Inteligência Computacional
Rede neural Convolucional: Classificação de imagens CIFAR-10

Gabriel Tambara Rabelo
Departamento de engenharia elétrica
Faculdade de Tecnologia - UnB
Brasília, Brasil
gtambararab@gmail.com

***Abstract*—The use of convolutional neural networks is widely used for image classification for its ability to toplogically identify features in images. One of the most widely used datasets for experimenting with image classification networks is CIFAR-10 and three networks have been modeled and designed to validate its functionality in classifying 10 different categories, as well as the effects of creating an 11th category, for low-confidence classifications.**

Keywords: convoluted; intelligence; artificial; neural; network.

***Resumo*—O uso de redes neurais convolucionais é amplamente utilizado para classificação de imagens pela sua capacidade de identificação topológica de características em imagens. Um dos conjuntos de dados mais utilizados para experimentar redes de classificação de imagens é o CIFAR-10 e três redes foram modeladas e projetadas para validar sua funcionalidade na classificação de 10 categorias diferentes, bem como os efeitos da criação de uma 11ª categoria, para classificações de baixa confiança.**

Palavras-chave: convolucional; inteligência; artificial; rede; neural.

I. Introdução

II. Fundamentação teórica

A. Convolutional Neural Network

1) Estrutura geral: Uma *Convolutional Neural Network*, ou CNN, é uma arquitetura de rede neural artificial profunda, portanto, parte das arquiteturas de *Deep learning*, ou redes profundas. As redes profundas são redes que se diferenciam de outras topologias por possuírem um maior número de camadas em sua estrutura, permitindo e objetivando alcançar um maior número de detalhes em sua identificação. Nesse modelo, cada camada é responsável por identificar pequenos padrões, e conforme o número de camadas aumenta, a percepção da rede se torna cada vez mais generalista a ponto de conseguir observar padrões maiores e mais complexos. A maior parte de suas aplicações se encontra em identificação de padrões visuais, o que, aplicando diretamente seus conceitos, consegue identificar pequenos detalhes em uma imagem a ponto de eventualmente conseguir identificar características de interesse para um operador da rede. Dentre as redes profundas, a que se especializa em identificação espacial é a CNN, ou Convoluted Neural Network. Enquanto outras redes neurais totalmente conectadas exigem que os dados de entrada sejam representados como um vetor unidimensional, perdendo a informações posicionais, as CNNs preservam essa estrutura com o uso de camadas convolucionais.

O maior mecanismo de sua funcionalidade é a convolução, em que um filtro, também chamado de kernel, percorre a matriz de entrada e realiza operações multiplicativas para extrair características. Como detalhe, esses filtros são aprendidos durante o treinamento da rede e muitas vezes um padrão inicial não é necessário. Através deste mecanismo, há o compartilhamento de pesos, onde à medida que o kernel desliza pela entrada, atuando em várias localizações, permite-se que as mesmas características sejam detectadas em diferentes partes da imagem. Isso leva a um compartilhamento de informações e uma redução significativa no número de parâmetros a serem aprendidos, tornando as CNNs adequadas quanto ao consumo de memória e processamento. Através do uso de camadas convolucionais, camadas de pooling, e camadas de compartilhamento de pesos, além outros componentes específicos das CNNs, a arquitetura é capaz de capturar informações espaciais e hierárquicas nas imagens, resultando em um desempenho geralmente superior em problemas de classificação de imagens em comparação com outras arquiteturas de redes neurais.

2) Camada de convolução: A camada de convolução receberá a matriz de entrada, geralmente um vetor 3D, com duas camadas para formar uma imagem plana e mais uma para dividir os três canais de cores em RGB (*red*, *green*, e *blue*). Com isso, ela gerará uma nova matriz tridimensional, compondo os filtros usados para identificar elementos, através dos pesos de cada pixel e seus adjacentes. Comumente são implementadas as funções de ativação como a RELU já nesta camada.

3) Camada de Pooling: As camadas de pooling são utilizadas para reduzir o tamanho das camadas de imagem da matriz na rede, e portanto, dos filtros obtidos, o equivalente à reduzir a resolução das imagens, o que reduziria o espaço para armazenar informações e as resumiria, de forma a facilitar para as próximas camadas abstrairem suas informações.

B. Camada de Dropout

As camadas de Dropout funcionam desativando aleatoriamente neurônios da rede durante o treinamento, impedindo sua contribuição para a propagação do gradiente e pesos. Essa desativação aleatória força a rede a aprender recursos redundantes e a torna mais robusta, reduzindo efeitos como o de overfitting, onde classifica-se melhor os dados de treinamento do que dados genéricos e diferentes apresentados à rede.

