UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

OTHON ALBERTO DA SILVA BRIGANÓ

CLASSIFICAÇÃO DE CARACTERES MANUSCRITOS DA BASE IRONOFF UTILIZANDO *DEEP LEARNING*

PONTA GROSSA

2021

OTHON ALBERTO DA SILVA BRIGANÓ

CLASSIFICAÇÃO DE CARACTERES MANUSCRITOS DA BASE IRONOFF UTILIZANDO *DEEP LEARNING*

IRONOFF dataset Handwritten Characters Classification using Deep Learning

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador(a): Prof. Dr. Erikson Freitas de Morais.

Coorientador(a): Prof.ª Dra. Simone Bello Kaminski Aires

PONTA GROSSA

2021

4.0 Internacional

Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

OTHON ALBERTO DA SILVA BRIGANÓ

CLASSIFICAÇÂO DE CARACTERES MANUSCRITOS DA BASE IRONOFF UTILIZANDO *DEEP LEARNING*

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 25 de novembro de 2021

_____

Erikson Freitas de Morais
Doutorado
Universidade Tecnológica Federal do Paraná

_____

Simone Bello Kaminski Aires
Doutorado
Universidade Tecnológica Federal do Paraná

_____

Hugo Valadares Siqueira
Doutorado
Universidade Tecnológica Federal do Paraná

_____

Ionildo José Sanches
Doutorado
Universidade Tecnológica Federal do Paraná

PONTA GROSSA

2021

Dedico este trabalho a todas e todos que de alguma forma contribuíram para esta pesquisa e para o meu desenvolvimento profissional e pessoal.

AGRADECIMENTOS

Inicio estes agradecimentos me desculpando, pois posso me esquecer de citar alguma pessoa especial. Todas e todos que fizeram parte da minha vida até este momento contribuíram para que este trabalho fosse realizado.

À minha família. Em especial aos meus pais, Elisabete e Gilberto, por terem feito tudo que estava aos seus alcances para que eu chegasse à universidade, pudesse me dedicar aos estudos e por sempre me mostrarem o valor do esforço. Ao meu primo Bernardo, pelos exemplos diretos e indiretos, e por ser uma fonte de inspiração e amizade. À minha companheira Patrícia, pelos tantos anos de companheirismo e apoio; por entender e me dar suporte nos últimos dias deste trabalho, em que muitos deles não me senti capaz.

Aos meus amigos e colegas. Ao Lucas, Caio, Rafael, Guth e Scaketti pelos estudos, risadas, campeonatos de FIFA e histórias. Ao Victor e Rodrigo, grandes amigos que fiz no trabalho e que me ouviram por muitas horas falando desta pesquisa e fazendo piadas de gosto duvidoso.

Ao meu grande amigo e irmão Pedro Botelho, por tantas e tantas horas de estudo, de papo furado, de idas ao estádio, de caronas, reclamações, de muito futebol, cervejas, risadas e apoio em momentos de dúvida. Nesses anos de universidade, passei muito mais tempo contigo do que qualquer outra pessoa citada aqui e tenho certeza que sem a nossa parceira tudo teria sido muito mais difícil.

A todos que fazem a UTFPR acontecer, sendo uma universidade pública, gratuita e de qualidade e que se esforçam para que a educação e a ciência sejam nortes para a nossa sociedade, mesmo quando isso parece cada vez mais difícil.

Ao Prof. Dr. Erikson Freitas de Morais, por todos esses anos de pesquisa juntos e por ser uma referência. À Prof.ª Dra. Simone Bello Kaminski Aires, por ter aceitado participar deste trabalho, mesmo sem me conhecer como aluno, e por todas as ricas colaborações que trouxe para esta pesquisa.

RESUMO

A classificação de caracteres manuscritos tem como objetivo reconhecer, a partir de informações visuais, qual caractere está sendo representado. Essa técnica é utilizada em importantes aplicações como processamento de cheques em bancos e leitura de códigos postais, economizando tempo em uma tarefa tão repetitiva. Esse reconhecimento é dificultado pelas características intrínsecas ao processo de escrita, pois manuscritos se diferenciam de uma pessoa para outra no estilo de escrita, formato e tamanho. Este trabalho realizou a classificação de caracteres manuscritos presentes na base IRONOFF, maiúsculos e minúsculos, utilizando *Deep Learning* e comparando diferentes arquiteturas de redes neurais. Para isso, foi utilizada uma rede VGG adaptada, uma rede LeNet-5 e aplicadas operações de pré-processamento. Os melhores resultados foram obtidos para a VGG adaptada com acurácia de 92,61%, 86,65% e 79,67%, para os subconjuntos de maiúsculas, minúsculas e ambas, respectivamente.

Palavras-chave: classificação; manuscritos; rede neural; IRONOFF.

ABSTRACT

The handwritten character classification aims to recognize, from visual information, which character is being represented. This technique is used in important applications, such as checks processing in banks and zip code reading, which saves time for such repetitive tasks. The intrinsic characteristics of the handwritten process hampers this recognition, once manuscripts are different for each person, in terms of style, format and size. This research carried out a handwritten character classification using the characters present in the IRONOFF dataset, which has uppercase and lowercase characters. The experiment was made by using Deep Learning and by comparing different neural networks architectures. To do so, a convolutional neural network architecture was built, LeNet-5 was implemented and pre-processing operations were applied. The best results were achieved for the proposed network with the accuracy of 92,61%, 86,65% and 79,67%, for the subsets of uppercase, lowercase and both of them, respectively.

Keywords: classification; handwritten; neural network; IRONOFF.

LISTA DE EQUAÇÕES

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3D	Três dimensões
TIFF	*Tag Image File Format*
RL	*Reinforcement Learning*
IA	Inteligência Artificial
CNN	*Convolutional Neural Network*
FC	*Fully Connected*
RNN	*Recurrent Neural Network*
MLPN	*Multi-Layer Perceptron Network*
BLSTM	*Bidirectional Long Short-Term Memory*
BRNN	*Bidirectional Recurrent Neural Network*
LSTM	*Long Short-Term Memory*
GPU	*Graphics Processing Unit*

SUMÁRIO

1 INTRODUÇÃO

A visão computacional é a ciência que, a partir de informações visuais, como imagens e vídeos, identifica e extrai dados como tamanho, cores, formatos, iluminação e distância. Segundo Szeliski (2010), a aplicação das técnicas estudadas por essa ciência é vasta, como vigilância, reconhecimento biométrico, captura de movimentos, construção de modelos 3D, inspeção de máquinas industriais, segurança automotiva e imagens médicas. Segundo Silva *et al.* (2019), a robótica utiliza a visão computacional para tentar simular a visão humana e, juntamente com o aprendizado de máquinas, realizar deduções e executar reações humanas ou atuar de forma inteligente. Esse conjunto de mecanismo é conhecido como inteligência artificial.

Neste sentido, a classificação de caracteres manuscritos tem como objetivo reconhecer, a partir de informações visuais, qual caractere está sendo representado. Essa técnica é utilizada em importantes aplicações como processamento de cheques em bancos, leitura de códigos postais, formulários, reconhecimento de autoria, assinaturas, dentre outras aplicações. Além disso, um computador é capaz de realizar mais tarefas que um humano ao mesmo tempo, portanto esse tipo de aplicação economiza tempo e dinheiro e remove a necessidade de um humano realizar uma tarefa tão repetitiva. Entretanto, esse reconhecimento é dificultado pelas características intrínsecas ao processo de escrita. Segundo Perwej *et al*. (2011), manuscritos se diferenciam de uma pessoa para outra e a diferença no estilo de escrita, formato e tamanho do alfabeto torna o reconhecimento de caracteres uma tarefa complexa.

