamento de linguagem natural, onde palavras ou letras sozinhas não são úteis, mas sim o contexto em que aparecem. Portanto, é necessário um novo tipo de topologia de rede que leve em consideração o histórico da entrada.

É aqui que as RNNs se destacam. Elas possuem uma capacidade interna de manter memória através de conexões de feedback, permitindo que a rede suporte o comportamento temporal. Por exemplo, a saída da camada oculta em uma RNN é retroalimentada para a própria camada oculta. Embora a rede ainda seja feed-forward (as entradas são aplicadas à camada oculta e, em seguida, à camada de saída), o estado interno da RNN é preservado por meio dos nós de contexto, que influenciam a camada oculta nas entradas subsequentes.

Em resumo, as redes neurais recorrentes (RNNs) são capazes de lidar com dados de séries temporais devido à sua capacidade de memória interna e feedback, permitindo que considerem o histórico das entradas. Essa propriedade torna as RNNs extremamente úteis em aplicações que exigem processamento temporal, como previsão de séries temporais, tradução automática e reconhecimento de fala.

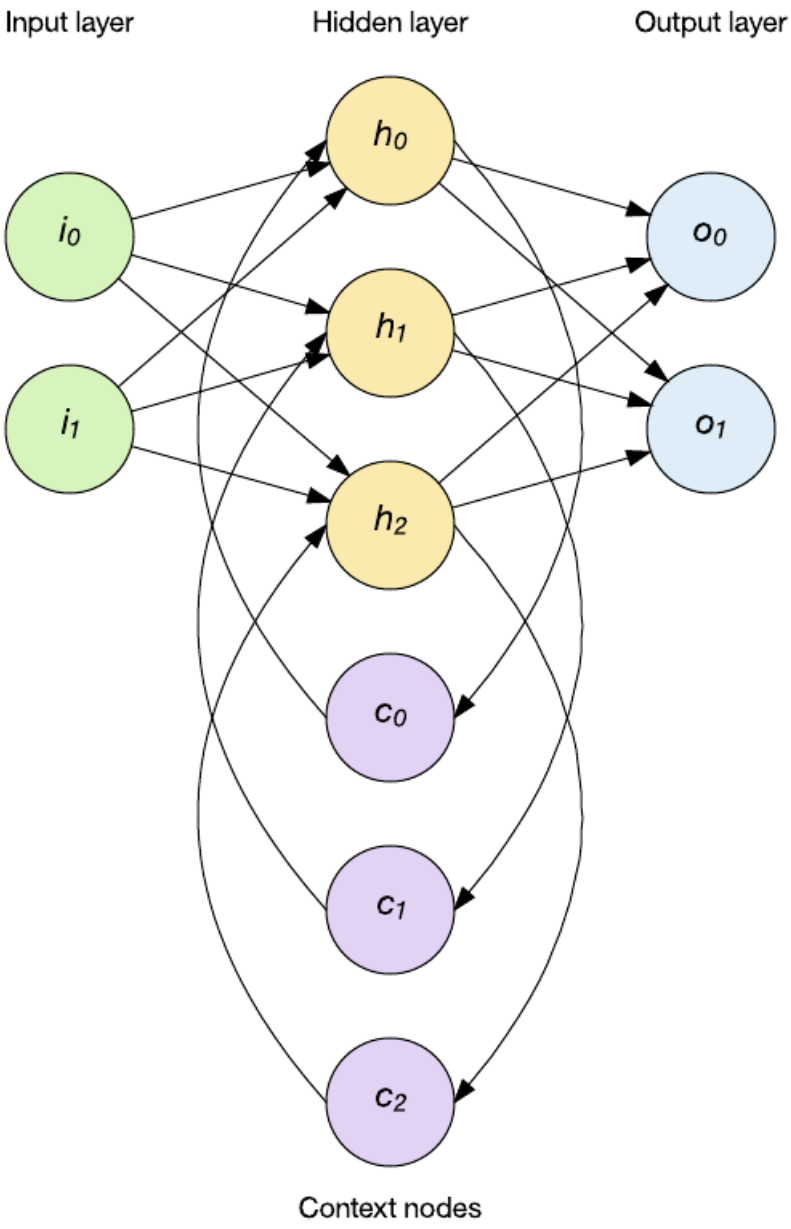


Figura 6 – RNN comum

2.6.1 Funcionamento da SimpleRNN

A camada RNN completa é apresentada como SimpleRnn. Ao contrário da arquitetura sugerida em muitos artigos, a implementação do Keras é bastante diferente, mas simples. Cada célula RNN recebe uma entrada de dados e um estado oculto que é passado de uma etapa única para a próxima. A arquitetura utilizada pelo keras é o jordan network.

Jordan Network

Rede Jordan, é um tipo de rede neural recorrente (RNN) que foi proposta pelo pesquisador Michael Jordan em 1986. Essa arquitetura recebe esse nome em referência ao seu criador.