1) Camada densa: A última camada geralmente é uma camada densa (totalmente conectada) com um número de neurônios igual ao número de classes. Essa camada é seguida por uma função de ativação, como a função softmax ou sigmoid, para produzir as probabilidades de cada classe no processo de identificação.

2) Data augmentation: Data augmentation, ou aumento de dados, é uma técnica utilizada no treinamento de redes neurais que consiste em criar novas amostras de dados artificiais a partir das amostras originais. O objetivo é aumentar a quantidade e a diversidade dos dados disponíveis para o treinamento, melhorando assim a capacidade do modelo de classificar informações diferentes das apresentadas no seu treinamento. Para que isto ocorra, são realizadas transformações controladas que incluem operações como rotação, espelhamento, zoom etc. Essas variações nos dados de treinamento ajudam a evitar o *overfitting* e a aumentar a robustez do modelo.

C. Situação problema

D. CIFAR-10

O CIFAR-10 é um conjunto de dados extensivamente utilizado para classificação de imagens que consiste em um conjunto de 60.000 imagens coloridas de baixa resolução, de 32x32 pixels, igualmente divididas em 10 classes diferentes. Cada classe representa uma categoria específica de objetos ou animais exclusivamente separados, impedindo interseção dos conjuntos.

No contexto da classificação do conjunto CIFAR-10, a aplicação de uma CNN é apropriada devido à natureza de grade das imagens. Uma CNN pode capturar efetivamente as características visuais das imagens em várias camadas convolucionais, aprendendo a distinguir entre as diferentes classes, como ”cachorro”, ”carro”e ”avião”.

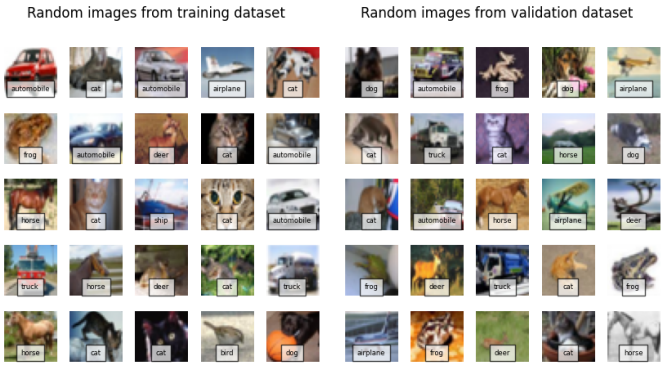
Portanto, objetiva-se testar a utilização de diferentes topologias de CNNs na resolução do problema de classificação das imagens da rede CIFAR-10, utilizando diferentes técnicas e analisando seus efeitos.

III. Laboratório

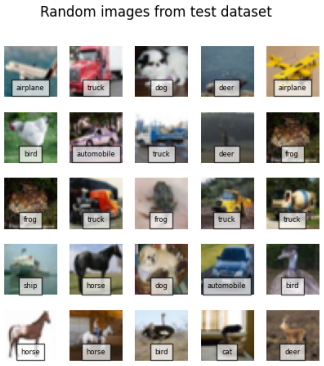
De início, foi utilizado o ambiente de simulação do Google Colab para todas as operações computacionais realizadas, bem como os pacotes da linguagem de programação Python, tensorflow e keras, para os processos de aprendizado de máquina.

A. Dados

Quanto aos dados do cifar-10, pode-se visualizar alguns dos exemplos aleatórios de cada um dos 3 grupos de dados utilizados, como apresentado pela Fig.[1]. Os conjuntos são separados em treinamento, para o processo de treinamento inicial da rede; validação, para ser utilizado em avaliar a performance da rede durante o treinamento; e testes, para validação da performance posterior ao treinamento. Ademais, não pode-se visualizar os efeitos de distorção utilizados para aumento de dados (flip horizontal, rotação aleatória e contraste), pois os mesmos são aplicados durante o treinamento para melhor aproveitamento de uso da GPU. Mais efeitos nao puderam ser aplicados dados os limites computacionais do ambiente gratuito do Google Colab.