De acordo com Plamondon *et al*. (2000), há dois tipos de sistemas de reconhecimento de manuscritos: *online* e *offline*. O primeiro deles utiliza dados comumente obtidos através da escrita com uma caneta eletrônica sobre uma superfície digitalizadora. Dessa forma, é possível obter informações como ordem do traçado, velocidade e pressão da escrita. Já no reconhecimento *offline*, a única informação disponível é uma imagem com o caractere escaneado, o que acentua a complexidade do processamento necessário para realizar a classificação e exige técnicas de aprendizagem de máquina para a realização dessa tarefa.

A aprendizagem profunda (*Deep Learning*) é uma técnica de aprendizado de máquina eficaz e precisa, que utiliza grandes quantidades de dados não estruturados. Quaisquer problemas que envolvam reconhecimento de voz, processamento de

imagem, análise de comportamento, entre outras características, podem ser factíveis de utilização de redes neurais de aprendizagem profunda. Dessa forma, o problema de classificação de manuscritos é uma aplicação adequada para o seu uso, já que esses dados não são estruturados, em formato de imagem e variando importantes características na entrada para a mesma classe de saída (Silva *et al*., 2019).

Este trabalho tem como objetivo realizar a classificação de caracteres manuscritos presentes na base IRONOFF, maiúsculos e minúsculos, utilizando redes neurais convolucionais e comparando redes neurais com diferentes arquiteturas, seus resultados e conclusões.

1.1 Objetivos

Esta seção apresenta o objetivo geral e os objetivos específicos deste trabalho.

1.1.1 Objetivo Geral

Implementar CNN (*Convolucional Neural Network*) para a classificação de caracteres maiúsculos e minúsculos da base de dados IRONOFF.

1.1.2 Objetivos Específicos

Para atingir o Objetivo Geral deste trabalho, foram definidos os seguintes objetivos específicos:

- Realizar revisão bibliográfica dos seguintes temas: aprendizagem profunda e seu uso em reconhecimento de manuscritos e base de dados IRONOFF.
- Organizar a base da dados IRONOFF, maiúsculas e minúsculas, em conjuntos de treinamento e teste;
- Avaliar técnicas de pré-processamento para aplicar nas imagens da base de dados;
- Avaliar as redes de aprendizagem profunda, que podem ser utilizadas nos experimentos;
- Realizar experimentos como prova de conceito;

- Analisar os resultados obtidos nos experimentos.

1.2 Organização do trabalho

Este trabalho está divido em seis capítulos. O Capítulo 2 fornece a fundamentação teórica necessária para este trabalho, apresentando conceitos de processamento de imagens, *machine learning*, redes neurais convolucionais e classificação de caracteres manuscritos.

O Capítulo 3 aborda os trabalhos relacionados ao tema de classificação de caracteres manuscritos com redes neurais convolucionais.

O Capítulo 4 trata da metodologia para o desenvolvimento deste trabalho, apresenta o *dataset* utilizado, técnicas de pré-processamento, ampliação e normalização de dados, arquiteturas das redes neurais e tecnologias utilizadas.

O Capítulo 5 apresenta os resultados e as discussões geradas pelos experimentos realizados.

Por fim, o Capítulo 6 descreve as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção tem como objetivo fornecer a fundamentação teórica necessária para o entendimento e desenvolvimento deste trabalho. Nela serão abordados os assuntos de Processamento Digital de Imagens, Visão Computacional, *Machine Learning*, Redes Neurais Convolucionais e Classificação de Caracteres Manuscritos.

2.1 Processamento Digital de Imagens e Visão Computacional

Segundo Marques Filho e Vieira Neto (1999), o interesse pelo estudo do processamento de imagens vem crescendo por viabilizar a construção de novas aplicações que permitem aprimorar a percepção humana de dados visuais e que realizam análises automáticas de informações em uma determinada imagem.

Ainda segundo os autores, um sistema de processamento de imagens é divido em três componentes: aquisição, processamento e saída. A etapa de aquisição consiste em transformar uma imagem, capturada por câmeras ou scanners, em informação digital. Ainda na etapa de aquisição, tem-se a sub etapa de armazenamento, que consiste em como armazenar a informação digital de imagens corretamente, visto que em aplicações que se utilizam imagens pode-se ter terabytes de dados. A etapa de processamento consiste em aplicar algoritmos para se obter determinadas informações daquele dado. A etapa de saída consiste em transformar a saída da etapa de processamento em uma informação visualizável, seja digitalmente, em um monitor de vídeo, ou fisicamente, em uma impressão. Esse processo de converter uma representação digital de uma imagem em imagens visualizáveis é chamada de visualização (VELHO; FRERY; GOMES, 2009).

Velho e Frery e Gomes (2009) definem que no processamento digital de imagens a entrada já é uma imagem, que sofre um determinado processamento e então é transformada em uma nova. Enquanto a Visão Computacional tem como objetivo, a partir de uma imagem digital, obter informações do mundo físico, como dados geométricos e distâncias, extraindo, assim, informações sobre aquela cena representada na entrada.

Algumas das técnicas originadas a partir do processamento digital de imagens e visão computacional são a detecção de bordas, aplicação de texturas e cores,

restauração de imagens, transformações, filtragem e reconhecimento e classificação de símbolos.

Além de reconhecer e analisar as cenas, a Visão Computacional busca aprender com elas. Dessa forma, sua combinação com o aprendizado de máquina mostra-se necessária.

2.2 *Machine Learning*

O Aprendizado de Máquina (*Machine Learning*) é uma disciplina científica que tem como foco entender como os computadores aprendem através dos dados, utilizando a união da Matemática com a Ciência da Computação. O estudo desta ciência é impulsionado pelos desafios computacionais de se construir modelos estáticos a partir da base de dados massivos, que podem conter milhões ou até trilhões de ocorrências (DEO, 2015).

Segundo Silva *et al*. (2019), diversas técnicas são envolvidas para a implementação de aprendizagem de máquina, como algoritmos de mineração de dados, estatística para auxiliar análises e predições de dados, algoritmos de árvores de decisão, processos de agrupamento (*clustering*) e redes Bayesianas.

Ainda de acordo com Silva *et al*. (2019), esta ciência surgiu com o objetivo principal de desenvolver sistemas que aprendem por conta própria, utilizando suas experiências e comportamentos passados, entrada de dados classificados e interação com o ambiente. Existem três formas de aprender e são chamadas, respectivamente, aprendizagem não supervisionada, aprendizagem supervisionada e aprendizado por reforço.

Entre as aplicações desta ciência podem-se citar algoritmos inteligentes para tomada de decisão em crédito bancário, diagnósticos de dispositivos mecânicos, como motores, classificação de objetos celestiais, preenchimento automático de formulários, entre outras (LANGLEY; SIMON, 1995).

2.2.1 Aprendizagem não supervisionada

Uma das formas de aprendizado de máquina, a aprendizagem não supervisionada tem como objetivo encontrar semelhanças e diferenças em um

conjunto de dados, encontrando padrões ocultos ou estruturas que sejam intrínsecas nos dados de entrada (Silva *et al*., 2019).