As redes Jordan são uma variante das redes neurais recorrentes, onde a saída em um determinado instante de tempo é alimentada de volta para a entrada da rede, a fim de fornecer uma forma de memória interna. Essa realimentação permite que a rede leve em consideração informações passadas durante o processamento sequencial. Em uma rede Jordan, a saída em um determinado instante de tempo é usada como entrada na próxima etapa de tempo, juntamente com a entrada atual. Essa estrutura permite que a rede tenha memória e possa armazenar informações relevantes ao longo do tempo.

Uma característica importante das redes Jordan é a sua capacidade de lidar com sequências temporais de comprimentos variáveis. Ao contrário de muitas arquiteturas RNNs mais simples, como as redes de Elman, as redes Jordan podem lidar com sequências de tamanho variável sem a necessidade de truncá-las ou preenchê-las com valores especiais.

No entanto, vale ressaltar que as redes Jordan não são tão amplamente utilizadas quanto outras arquiteturas RNNs, como as redes LSTM (Long Short-Term Memory) e GRU (Gated Recurrent Unit). Essas arquiteturas mais recentes têm mostrado melhores resultados em muitas tarefas de processamento de linguagem natural e outras aplicações que envolvem sequências temporais.

A célula RNN tem a seguinte aparência,

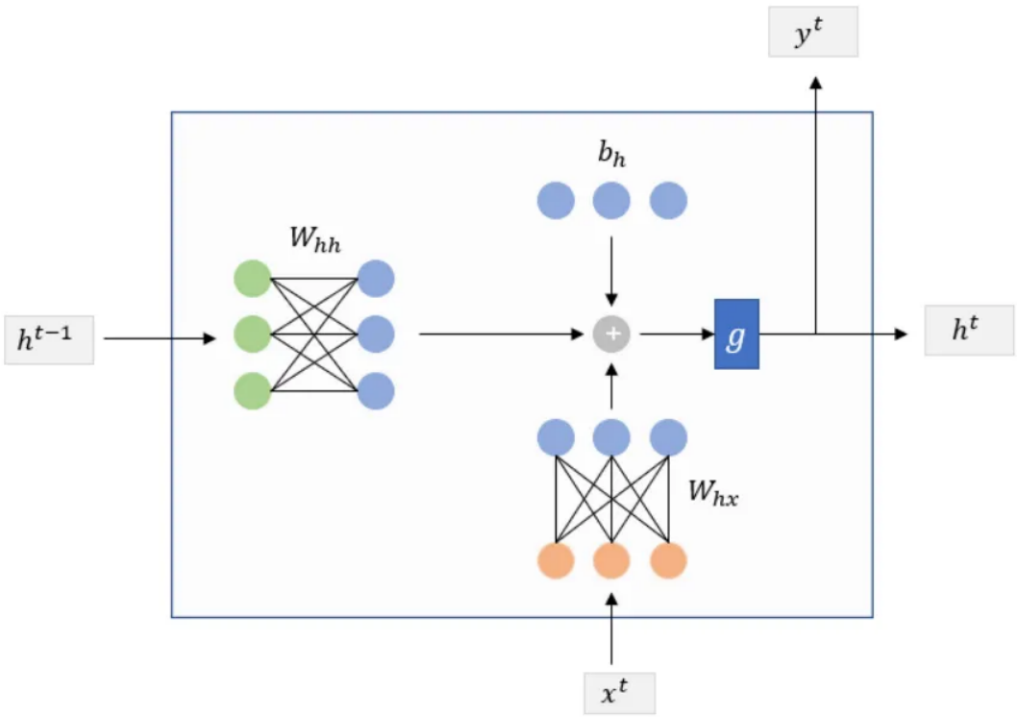


Figura 7 – Ilustração de uma camada simpleRNN no keras

A formulação completa de uma célula RNN é,

$$\begin{aligned} h^t &= g(W_{hh}h^{t-1} + W_{hx}x^{t-1} + b_h) \\ y^t &= h^t \end{aligned} \tag{2.1}$$

aqui, ht e ht-1 são os estados ocultos do tempo t e t-1. xt é a entrada no tempo t e yt é a saída no tempo t. O importante a notar é que existem duas matrizes de peso Whh e Whx e um termo de viés bh. Cada uma dessas matrizes pode ser considerada como uma rede neural interna de 1 camada com tamanho de saída conforme definido no parâmetro units, também o viés tem o mesmo tamanho. yt é puro ht e não aplicamos outra matriz de peso aqui, conforme sugerido por muitos artigos. Isso representa uma célula individual de RNN e a combinação sequencial de células (contagem igual a intervalos de tempo nos dados) cria a camada RNN completa. Lembre-se de que as mesmas matrizes de peso e viés são compartilhadas pelas células RNN

2.6.2 Aplicando simpleRNN

2.6.2.1 Pré-processamento

Aplicando Fourier: Para realizar uma análise mais aprofundada dos dados, utilizei a Transformada de Fourier como parte do processo de pré-processamento. A Transformada de Fourier é uma técnica matemática que nos permite decompor um sinal em suas frequências constituintes. Ao aplicar essa transformada aos dados, pude extrair informações valiosas sobre as amplitudes e frequências presentes em cada dado.