(a) Dados de treinamento (b) Dados de validação

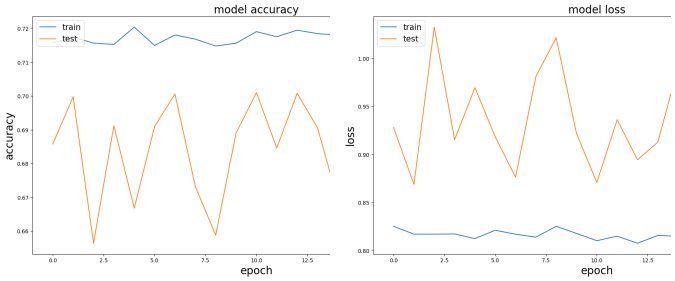


(c) Dados de teste

Figura 1: Conjuntos aleatórios de dados

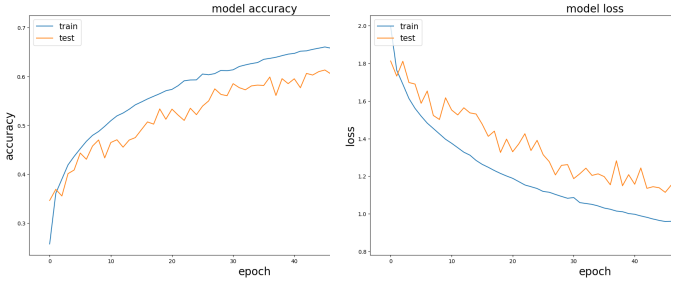
B. Primeira rede

A primeira rede desenvolvida, foi modelada através de um ciclo de tentativas iniciais de ordenação arbitrária de camadas de convolução, pooling e dropout. A estrutura dessa rede é dada pela Fig.[4]. Houveram duas tentativas de se implementar esta rede, mas foram necessários alguns ajustes como a mudança da função de ativação de sigmoid para softmax, objetivando uma melhor qualidade de classificação, além da redução da taxa de aprendizado em 10 vezes, para tentar reduzir os efeitos de overfitting. Os resultados podem ser comparados pelas Figs. [2, 3]. Na primeira tentativa da rede, pôde-se perceber a grande dissonância entre a rede de treinamento e validação, o que denomina um alto grau de overfitting, além de uma excessivamente rápida convergência de resultado.



(a) Acurácia por épocas de treinamento (b) Perda por épocas de treinamento

Figura 2: Performance na primeira tentativa da rede 1



(a) Acurácia por épocas de treinamento (b) Perda por épocas de treinamento

Figura 3: Performance na segunda tentativa da rede 1

A topologia final do primeiro modelo é apresentada pela Fig[4].

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 32, 32, 3)	0
conv2d_4 (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d_2 (MaxPooling 2D)	(None, 16, 16, 32)	0
conv2d_5 (Conv2D)	(None, 16, 16, 64)	18496
dropout_5 (Dropout)	(None, 16, 16, 64)	0
conv2d_6 (Conv2D)	(None, 16, 16, 128)	73856
max_pooling2d_3 (MaxPooling 2D)	(None, 8, 8, 128)	0
conv2d_7 (Conv2D)	(None, 8, 8, 256)	295168
dropout_6 (Dropout)	(None, 8, 8, 256)	0
flatten_1 (Flatten)	(None, 16384)	0
dropout_7 (Dropout)	(None, 16384)	0
dense_4 (Dense)	(None, 512)	8389120
dropout_8 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 512)	262656
dropout_9 (Dropout)	(None, 512)	0
dense_6 (Dense)	(None, 256)	131328
dense_7 (Dense)	(None, 10)	2570

Total params: 9,174,090
Trainable params: 9,174,090
Non-trainable params: 0

Figura 4: Topologia da primeira rede

Para avaliar os resultados, têm-se a matriz de confusão da primeira matriz, apresentada pela Fig.[5]. Percebe-se que as categorias foram bem classificadas e que houveram poucos erros que desviavam a matriz de sua diagonalidade.

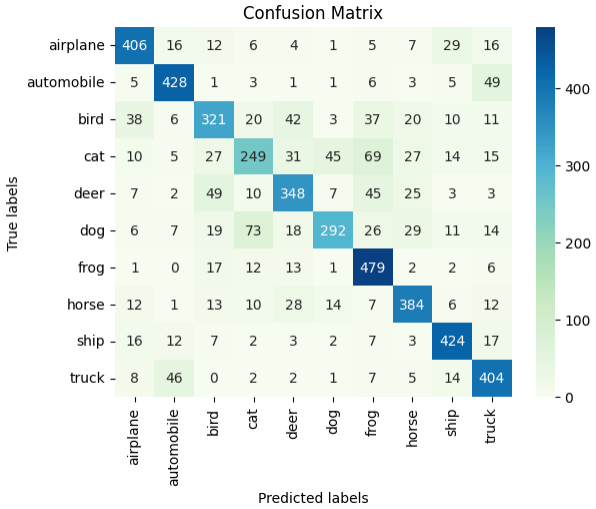


Figura 5: Matriz de confusão da primeira rede

A fins de análise dos filtros gerados, pode-se analisar um dos exemplos de saída de uma das camadas da primeira rede, apresentado pela Fig.[6].