De acordo com Norvig e Russel (2013), neste tipo de aprendizagem, o sistema é capaz de aprender os padrões na entrada de dados mesmo que não sejam fornecidos feedbacks explícitos, que indiquem se o sistema está aprendendo corretamente. Um dos exemplos abordados pelo autor é um agente de táxi, no qual o sistema pode desenvolver conforme aprende o que é um dia de tráfego "bom" e um dia de tráfego “ruim”, mesmo que nunca tenha sido exemplificado de maneira explícita tais conceitos. Esse tipo de tarefa é conhecido como agrupamento.

Segundo Coppin (2010), o Mapa de Kohonen é um bom exemplo de aprendizagem de máquina não supervisionado. O Mapa de Kohonen é uma rede neural capaz de classificar os dados de entrada sem receber rótulos indicando o grupo correto e quais são as categorias de classificação. Ainda segundo o autor, este método de aprendizagem é útil em situações nas quais precisa-se classificar ou agrupar dados de entrada, mas não se conhece previamente quais são as classes presentes.

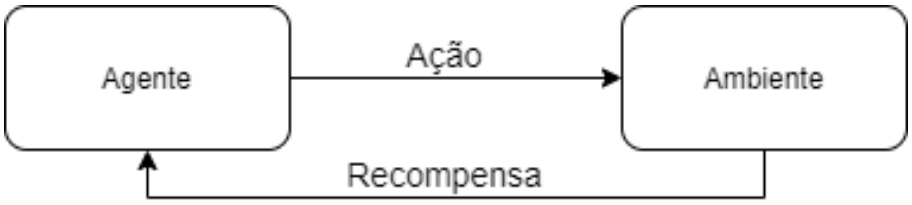
2.2.2 Aprendizagem por reforço

A aprendizagem por reforço, *Reinforcement Learning (RL),* tem como objetivo utilizar feedbacks explícitos e experiências para melhorar a assertividade de um determinado sistema que realiza ações contínuas determinadas por ele.

De acordo com Coppin (2010), um sistema que utiliza RL receberá um reforço positivo, chamado de recompensa, cada vez que realizar uma ação correta no ambiente, e um reforço negativo cada vez que tomar uma decisão equivocada, como é ilustrado na Figura 1.

O autor utiliza como exemplo um robô que tem como tarefa pegar um objeto. Ao pegar o objeto corretamente, ele recebe um reforço positivo. Dessa forma, será capaz de entender que aquela ação foi a correta.

Figura 1 - Diagrama que ilustra o funcionamento de um sistema de aprendizagem por reforço



Fonte: Autoria própria

2.2.3 Aprendizagem supervisionada

Segundo Coppin (2010), redes que utilizam aprendizagem supervisionada aprendem ao terem contato com dados de treinamento já classificados, ajustando seu comportamento a partir das classes já informadas no momento da entrada de dados. Neste tipo de sistema, cada instância da entrada é um par contendo o dado e seu respectivo rótulo, que o classifica entre as classes possíveis.

Essa técnica é utilizada em redes neurais, que utilizam desse tipo de aprendizagem para modificar os pesos de suas conexões de forma a maximizar a acurácia de suas predições. A aprendizagem supervisionada pode ser analisada como um problema de otimização (HAYKIN, 2009).

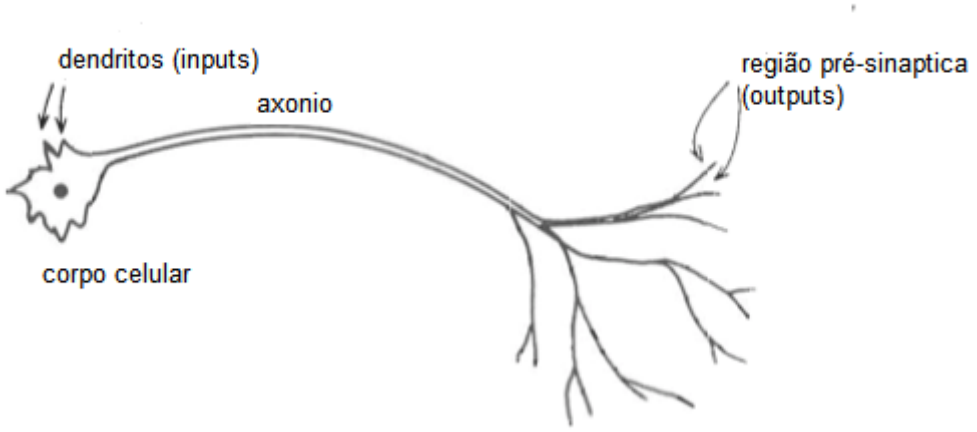
2.2.4 Redes Neurais Artificiais

As redes neurais são uma subárea do Aprendizado de Máquinas. Esta ciência visa desenvolver sistemas inteligentes inspirados pelo comportamento do sistema nervoso dos organismos superiores, simulando as atividades dos neurônios, aprendendo e agindo como tal, utilizando modelos matemáticos (Silva et al., 2019).

Uma rede neural biológica é composta por células nervosas, chamadas de neurônios, que são interconectadas. A computação de informação nos neurônios se dá no corpo da célula. Um sinal elétrico é responsável por transportar a informação de um neurônio para outro através dos axônios, que são ligados a outros neurônios na rede neural por meio de terminais, como mostrado na Figura 2. A esse processo de comunicação entre neurônios dá-se o nome de sinapse neural (GRAUPE, 2007). Embora cada neurônio seja, de maneira isolada, extremamente simples, a enorme

quantidade deles presentes na rede neural torna capaz o processamento de informações complexas e com uma grande velocidade (COPPIN, 2010).

Figura 2 - Diagrama de um neurônio biológico



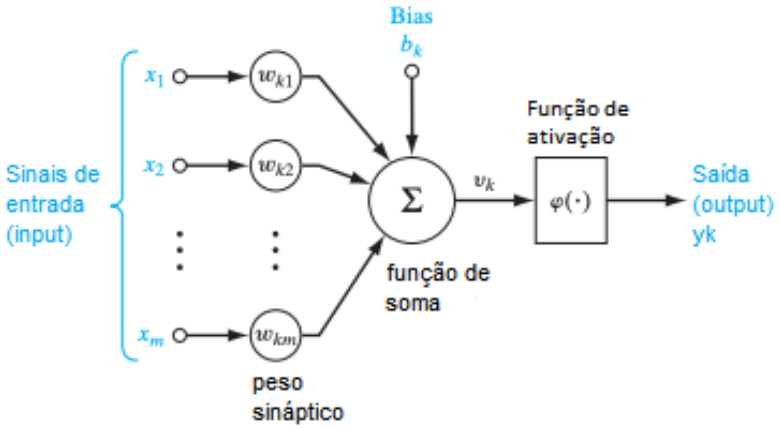
Fonte: Adaptado de Graupe (2007, p. 24)

Assim como no cérebro humano, redes neurais artificiais contêm uma grande quantidade de neurônios artificiais conectados, que realizam comunicação uns com os outros. Em cada uma dessas conexões há um peso, que determina quanto de conhecimento (valor) será passado ao próximo componente neural.