Essas informações extraídas foram extremamente úteis para compreender melhor os padrões presentes nos dados. Por exemplo, se houvesse uma alta amplitude em uma frequência específica, poderia indicar a presença de um evento periódico ou padrão recorrente nos dados. Por outro lado, uma baixa amplitude em uma determinada frequência poderia sugerir que essa frequência não desempenha um papel significativo nos dados analisados.

Adicionando rótulos aos dados: Ao lidar com um conjunto de dados, muitas vezes é necessário atribuir rótulos ou categorias a cada instância ou amostra. Esses rótulos ajudam a identificar e distinguir diferentes classes ou grupos de dados. No meu caso, o objetivo é rotular os dados em três classes distintas: 0, 1 e 2.

Adicionando pesos aos dados: A multiplicação dos conjuntos de dados por pesos é uma técnica utilizada para ampliar a diferenciação entre os dados durante o processo de aprendizado de máquina. Essa etapa é particularmente útil quando se deseja atribuir maior importância a certos conjuntos de dados em relação a outros, com o objetivo de destacar características específicas ou enfatizar determinadas informações relevantes. Ao introduzir maior diferenciação entre os conjuntos de dados por meio da multiplicação por pesos, os modelos de aprendizado de máquina são capazes de capturar melhor as nuances e as peculiaridades presentes nos dados, proporcionando uma melhor representação e um aprendizado mais adaptado às características específicas do problema em questão.

2.6.2.2 Construção

Batch Normalization: é uma técnica comumente usada em redes neurais profundas para acelerar o treinamento e melhorar o desempenho dos modelos. Ela normaliza as ativações de uma camada, ajustando-as para ter uma média zero e uma variância unitária, através do cálculo de média e desvio padrão com base em um mini-batch de exemplos durante o treinamento. Para mais informações

Glorot uniforme: também conhecida como inicialização uniforme de Xavier, é uma técnica popular de inicialização de pesos usada em aprendizado profundo. Ela recebe o nome de seu criador, Xavier Glorot. A inicialização Glorot uniforme tem como objetivo definir os pesos iniciais das camadas de uma rede neural de forma a facilitar o treinamento eficaz inicialização Glorot uniforme extrai os pesos iniciais de uma distribuição uniforme com limites dependentes do número de unidades de entrada e saída da camada. A distribuição é simétrica em torno de zero, e a faixa da distribuição é calculada para garantir que a variância dos pesos seja aproximadamente igual a 1,0. Isso ajuda a evitar que as ativações sejam muito grandes ou muito pequenas durante a propagação direta e inversa, o que pode prejudicar o treinamento. Para mais informações

Orthogonal: A inicialização de pesos é uma etapa crítica ao construir uma rede neural, pois pode afetar significativamente o desempenho e a convergência do modelo. Métodos de inicialização tradicionais, como inicialização uniforme ou inicialização normal, podem gerar matrizes de pesos que não são ortogonais, o que pode causar problemas como desvanecimento ou explosão do gradiente. O inicializador orthogonal aborda esse problema calculando uma matriz ortogonal que é usada para inicializar os pesos das camadas da rede neural. Para fazer isso, aplica-se um algoritmo chamado decomposição em valores singulares (SVD, do inglês Singular Value Decomposition) a uma matriz aleatória com dimensões apropriadas. A matriz resultante é usada para inicializar os pesos da camada. Para mais informações

Activation: refere-se à função matemática aplicada a um neurônio ou unidade de processamento em uma camada de uma rede neural. A ativação é aplicada à saída de cada neurônio para introduzir não linearidade na rede e permitir que a rede aprenda relações complexas entre os dados de entrada.

• Função de ativação tangente hiperbólica (tanh): Similar à função sigmoide, a função tangente hiperbólica também mapeia os valores de entrada para um intervalo entre -1 e 1. Ela é frequentemente usada em camadas ocultas de redes neurais.

• Função de ativação ReLU (Rectified Linear Unit): A função ReLU retorna zero para valores de entrada negativos e a própria entrada para valores positivos. É uma função não linear e tem sido amplamente utilizada nas camadas ocultas de redes neurais devido à sua eficiência computacional e à sua capacidade de evitar o problema do desaparecimento do gradiente.