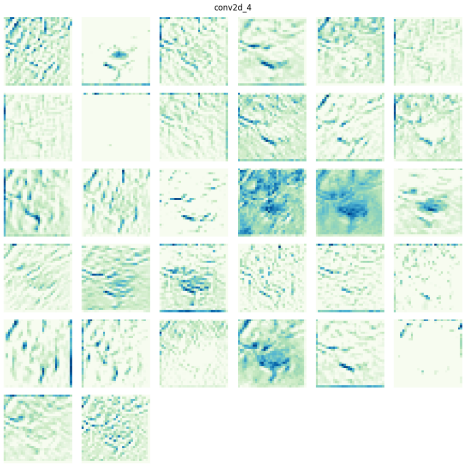


Figura 6: Feature map da topologia 1

C. *Segunda rede*

Quanto à segunda das três rede a serem criadas, foi reduzido o batch size objetivando melhorar os efeitos de overfitting de outra forma, contornando mínimos locais, além de que foram adicionadas diferentes camadas de pooling. Sua topologia pode ser vista pela Fig.[7]. Para esta rede, percebeu-se um grande aumento de tempo de treinamento, podendo estar relacionado à quantidade de filtros em cada camada convolucional, bem como pela quantidade de camadas no geral. O tempo de

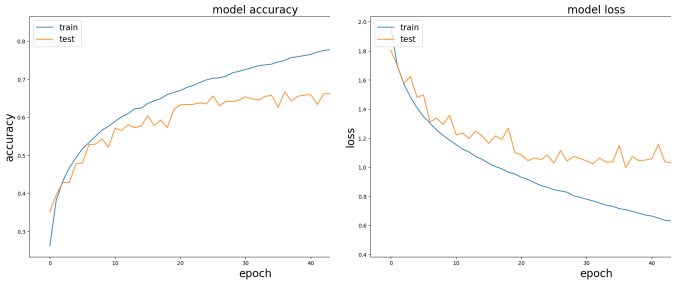
processamento geral, em comparação entre as redes, pode ser visto pela Tabela[I].

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 32, 32, 3)	0
conv2d_8 (Conv2D)	(None, 32, 32, 150)	28950
max_pooling2d_4 (MaxPooling 2D)	(None, 16, 16, 150)	0
dropout_10 (Dropout)	(None, 16, 16, 150)	0
conv2d_9 (Conv2D)	(None, 16, 16, 300)	2880300
conv2d_10 (Conv2D)	(None, 16, 16, 300)	5760300
max_pooling2d_5 (MaxPooling 2D)	(None, 8, 8, 300)	0
dropout_11 (Dropout)	(None, 8, 8, 300)	0
conv2d_11 (Conv2D)	(None, 8, 8, 300)	5760300
flatten_2 (Flatten)	(None, 19200)	0
dense_8 (Dense)	(None, 600)	11520600
dense_9 (Dense)	(None, 300)	180300
dense_10 (Dense)	(None, 150)	45150
dense_11 (Dense)	(None, 10)	1510

Total params: 26,177,410
Trainable params: 26,177,410
Non-trainable params: 0

Figura 7: Topologia da segunda rede

Para avaliar os resultados, segue os gráficos de performance durante o treinamento da rede, apresentados pela Fig.[8]. Além disso, têm-se a matriz de confusão da primeira matriz, apresentada pela Fig.[9]. Para esta rede, percebe-se que ainda assim, apesar dos esforços, houvue um severo grau de overfitting conforma as épocas do treinamento se passavam, de modo que o treinamento, mesmo com uma redução da taxa de aprendizado, ainda assim não possibilitou um bom resultado neste aspecto. Contudo, a acurácia final da rede demonstrou-se bem melhor, alcançando um valor acima de 80% para a rede de validação, conforme pode ser visto pela Tabela[I].