Segundo Haykin (2009), os elementos básicos de um modelo neural são neurônios conectados uns aos outros, contendo um peso sináptico. Desse modo, cada sinal de entrada de uma sinapse, conectado a um neurônio, é multiplicado por um peso sináptico. Após o processo de multiplicação, cada sinal de entrada é somado para originar um único valor, que é somado por um *bias* (viés), que também contém um peso ajustável. Esse comportamento é ilustrado na Figura 3, que apresenta um neurônio artificial genérico.

Um conceito fundamental para o aprendizado é a plasticidade sináptica. Essa plasticidade permite com que as conexões sinápticas se alteram com base nas atividades neurais. Nas redes neurais artificias essa plasticidade diz respeito aos ajustes nos pesos sinápticos de cada neurônio (HAYKIN, 2009).

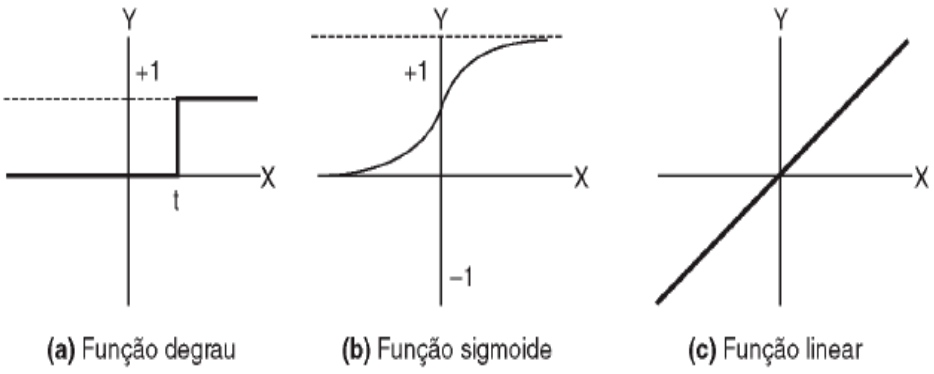
Figura 3 - Diagrama de um neurônio genérico em uma rede neural artificial



Fonte: Adaptado de Haykin (2009, p. 12)

Cada um desses neurônios recebe entradas de dados e uma função, conhecida como função de ativação, aplica essas entradas e calcula qual será o valor de aceitação deste neurônio, o que é conhecido como nível de ativação (COPPIN, 2010). Na Figura 4, é possível observar algumas das funções de ativação mais comuns: degrau, sigmoide e linear. Além das funções de ativações apresentadas na Figura 4, outra função de ativação comumente usada é a ReLU (*rectified linear unit*). Sua principal característica é que esse função retorna valor 0 para qualquer valor negativo e o próprio valor para casos positivos.

Figura 4 - Funções de ativação



Fonte: (COPPIN, 2009)

A dificuldade de tratar problemas complexos, com dados pouco estruturados e com características diversas impulsionaram o surgimento de novas técnicas de *machine learning*, como o *Deep Learning.*

2.2.5 Aprendizagem profunda

Segundo Goodfellow e Bengio e Courville (2016), o desenvolvimento das técnicas de Aprendizagem Profunda (*Deep Learning*) foram motivadas em partes pela falha de algoritmos tradicionais de IA (Inteligência Artificial) em generalizar bem certas tarefas, como reconhecimento de voz e reconhecimento de objetos, bem como trabalhar com quantidades massivas de novos dados coletados a todo momento.

De acordo com Lecun e Bengio e Hinton (2015), esses algoritmos têm limitações em lidar com dados naturais em formato original, necessitando assim de transformações para processá-los e gerar conhecimento.

Segundo Silva *et al.* (2019), a *Deep Learning* é uma técnica eficaz de aprendizado de máquina, que se utiliza de *datasets* com grande quantidade de dados não estruturados, utilizando técnicas para que o aprendizado de padrões ocorra. Dessa forma, tarefas como processamento de imagens, reconhecimento de voz, análise de comportamentos, entre outros, podem ser atacadas utilizando Aprendizagem Profunda.

2.2.6 Redes Neurais Convolucionais

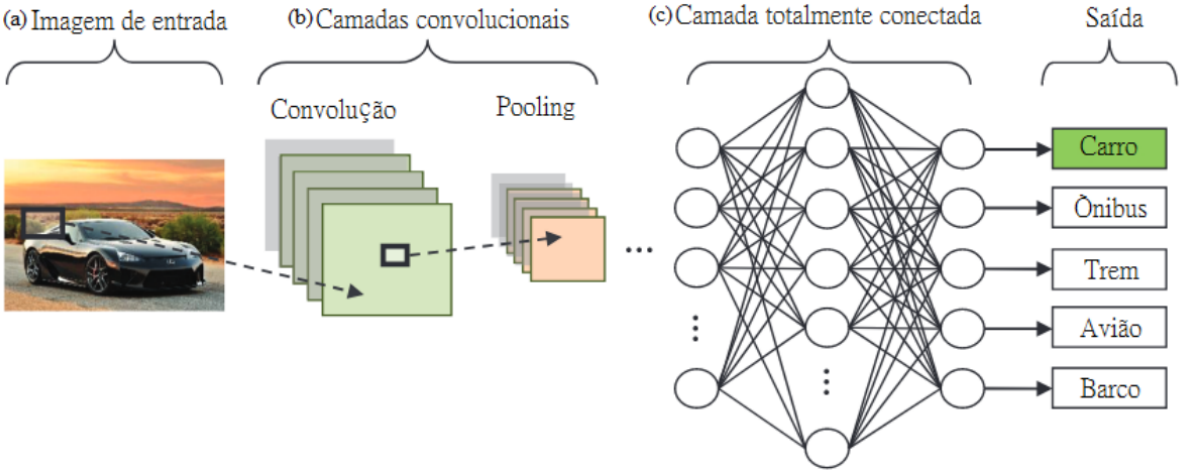
Redes Neurais Convolucionais, também chamadas de CNN (*Convolutional Neural Network*) são um tipo de especialização de rede neural para processamento de dados em formato de grade e/ou *arrays*. Exemplos desse tipo de estruturação de dados são séries temporais, armazenadas em formato de *array* unidimensional, e imagens, representadas por matrizes bidimensionais ou tridimensionais, nas quais cada posição contém o valor de um pixel (GOODFELLOW; BENGIO; COURVILLE, 2016). Em razão dessas características, CNNs são largamente aplicadas em processamento de imagens.

De maneira geral, uma Rede Neural Convolucional é composta por três tipos de camadas: camada convolucional, camada de pooling e camada completamente

conectada. Na Figura 5, é possível visualizar um exemplo de uma CNN e suas camadas destacadas.

Uma das principais características de uma rede neural convolucional é que, a cada camada, diferentes filtros e processamentos são aplicados a fim de acentuar as características relevantes da entrada de dados, modificando assim seus valores e dimensões.

Figura 5 - Pipeline de classificação de uma imagem em uma CNN



Fonte: (HAN, 2020)

A rede se inicia recebendo uma imagem de entrada (a) e as primeiras camadas são as convolucionais (convolução e pooling) (b), seguidas da camada totalmente conectada (c) que tem como saída a classe predita para aquela determinada entrada.

2.2.6.1 Camada Convolucional

Esta seção tratará da camada convolucional, presente em uma rede neural convolucional. Para melhor entendimento, primeiro será definido o que é uma convolução e então a aplicação em uma CNN.