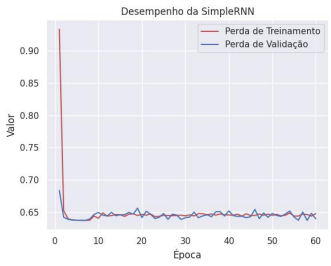
• Função de ativação softmax: A função softmax é usada principalmente na camada de saída de uma rede neural para problemas de classificação multiclasse. Ela normaliza as saídas em uma distribuição de probabilidades, permitindo a seleção da classe mais provável.

Optimizador: Adamax é uma variação do otimizador Adam, um algoritmo popular usado para otimização de gradientes em redes neurais. Enquanto o Adam combina os métodos de atualização do momento (momentum) e do RMSprop, o Adamax adiciona um elemento adicional ao cálculo das atualizações dos pesos. O Adamax utiliza a norma L (norma do infinito) para calcular o elemento adicional. A norma L de um vetor é simplesmente o valor absoluto máximo dos seus componentes. Ao utilizar essa norma, o Adamax estima o valor máximo absoluto dos gradientes em vez de calcular uma média móvel exponencial.

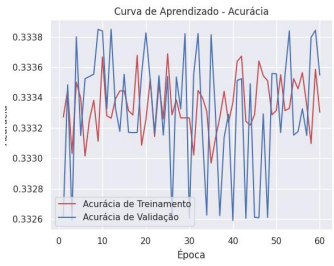
Dense: Em Keras, uma camada Dense (densa) é uma camada totalmente conectada em uma rede neural. Também é conhecida como camada de neurônios totalmente conectados ou camada de multiplicação matricial. A camada Dense é responsável por aplicar uma transformação linear seguida de uma função de ativação aos dados de entrada. Cada neurônio na camada está conectado a todos os neurônios da camada anterior. Essa conectividade total permite que a camada Dense aprenda representações complexas dos dados. Para mais informações acesse

2.6.2.3 Processamento

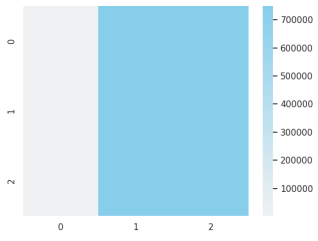
Vamos aplicar agora o código da etapa nomeado “simpleRNN” disponibilizado no inicio desse tópico. Vamos comparar como o código funciona com e sem aplicação de pesos as dados de entrada do modelo. Vamos analisar três métricas: perda, acurácia e matrix de confusão.



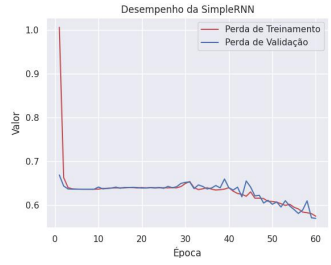
(a) perda



(b) acuracia

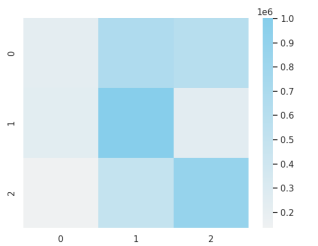


(c) matriz



(d) perda

(e) acuracia



(f) matriz

Figura 8 – imagens a,b e c referentes a aplicação sem pesos. imagens d,e e f referentes a aplicação com pesos.

2.6.3 RNN vertical profunda(Seq2Seq)

Em uma RNN vertical, a saída de uma camada RNN é usada como entrada para a próxima camada RNN acima dela. Dessa forma, as informações fluem verticalmente através das camadas RNN, permitindo que cada camada aprenda representações hierárquicas e abstratas dos dados sequenciais.

A arquitetura de uma RNN vertical pode ser visualizada como uma pilha de camadas RNN, onde as saídas de uma camada são propagadas como entradas para a camada seguinte. A última camada RNN geralmente é seguida por uma camada de saída, como uma camada densa, que pode gerar a saída final.

A vantagem de utilizar uma RNN vertical é que ela pode capturar representações mais complexas e abstratas dos dados sequenciais. À medida que as informações fluem através das camadas, cada camada pode aprender a detectar padrões e dependências de diferentes níveis de abstração.