(a) Acurácia por épocas de treinamento (b) Perda por épocas de treinamento

Figura 8: Performance da segunda rede

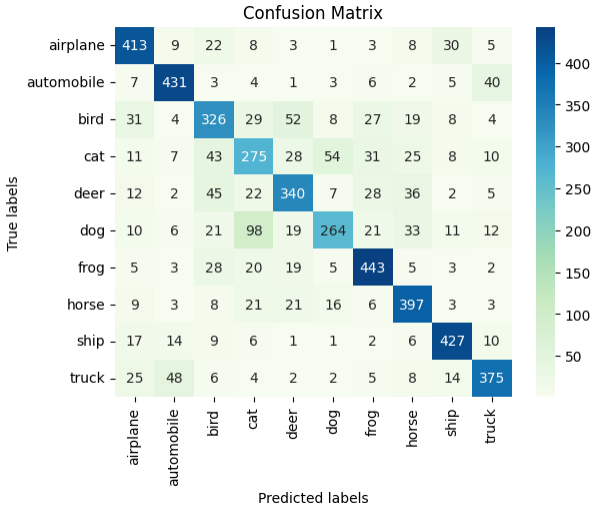


Figura 9: Matriz de confusão da segunda rede

A fins de análise dos filtros gerados, pode-se analisar um dos exemplos de saída das camadas da rede, apresentado pela Fig.[10].

Figura 10: Feature map da topologia 2

D. Terceira rede

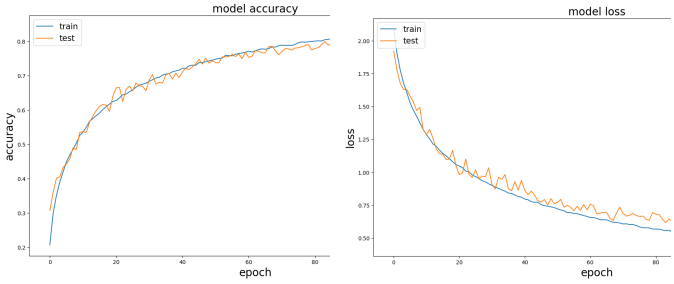
Quanto à terceira e última rede, foi gerado um modelo genérico através da ferramenta chat-GPT para basear seu modelo inicial e após algumas iterações pôde-se alcançar um modelo de melhores resultados de acurácia. Sua topologia pode ser vista pela Fig.[11].

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 32, 32, 3)	0
conv2d_4 (Conv2D)	(None, 32, 32, 32)	1568
conv2d_5 (Conv2D)	(None, 32, 32, 64)	32832
max_pooling2d_2 (MaxPooling 2D)	(None, 16, 16, 64)	0
dropout_2 (Dropout)	(None, 16, 16, 64)	0
conv2d_6 (Conv2D)	(None, 16, 16, 128)	131200
conv2d_7 (Conv2D)	(None, 16, 16, 256)	524544
max_pooling2d_3 (MaxPooling 2D)	(None, 8, 8, 256)	0
dropout_3 (Dropout)	(None, 8, 8, 256)	0
conv2d_8 (Conv2D)	(None, 8, 8, 256)	1048832
conv2d_9 (Conv2D)	(None, 8, 8, 512)	2097664
max_pooling2d_4 (MaxPooling 2D)	(None, 4, 4, 512)	0
dropout_4 (Dropout)	(None, 4, 4, 512)	0
flatten_1 (Flatten)	(None, 8192)	0
dense_4 (Dense)	(None, 1024)	8389632
dropout_5 (Dropout)	(None, 1024)	0
dense_5 (Dense)	(None, 512)	524800
dense_6 (Dense)	(None, 10)	5130

Total params: 12,756,202
Trainable params: 12,756,202
Non-trainable params: 0

Figura 11: Topologia da segunda rede

Para avaliar os resultados, segue os gráficos de performance durante o treinamento da rede, apresentados pela Fig.[12]. Além disso, têm-se a matriz de confusão da primeira matriz, apresentada pela Fig.[13]. Nesta rede, observa-se um alto grau de paridade entre as variáveis de treinamento e de validação, o que é ótimo quanto à aspectos de qualidade e robustez da rede. Também percebe-se um alto grau de acurácia ao fim do treinamento, de quase 85%, e a menor taxa de perda, como pode ser observado pela Tabela[I]. Além disso, a matriz de confusão se encontra majoritariamente limpa e bem diagonalizada, o que também corrobora para uma boa topologia.