A convolução é uma operação usada largamente no processamento digital de imagens. Seu funcionamento básico consiste em uma operação matemática entre duas funções (entrada) que dá resultado a uma terceira função (saída). Matematicamente, a convolução é o somatório do produto das duas funções de entrada, ao longo de toda a região das funções.

Quando se trata de processamento digital de imagens, as duas funções de entrada são a imagem e o filtro aplicado e a função de saída é a imagem filtrada. Ao trabalhar com imagens, chama-se os *kernels* de filtros, matrizes normalmente de pequenas dimensões que são multiplicadas por toda a entrada de dados (O'SHEA; NASH, 2015). Ou seja, a imagem de saída é o somatório do produto dos pixels da imagem multiplicados pelo *kernel*, utilizando o deslizamento para que todos os pixels da imagem sejam utilizados.

Equação 1 - Convolução matemática

$$(\boldsymbol{f} * \boldsymbol{g})(\boldsymbol{k}) = \boldsymbol{h}(\boldsymbol{k}) = \sum_{\boldsymbol{j}=\boldsymbol{0}}^{\boldsymbol{k}} \boldsymbol{f}(\boldsymbol{j}) \cdot \boldsymbol{g}(\boldsymbol{k} - \boldsymbol{j}) \qquad (\mathbf{1})$$

Fonte: Autoria própria (2021)

A Equação 1 apresenta a definição matemática de uma convolução. Entretanto, quando trabalhamos com imagem tem-se, comumente, imagens de duas dimensões. Dessa forma, a equação é modificada como apresenta a Equação 2, na qual f é o *kernel* e g a imagem de entrada.

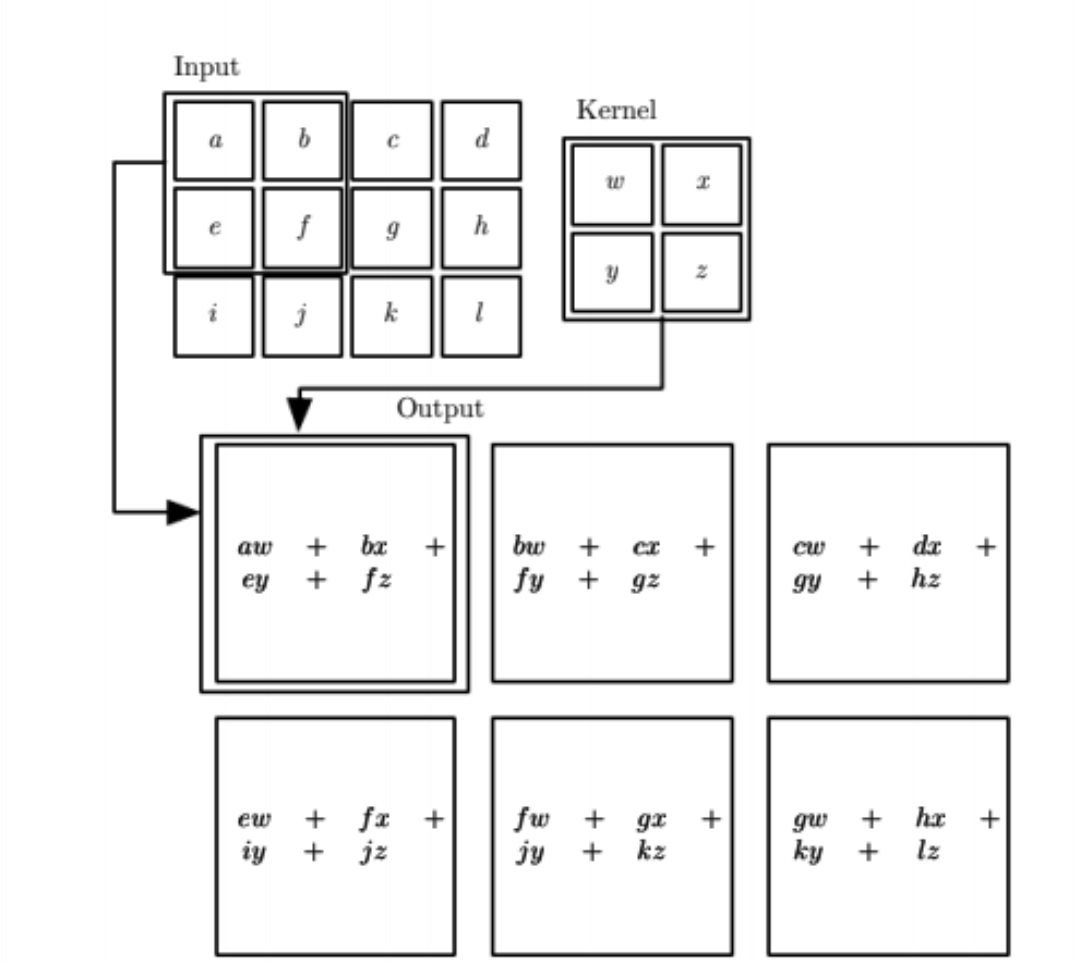
Equação 2 - Convolução para uma imagem com duas dimensões

$$(f * g)(x.y) = h(x,y) = \sum_{j=-a}^{a} \sum_{t=-b}^{t} f(j,t) \cdot g(x-s, y-t) \qquad (2)$$

Fonte: Autoria própria (2021)

Assumindo uma imagem de duas dimensões como entrada, cada posição da matriz será multiplicada por um *kernel* e, então, calculado um novo valor. Essa operação é ilustrada na Figura 6.

Figura 6 - Exemplo da aplicação de um *kernel* 2x2 em uma entrada 4x3



Fonte: (GOODFELLOW; BENGIO; COURVILLE, 2016)

Como pode-se observar na Figura 6, cada região da entrada (input) é multiplicada pelo *kernel* e então dá resultado a novos valores na imagem de saída (output). O Filtro Sobel, por exemplo, é utilizado para detecção de bordas e é um exemplo do uso de convolução em processamento de imagens digitais (FREDJ et al., 2017)

Nesse filtro, comumente utiliza-se de um *kernel* 3x3, ou 5x5, que, quando aplicado na imagem de entrada, realça as bordas, removendo pixels com valores de intensidade menores que um determinado valor de referência. Uma aplicação deste filtro é mostrada na Figura 7, na qual é possível visualizar que após sua aplicação, as bordas são realçadas enquanto o resto da imagem é atenuado.

Uma camada convolucional presente em uma CNN contém um conjunto de filtros que são aplicados à imagem de entrada. Essa camada atua como um extrator

de características, pois é capaz de aprender as representações relevantes de uma entrada (RAWAT; WANG, 2017).