No entanto, é importante considerar o aumento da complexidade computacional ao empilhar várias camadas RNN. Além disso, o treinamento de RNNs verticais profundas pode ser mais desafiador devido ao problema do gradiente que tende a diminuir ou explodir ao retro propagá-lo através de várias camadas

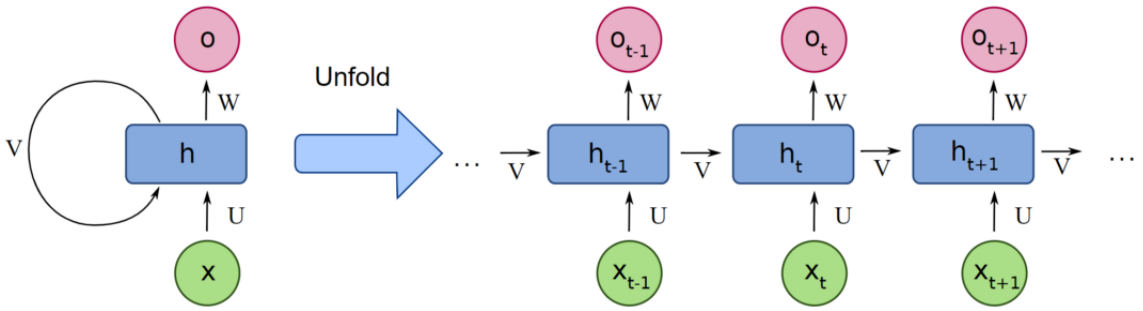
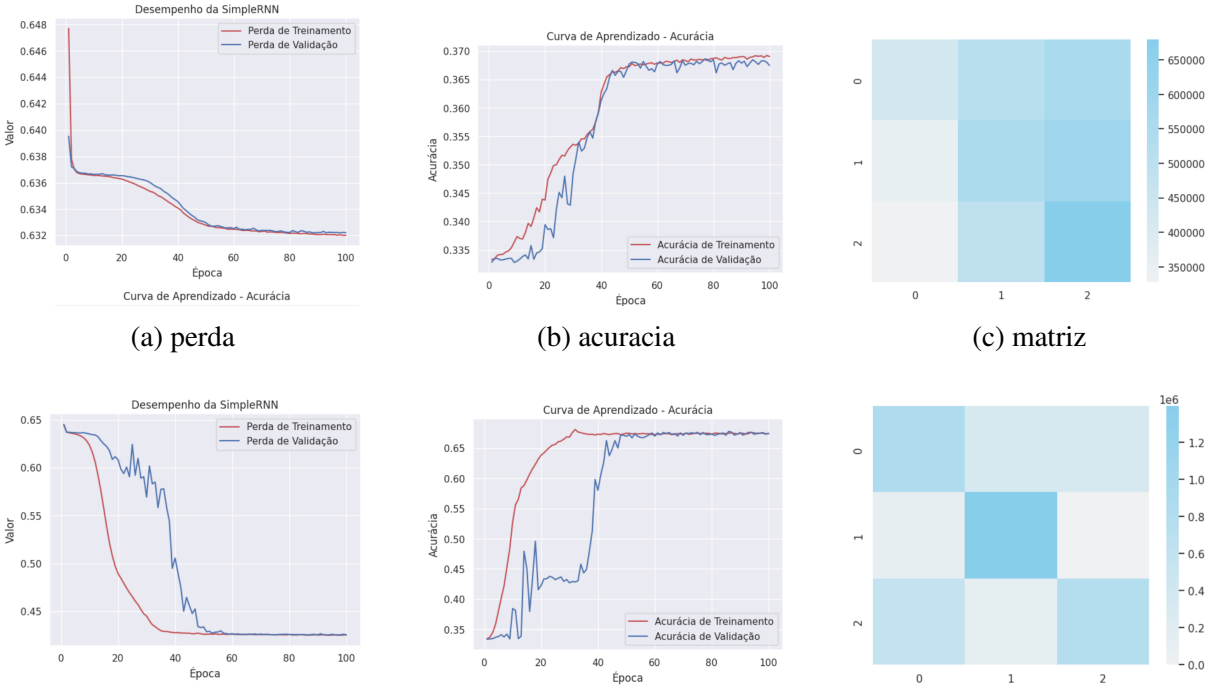


Figura 9 – Ilustração de uma dupla camada

Para esse modelo, assim como no anterior, vamos utilizar eventos com pesos e sem aplicação de pesos, vamos comparar como o algoritmo se comporta em cada situação

2.6.4 Comentários finais

No trabalho realizado, foi explorado o uso de redes neurais recorrentes (RNNs) com as camadas SimpleRNN do Keras para a tarefa de sequência para sequência (Seq2Seq). A abordagem adotada permitiu a geração de insights valiosos sobre os pontos positivos e negativos dessa configuração específica. Uma das principais vantagens do uso de RNNs com as camadas SimpleRNN é a sua simplicidade e facilidade de implementação. O Keras, como uma biblioteca de alto nível, oferece uma interface intuitiva para a construção e treinamento desses modelos, permitindo que os pesquisadores e desenvolvedores se concentrem mais nos



(a) perda (b) acuracia (c) matriz

(d) perda (e) acuracia (f) matriz

Figura 10 – imagens a,b e c referentes a aplicação sem pesos. imagens d,e e f referentes a aplicação com pesos.