(a) Acurácia por épocas de treinamento (b) Perda por épocas de treinamento

Figura 12: Performance da terceira rede

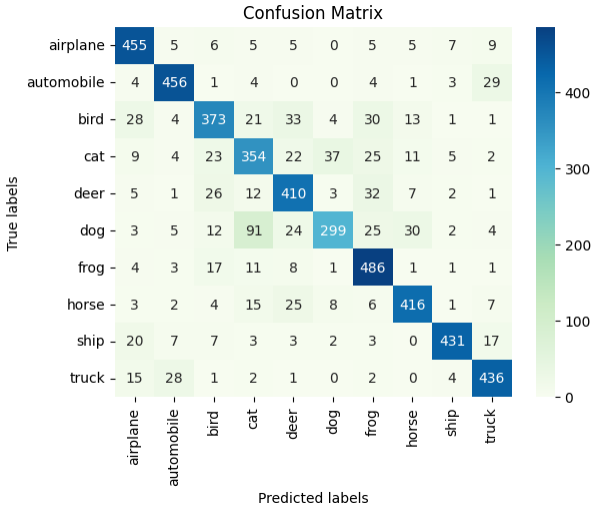


Figura 13: Matriz de confusão da segunda rede

A fins de análise dos filtros gerados, pode-se analisar um dos exemplos de saída das camadas da rede, apresentado pela Fig.[14].

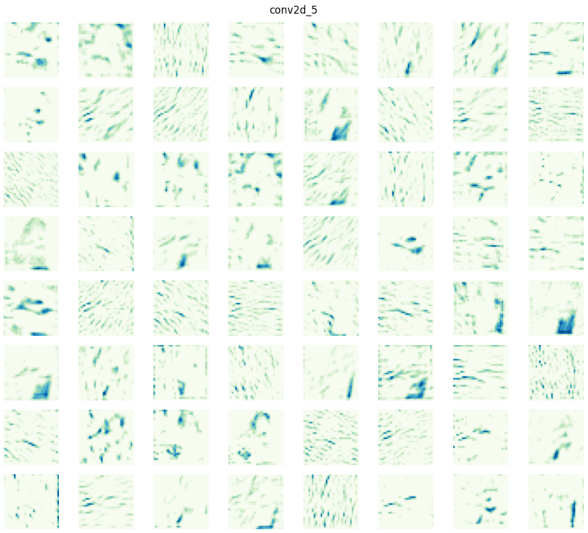


Figura 14: Feature map da topologia 3

E. Análises finais e melhorias

Para finalizar a comparação entre as redes, é apresentada a seguinte tabela contendo as caracteríticas de treinamento finais das redes, bem como o tempo em minutos de treinamento.

	Loss real	Loss	Accuracy real	Accuracy	Tempo(m)
model_1	**1.0578**	**0.8400**	**0.6370**	**0.7029**	9
model_2	**1.0710**	**0.4590**	**0.6774**	**0.8361**	121
model_3	**0.5873**	**0.4466**	**0.8166**	**0.8425**	77

Tabela I: Tabela de comparação de performance das redes

Por fim, foi criara uma nova classificação denominada 'outros', para reclassificar os dados sempre que a confiança fosse menor do que 0.4. Os resultados para a primeira rede foram tais que obteve-se uma acurácia de 0.664, o que piorou a acurácia da rede, porém tornou sua classificação mais robusta. O mesmo foi percebido para a segunda rede, onde a classificação nova foi de 0.7284, um pouco a baixo dos 0.8361 obtidos durante os vários testes. E para a última topologia, obteve-se uma acurácia de 0.8232, apenas levemente abaixo do valor prévio de 0.8425, o que também denota sua robustez natural.

IV. Conclusão

Foram realizadas diversas simulações nos três tipos de topologia apresentados e os três resultados se mostraram ao menos razoáveis para a classificação, com duas topologias com acurácia real acima de 80%, denominando-se boas redes. Também foram realizadas pequenas mudanças para melhorar sua confiança criando uma nova categoria de classificação, o que reduziu um pouco sua acurácia, porém ainda assim manteve as topologias em um bom patamar de performance. O processo como um todo foi bem exaustivo computacionalmente, exigindo diversos dias de simulação para se obter um bom conjunto de dados amostrais, contudo, as redes finais ainda podem ser melhoradas com auxílio de novas tecnologias e camadas a serem usadas, como mais efeitos de data augmentation e otimizadores.

Referências

[1] S. Haykin, "Neural Networks and Learning Machines," 3rd ed., 2009.
[2] Keras official documentation, https://keras.io/