Figura 7 - Aplicação do filtro Sobel

Fonte: (FREDJ *et al.*, 2017)

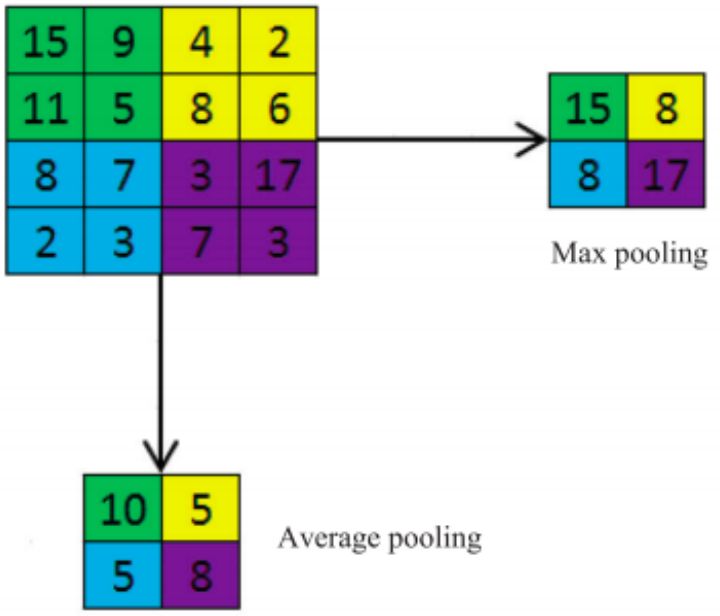
2.2.6.2 Camada de *Pooling*

As camadas de *pooling* têm como objetivo reduzir gradualmente a dimensionalidade de um mapa de características, produzida pela camada anterior (comumente, uma camada convolucional). Dessa forma, diminui-se o número de parâmetros e complexidade de processamento computacional (O'SHEA; NASH, 2015).

Além disso, segundo Lecun e Bengio e Hinton (2015), a utilização de uma camada de *pooling* ajuda a tornar a representação relativamente invariante em relação à translação do dado de entrada. Isso ocorre porque a maior parte dos dados de saída do *pooling* não mudam. Ainda segundo o autor, a invariância em relação à translação local é relevante quando se deseja saber se uma determinada característica está presente na entrada, porém a localidade exata não é relevante. Por exemplo, ao analisar se há um rosto em uma imagem, podemos verificar se há um olho do lado esquerdo e outro do lado direito, entretanto a localidade exata não é relevante.

Em camadas de *pooling*, podem ser aplicadas diversas funções que afetam o resultado dos novos pixels. As mais utilizadas são *Max Pooling* e *Average Pooling*. O *Average Pooling* consiste em calcular a média dos valores da região de *pooling* e, então, utilizar este valor como o valor do novo pixel na saída desta camada. O *Max Pooling* seleciona, dentre os valores da matriz da região, o maior valor e, então, assume aquele valor como representante do novo pixel. Ambas funções podem ser observadas na Figura 8.

Figura 8 - Exemplo de *Average Pooling* e *Max Pooling*



Fonte: (RAWAT; WANG, 2017)

2.2.6.3 Camada Totalmente Conectada

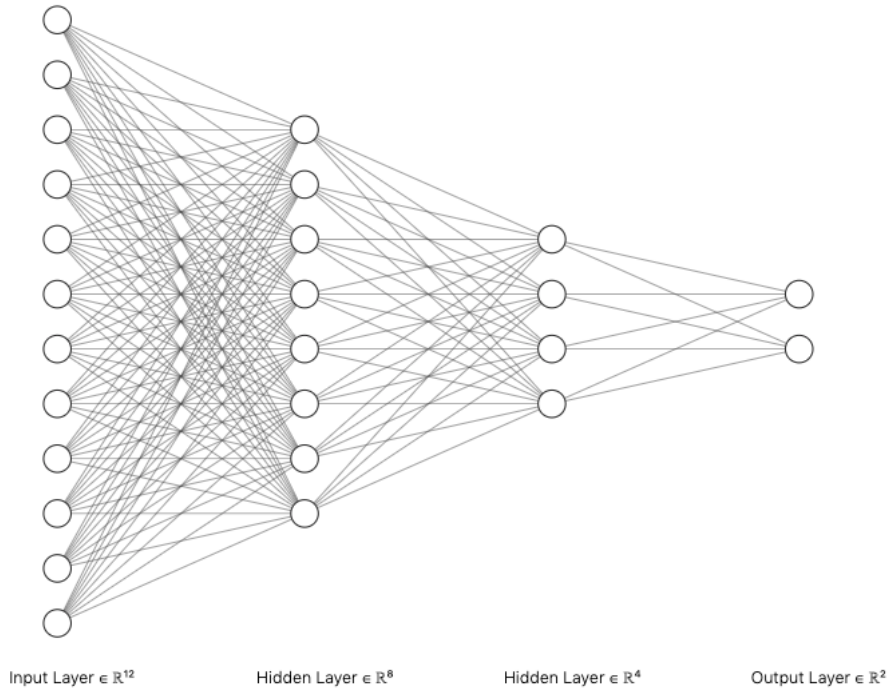
Segundo Zhang *et al.* (2017), a Camada Totalmente Conectada, também chamada de FC (*Fully Connected*), é uma das mais importantes em uma rede neural convolucional e que requer milhões de parâmetros, chegando a ter a maioria dos parâmetros de uma rede. Nesta camada, a cada execução seus pesos são inicializados com valores aleatórios e os ajustes desses valores são realizados através de *backpropagation*.

Segundo Rawat e Wang (2017), em uma Camada Totalmente Conectada cada um dos neurônios de uma camada é ligado a todos os neurônios da camada seguinte, até que se chegue à última camada, que contém o número de neurônios igual ao número de classes que se deseja realizar a classificação da entrada. Na Figura 9, pode-se observar um exemplo de Camada Totalmente Conectada, na qual há duas classes para classificação.

Por conta da alta de parâmetros desta camada em relação a toda a rede, a Camada Totalmente Conectada pode causar *overfitting* (XU; PAN, 2017), fenômeno que ocorre quando a rede aprende a identificar extremamente bem os dados de

treinamento, porém, com isso, perde a capacidade de generalizar, ou seja, ter um bom índice de assertividade para entradas que a rede nunca viu.

Figura 9 - Exemplo de Camada Totalmente Conectada



Fonte: Autoria própria (2021)

2.2.6.4 *Dropout*

Segundo Jabbar e Khan (2014), o fenômeno de sobretreinamento é um problema chave em aplicações de aprendizagem supervisionada. Esse comportamento ocorre quando o algoritmo aprende tão bem o conjunto de entradas que absorve e memoriza os ruídos e peculiaridades dos dados. Dessa forma, este comportamento atrapalha a generalização das propriedades do modelo e, portanto, o resultado do treinamento do algoritmo é prejudicado quando aplicado a outros dados que não pertencem aos conjuntos de treinamento.

Em razão do crescimento do uso de *Deep Learning* em tarefas de aprendizagem de máquina, a técnica de *Dropout* tem sido usada para prevenir que Redes Neurais com vários parâmetros causem sobretreinamento (PHAM; et al., 2014).

A técnica de *Dropout* foi apresentada por Srivastava et al. (2014) como um método de regularização dos dados de uma rede neural durante seu treinamento como forma de prevenir o sobretreinamento. Este método atua descartando aleatoriamente uma quantidade pré-definida de neurônios durante o processo de treinamento, ajudando a prevenir que a rede aprenda ruídos (SRIVASTAVA et. al., 2014).