aspectos específicos do problema em vez de se preocuparem com detalhes de baixo nível. No entanto, ao utilizar as camadas SimpleRNN, é importante levar em consideração algumas limitações. Uma delas é o desafio do desvanecimento ou explosão do gradiente. À medida que as informações são propagadas ao longo da rede, os gradientes podem se tornar muito pequenos ou muito grandes, dificultando o treinamento eficiente do modelo. Outro ponto a ser considerado é a capacidade limitada das camadas SimpleRNN em capturar dependências de longo prazo. Essa limitação pode resultar em dificuldades na compreensão de contextos complexos ou sutilezas em sequências de dados mais extensas. Em casos em que a tarefa exige um entendimento mais profundo do contexto, outras variantes de RNNs, como as redes neurais recorrentes de memória de longo prazo (LSTM) ou as redes neurais recorrentes com portas de unidades de memória (GRU), podem ser mais adequadas. Em conclusão, o modelo Seq2Seq mostrou resultados promissores, com um aumento significativo na acurácia em comparação com a aplicação de uma única camada. Obtivemos quase o dobro da acurácia, o que indica o potencial dessa abordagem para a tarefa em questão. No entanto, é importante considerar cuidadosamente fatores como o número de neurônios, taxa de aprendizado e tamanho do lote (batch size) ao aplicar essa abordagem. O algoritmo Seq2Seq demonstrou ser sensível a esses fatores, conforme observado durante os experimentos. À medida que aumentamos o número de camadas, a influência desses fatores se tornou mais evidente, afetando o desempenho do modelo. Portanto, é necessário ajustar e otimizar esses parâmetros para obter os melhores resultados com a abordagem Seq2Seq utilizando as camadas SimpleRNN.

2.7 ResNet - Residual Network

2.7.1 A Rede Neural Residual

As Redes Neurais Convolucionais já se provaram bastante capazes de encontrarem padrões e obter informações para dados de sinais sem fios, sendo bastante utilizadas para classificação de modulação e sensoriamento espectral. Todavia, na visão computacional as chamadas Redes Neurais Residuais (ResNet) estão se tornando cada vez mais eficientes e estão sendo efetivamente usadas para séries temporais para dados de áudio, por exemplo.

O principal motivação de utilizar uma Rede Neural Residual é pelo fato de que ela consegue ser amplamente dimensionada, criando várias camadas obtendo assim uma analise efetiva dos dados a diferentes escalas. O motivo por trás disso é que a ResNet é capaz de efetivamente evitar o problema do gradiente de fuga, que acontece com redes muito profundas, onde o gradiente se torna muito pequeno ou até mesmo zero de modo que a rede se torna incapaz de aprender caracteristicas sobre os dados por meio de *BackPropagation*. A ResNet evita este problema e similares, pois utilizam camadas convolucionas e “saltos de ligações” entre essas camadas, de modo que o valor dos pesos antes de passar por uma determinada camada convolucional “salta” e soma-se com os novos pesos na saída desta camada de convolução. Deste modo a ResNet pode ser capaz de manter e passar informações do gradiente para interações posteriores ao "saltar-las"a frente, assim evitando a dimiuição expoêncial do gradiente e sendo capaz de criar redes neurais com mais camadas ou com maior de velocidade convergência ao utilizar menos épocas de treino.

2.7.2 Modelo ResNet e conjunto de dados

Neste trabalho utilizamos a ResNet para sensoriamento espectral de dados de sinais sem fio de WiFi e LTE, além de um conjunto de dados sem sinal (ruído). Foram utilizado um conjunto de dados de 300 milhões de amostras de sinais em fase e quadratura, equivalente a 15 segundos de captação desses sinais. No total foram 100 milhões amostras para cada um dos tipos de sinais que utilizamos, ou seja, equivalente a 5 segundos de captação para WiFi, LTE e para o ruído.

Foi utilizado um modelo de ResNet utilizando Residual Stack ou Pilha Residual onde agrupamos duas unidades residuais e realizamos um *MaxPooling* na saída da mesma. Podemos ver na figura abaixo como é constituído o Residual Stack.

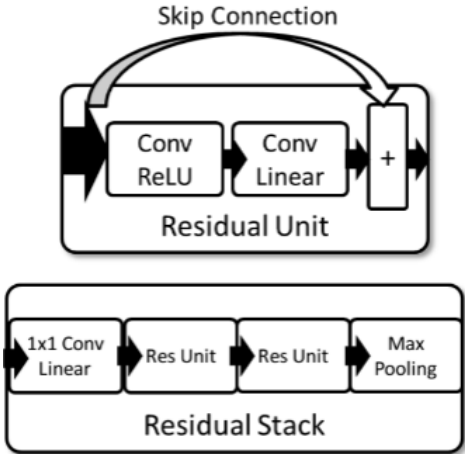


Figura 11 – Residual Unit e Residual Stack (O'SHEA et al., 2018)