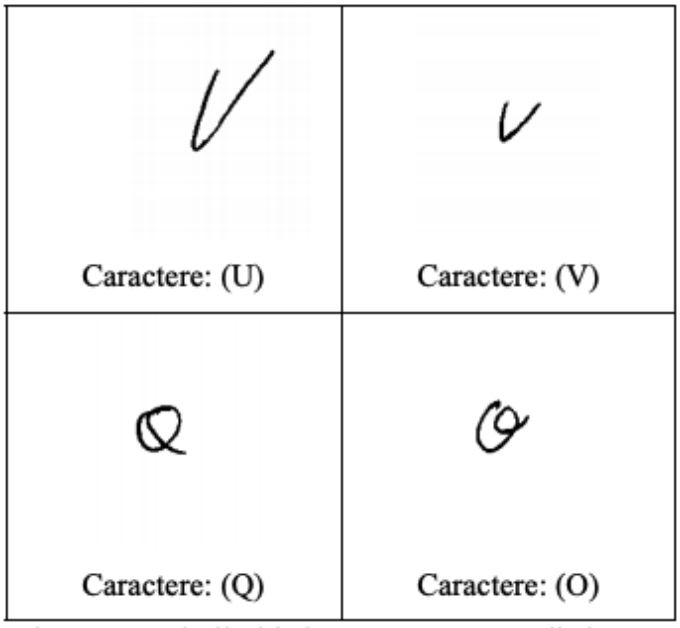
2.3 Classificação de Caracteres Manuscritos

O problema da classificação de caracteres manuscritos consiste em, através de informações, online ou offline, identificar qual é o caractere escrito à mão em uma determinada entrada. Essa classificação é dificultada pelo fato de cada pessoa escrever o mesmo caractere de diferentes maneiras.

Dessa forma, o algoritmo, ou conjunto de algoritmos, responsável por aprender qual o caractere que está sendo fornecido como entrada encontra maiores dificuldades para identificar quais são as características que identificam aquela classe.

Segundo Aires (2005), além das dificuldades em relação à diferenciação de estilo de escrita de cada pessoa, além de a mesma pessoa escrever o mesmo caractere de maneiras diferentes, letras como V, U, Q e O possuem similaridades entre si, como mostrado na Figura 10. Dessa forma, o uso de aprendizagem profunda para classificação de caracteres manuscritos tem importante aplicação.

Figura 10 - Similaridade entre caracteres distintos



Fonte: (AIRES, 2005)

Por conta desses desafios, técnicas de *Deep Learning* são utilizadas para realizar a classificação de manuscritos de diferentes idiomas e se mostram eficientes.

2.4 Métricas de avaliação

Neste trabalho são utilizadas duas métricas para avaliação das redes neurais: acurácia e perda, além do uso da matriz de confusão para análise da classificação de cada caractere.

O valor de acurácia de uma rede, mostrado na Equação 3, é calculado a partir do número de amostras para classificação que obtiveram a predição correta. Ou seja, a quantidade total de amostras que foram classificadas corretamente em um determinado teste. Esse valor pode ser utilizado para calibrar os parâmetros de uma rede neural após cada época, buscando obter uma acurácia mais alta

As redes deste trabalho utilizam essa métrica para determinar o fim do processo de treinamento: ao passar por quinze épocas em sequência sem apresentar melhoria na acurácia, o treinamento é encerrado.

Equação 3 - Fórmula da acurácia

$$acuracia = \frac{predicoes\ corretas}{total\ de\ predicoes} \quad (3)$$

Fonte: Autoria própria (2021)

O valor de perda mostra a soma das diferenças entre os valores esperados e os valores reais. Dessa forma, quanto maior o valor de perda, maior é quantidade de erro presente em uma classificação. Por isso, deseja-se que essa métrica seja a menor possível. Neste trabalho, foi utilizada a função de perda de *cross entropy loss*, apresentada na Equação 4, na qual y é a classe correta e $\hat{y}$ a classe predita, para m predições.

Equação 4 - Função de perda *cross entropy loss*

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i) \quad (4)$$

Fonte: Autoria própria (2021)

A matriz de confusão é uma matriz que exibe a frequência com que cada classe obteve uma determinada classificação. Com essa matriz, é possível identificar se as classes estão sendo classificadas corretamente, quantas classificações foram errôneas e quais caracteres são mais confundidos entre si.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta pesquisas relacionadas à utilização de redes neurais para a classificação de manuscritos, para diversos datasets em diferentes idiomas.

A pesquisa de Pal e Singh (2010) classifica caracteres da língua inglesa, em letras maiúsculas presentes em um dataset próprio. Para isso, primeiramente o algoritmo utiliza-se da técnica de esqueletização, que consiste em afinar traços da imagem, transformando os traçados do caractere em traçados com apenas um pixel de largura. Após a esqueletização, há uma normalização para que cada imagem tenha dimensão 30x30 e para que o caractere fique no topo esquerdo da imagem. Em seguida, ocorre a extração de características com Fourier descriptors, a classificação utilizando a rede neural Perceptron de múltiplas camadas (MLP) e então o reconhecimento e classificação. Nos experimentos, foi alcançado o resultado de 94% de acurácia.

Na pesquisa de Sutha e Ramaraj (2007), tem-se como objetivo a classificação de caracteres Tamil manuscritos, utilizando redes neurais e informações offline. O alfabeto Tamil, do sul da Índia, é composto por 18 consoantes, 12 vogais e um caractere especial. Cada vogal e consoante pode ser combinada, gerando assim 247 caracteres no total. Na etapa de pré-processamento, Sutha e Ramaraj (2007) aplicam o filtro gaussiano para suavização, seguido da aplicação do método Otsu's de thresholding e da técnica de esqueletização. Assim como na pesquisa de Pal e Singh (2010), ocorre a extração de característica com Fourier descriptors e a classificação com a rede MLPN. Ao final dos experimentos, chegou-se ao resultado de 97% de acurácia.

Na pesquisa apresentada por Yanmei e Bo e Zhaomin (2020), uma rede neural convolucional baseada na LeNet-5 foi construída. Nesse trabalho, foi utilizado o dataset MINST, que é amplamente utilizado para pesquisas de reconhecimento de caracteres e contém caracteres manuscritos de numerais arábicos. A rede neural desenvolvida durante o trabalho é totalmente baseada na LeNet-5, criada por LeCun *et al*. (1989). As modificações são o aumento no número de kernels convolucionais, a utilização da função de ativação ReLU, a substituição da quinta camada convolucional, a remoção da camada de subsampling e a transformação da terceira camada convolucional em uma camada totalmente conectada. Obteve-se com a rede neural VGG adaptada uma acurácia de 99,3%, contra 99,1% da LeNet-5.

Na pesquisa desenvolvida por Rahman et al. (2015), é apresentada uma nova arquitetura de rede neural convolucional chamada BHCR-CNN, para a classificação de caracteres manuscritos do alfabeto bengali. Na pesquisa, foram utilizados 50 caracteres desse alfabeto. A rede proposta é composta por duas camadas convolucionais, com kernel de 5x5, e duas camadas de Average Pooling 2x2. Nos experimentos realizados, atingiu-se acurácia de 85,96%.

No trabalho de Yuan et al. (2012), ocorre a classificação de caracteres manuscritos da língua inglesa. Para isso, a pesquisa utiliza o dataset UNIPEN, que, assim como o IRONOFF, contém 26 caracteres maiúsculos e 26 caracteres minúsculos. A rede neural convolucional desenvolvida é uma versão modificada da LeNet-5, alterando número de neurônios, gerando diferente redes para testes, e conexão entre a segunda e quarta camada convolucional. A melhor acurácia obtida neste trabalho foi 89,17%.