A arquitetura final do modelo pode ser vista na figura abaixo:

Layer	Output dim.
Input	2 x 20000
ResidualStack	4 x 10000
ResidualStack	4 x 2000
ResidualStack	4 x 1000
ResidualStack	4 x 200
ResidualStack	4 x 100
ResidualStack	4 x 20
ResidualStack	4 x 10
FC/Tanh	16
FC/Tanh	16
FC/Softmax	3

Figura 12 – Arquitetura da ResNet (BALDESI et al., 2022)

O conjunto de dados foi divido em três partes, sendo elas um conjunto de dados para treinamento da ResNet contendo 60% dos dados, um conjunto para validação com 20% do total e outros 20% foram utilizados para testes no modelo.
Como parametro de entrada da ResNet foi utilizado uma matriz de tamanho de 20000×2, o equivalente às partes em fase e quadratura de uma série temporal com 20 mil amostras de determinado sinal, totalizando 1 milisegundo amostral.
O modelo foi treinado utilizando uma taxa de aprendizado de 0.001 e um total de 200 épocas de treino com *batch size* de 45 e 4 filtros por camada.

2.7.3 Treinamento do modelo

Podemos ver na figura abaixo o resultado da curva obtida para função objetivo (Loss Function) no treinamento e para os dados de validação, podemos notar que para os dados de treinamento obtivemos uma curva mais suave porém na validação houve algumas oscilações maiores, todavia ainda seguindo a padrão de precisão do treinamento. Vemos tambéom que a curva se

mantém constante próxima a zero porém sempre diminuindo informando que o modelo conseguiu realizar um treinamento eficiente sem o problema da fuga do gradiente.

Figura 13 – Loss de treino e validação.

Agora na figura 40 notamos a acurácia do modelo para os dados de treinamento e validação, vemos que o modelo conseguiu mesmo com um valor menor de acurácia para treinamento, um valor ínicio para os dados de validação já acima de 90% antes mesmo de 25 epócas, comprovando a eficência no modelo ao "saltar" informações sobre o gradiente para futuras interações logo no ínicio do treinamento. Verificamos também que mesmo obtendo uma acurácia elevado o modelo não está apresentando *overfitting* pois conseguimos altos valores de acurácia também para os dados de validação.(JUNIOR, 2023)

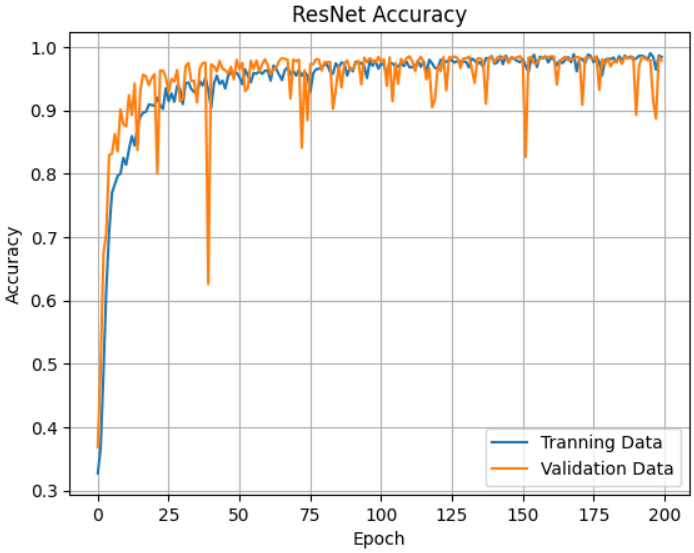


Figura 14 – Acurácia de treino e validação.

2.7.4 Testando o modelo

Para verificar se o modelo realmente é eficaz, inserirmos novos dados não vistos antes no pelo modelo no treinamento e validação para realização de testes. Ao inserir novos dados o modelo prever a probabilidade de cada classe para cada *input* de dados. Classificamos a classe com maior probabilidade retornada pelo modelo como a classe do *input*. O conjunto de dado de testes continha 3000 séries temporais de 20 mil amostras cada. O modelo conseguiu uma acúrácia de 97.87% para os dados de teste evidênciando a capacidade do modelo de sensoriamento dos dados e de generalizar os resultados para dados ainda não vistos. Abaixo verificamos o resultado da classificação dos dados de testes pela matriz de confusão dos dados. Podemos notar que o resultado final entre sinais LTE e WiFi não foram classificados em nenhum caso errado entre eles. Vemos que o modelo teve dificuldade apenas em classificar WiFi e o Ruído, onde alguns dados foram classificados erroneamente como ruído sendo amostras de WiFi e vice-versa. Porém a quantidade de dados classificados errados é muito pequena comparado a quantidade de dados classificados corretamente, isto prova a eficiência do modelo em classificar os dados.

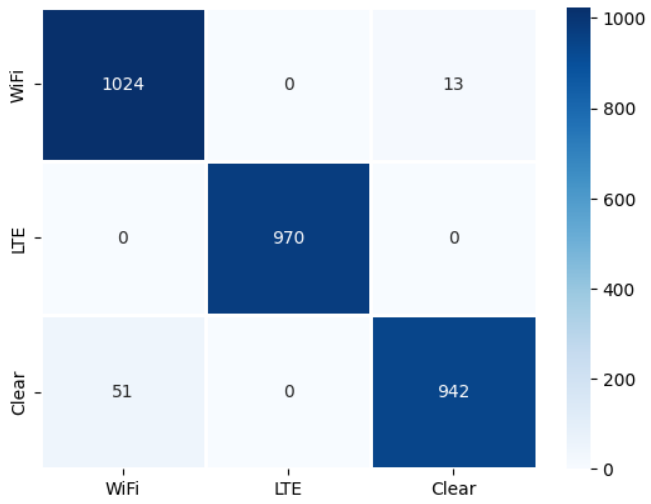


Figura 15 – Matriz de confusão dos dados de testes.

2.7.5 Comentários finais

Com base nos resultados podemos verificar que a ResNet consegue realizar o sensoriamento espectral a partir do treinamento do modelo com dados brutos de amostras de sinais sem fio, apenas com as componentes em fase e quadratura. Podemos verificar também que o modelo é capaz de rapidamente obter uma acurácia alta com poucos treinamentos, como foi dito anteriormente, onde com menos de 25 epócas o modelo já era capaz de obter mais de 90% de acurácia nos dados de validação, mas também é capaz de obter resultados melhores com o aumento de epócas. No final de 200 epócas o modelo foi capaz de conseguir até 98% de acurácia com os dados de validação e embora a curva de aprendizado estivesse menor, o

modelo não ainda estava aprendendo evidênciando a sua capacidade de aprender sem fuga do gradiente.

Podemos verificar também que o modelo foi capaz de corretamente separar 100% das amostras de WiFi e LTE entre si, apenas falhando em separar WiFi e ruído em alguns casos. Essa falha no modelo poderá ser mitidaga possívelmente com aumento do conjunto de dados, pois neste modelo foram utilizados poucos dados e ainda sim, conseguiu obter resultados expressivos.

O modelo também poderá ser modificado alterando a quantidade de filtros e Residual Stack o que causaria uma maior complexidade computacional porém aumentaria a eficiencia do modelo.

3 Conclusão e perspectivas

Ao final deste trabalho, temos em mãos um material de pesquisa que apesar de não muito recente, está sempre buscando acompanhar as inovações tecnológicas no mercado, desta forma o trabalho não adquire características obsoletas, e está sempre aberto a adições e melhoras.
Tendo em vista o que foi abordado em todo o trabalho, as técnicas de sensoriamento espectral são muito valiosas se inserirmos no contexto tecnológico atual. Vivemos em um mundo onde a tendência é que cada vez mais hajam aparelhos conectados à internet, e necessitando dos meios de comunicação, e a medida que os avanços tecnológicos ocorrem, deve-se ter ferramentas e soluções para otimizar a utilização dos mesmos.
A exemplo disso, no Brasil, recentemente tivemos a implementação da tecnologia 5G, com seu advento, tendências como IoT (Internet of Things - Internet das Coisas), farão com que não mais, apenas nossos smartphones estejam conectados, mas também a maior parte de nossos dispositivos, sejam moveis ou estacionários, o que corrobora com a ideia de "superutilização "do espectro. Enquanto no Brasil há a recente implementação do 5G, ao redor do mundo já existem pesquisas sobre o 6G.
Em linhas gerais, verificamos que para o conjunto de dados que possuímos de 6 min de captura de informação para o sinal LTE, CLEAR, WIFI. Verificamos que a melhor acurácia foi destacada pela rede neural CNN , Resnet, MLP e RNN , nessa perspectiva ordem.

Referências

BALDESI, L.; RESTUCCIA, F.; MELODIA, T. ChARM: NextG Spectrum Sharing Through Data-Driven Real-Time O-RAN Dynamic Control. In: **IEEE INFOCOM 2022 - IEEE Conference on Computer Communications**. [S.l.: s.n.], 2022.

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

JUNIOR, J. **Codigo fonte CNN-pytorch**. 2023. Disponível em: <https://github.com/JanathanPlanas/Deep-learning-AICT/tree/Models-DeepLearning/Cnn-Pytorch>.

O'SHEA, T. J.; ROY, T.; CLANCY, T. C. Over-the-air deep learning based radio signal classification. **IEEE Journal of Selected Topics in Signal Processing**, IEEE, v. 12, n. 1, p. 168–179, 2018.

RASCHKA, S. **Python machine learning**. [S.l.]: Packt publishing ltd, 2015.