A pesquisa de Pradeep e Srinivasaan e Himavathik (2011) apresenta uma rede neural desenvolvida com três camadas, para a classificação de um dataset próprio de caracteres manuscritos sem o uso de extração de características. O processo empregado no trabalho se divide em aquisição de imagens, pré-processamento, segmentação e classificação. A melhor acurácia obtida, para todo o dataset, foi de 90,19%.

A pesquisa apresentada por Maalej e Kherallah (2018) ataca o desafio de classificar caracteres arábicos. Para isso, foi construído um modelo híbrido, que combina CNN com BLSTM. A BLSTM (Bidirectional Long Short-Term Memory), é uma rede neural recorrente bidirecional que se baseia na LSTM (Long Short-Term Memory. Com essa arquitetura híbrida, a pesquisa chegou a uma acurácia de 92,21%, para a base de dados IFN/ENIT.

Em sua pesquisa, Elsawy e Loey e El-Bakry (2017) desenvolveram uma rede neural convolucional para classificar caracteres manuscritos arábicos. A arquitetura desenvolvida contém duas camadas convolucionais, com camada de pooling após cada convolucional, seguidas de duas camadas totalmente conectadas, com 1024 e 28 neurônios. Classificando um dataset próprio, esse trabalho apresentou acurácia de 94,9%.

Na pesquisa de Nam e Hung (2019), é apresentada uma arquitetura híbrida para o problema de reconhecimento de manuscritos na língua japonesa. Para esse propósito, a arquitetura é desenvolvida com uma CNN e uma RNN. A rede neural

convolucional desenvolvida segue uma arquitetura similar a VGG16. Já a etapa de RNN (Recurrent Neural Network) utiliza uma LSTM (Long Short-Term Memory). Essas redes são redes neurais recorrentes que têm como principal característica maior capacidade de memória. Para cumprir esse objetivo, suas camadas escondidas (hidden layers) contêm sub-redes recorrentes conectadas, chamadas de memory-blocks (blocos de memórias) (GRAVES; et al., 2009). Com essa arquitetura, a pesquisa atingiu uma acurácia máxima de 96,41% com a base de dados ETL.

Por fim, a pesquisa desenvolvida por Rabby (2018) busca reconhecer manuscritos na língua Bangla, língua mãe de Bangladesh, de três diferentes bases de dados: BanglaLekha-Isolated, CMATERdb e ISI. A rede desenvolvida por esse trabalho consiste em uma CNN de 13 camadas e obteve os seguintes resultados em relação à acurácia: 98% para o dataset CMATERdb; 96,81% para o dataset ISI; 95,71% para o dataset BanglaLekha-Isolated e 96,40% para um dataset construído a partir da mistura dos três datasets citados.

Os trabalhos citados nesta seção se relacionam com este trabalho por realizarem a classificação de caracteres manuscritos utilizando Redes Neurais Convolucionais. Um comparativo dos trabalhos pode ser observado na Tabela 1.

Tabela 1 - Comparativo de trabalhos relacionados

(continua)

Autor	Rede neural	*Dataset*	Acurácia (%)
Pal e Singh (2010)	MLNP	Própria	94%
Sutha e Ramaraj (2007)	MLPN	Própria	97%
Yanmei e Bo e Zhaomin (2020)	Própria, baseada na LeNet-5	MINST	99,3%
Rahman *et al.* (2015)	BHCR-CNN	Própria	85,96%
Yuan *et al*. (2012)	Própria, baseada na LeNet-5	UNIPEN	89,17%

Fonte: Autoria Própria (2021)

Tabela 1 – Comparativo de trabalhos relacionados

(conclusão)

Trabalho	Rede neural	Base de dados	Acurácia

Pradeep e Srinivasaan e Himavathik (2011)	CNN própria	Própria	90,19%
Maalej e Kherallah (2018)	CNN + BLSTM	IFN/ENIT	92,26%
Elsawy e Loey e El-Bakry (2017)	CNN própria	Própria	94,9%
Nam e Hung (2019)	CNN (baseada na VGG16) + LSTM	ETL	96,41%
Rabby (2018)	Própria	BanglaLekha-Isolated, CMATERdb, ISI e *dataset* combinando os três	98%, 96,81%, 95,71% e 96,40%, respectivamente

Fonte: Autoria Própria (2021)

4 DESENVOLVIMENTO

Este trabalho implementa redes neurais convolucionais para a classificação de caracteres maiúsculos e minúsculos da base de dados IRONOFF. Ainda, faz-se a comparação entre duas arquiteturas de redes neurais, realiza-se testes com e sem *data augmentation* divididos em outros três datasets: apenas com caracteres minúsculos, apenas com caractere maiúsculos e ambos. Dessa forma, tem-se no total 12 experimentos para análise, como exibido na Tabela 2.

Tabela 2 - Experimentos realizados neste trabalho

Experimento	*Data Augmentation*	Conjunto	Rede
1	Não	Minúsculas	VGG adaptada
2	Não	Maiúsculas	VGG adaptada
3	Não	Minúsculas e maiúsculas	VGG adaptada
4	Não	Minúsculas	LeNet-5
5	Não	Maiúsculas	LeNet-5
6	Não	Minúsculas e maiúsculas	LeNet-5
7	Sim	Minúsculas	VGG adaptada
8	Sim	Maiúsculas	VGG adaptada
9	Sim	Minúsculas e maiúsculas	VGG adaptada
10	Sim	Minúsculas	LeNet-5
11	Sim	Maiúsculas	LeNet-5
12	Sim	Minúsculas e maiúsculas	LeNet-5

Fonte: Autoria Própria (2021)

Nesta tabela, é possível observar que existem 6 datasets, que serão detalhados na seção 4.4. Cada uma dessas bases de dados será utilizada pelas duas redes neurais convolucionais, chegando assim à 12 experimentos.

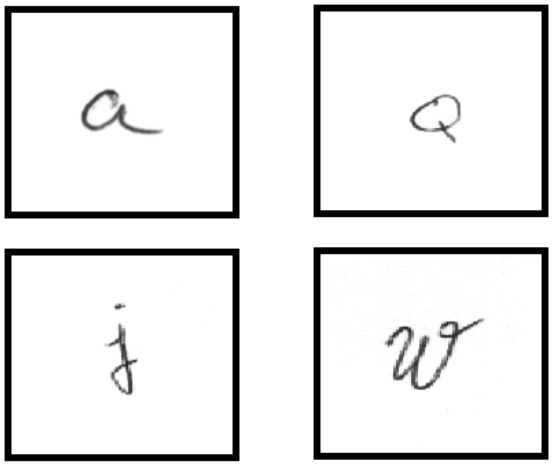
Este capítulo irá detalhar como este trabalho foi desenvolvido para gerar os dados para tais comparações, abordando tópicos como base de dados, data augmentation da base de dados e criação e utilização das arquiteturas das redes neurais.

4.1 Base de dados

Para a realização dos experimentos, é utilizada a base de dados IRONOFF. Este *dataset* foi desenvolvido por Viard Gaudin *et al.* (1999) e contém imagens de caracteres e palavras manuscritas, com informações *online* e *offline*.

Este trabalho utiliza o conjunto de dados *offline*, que são imagens em formato TIFF (*Tagged Image File Format*) na resolução 167px x 214px. No total, esse subconjunto tem 37706 imagens, dividas em 550 pastas, cada uma delas com manuscritos de uma pessoa diferente. Na Figura 11, é possível visualizar alguns exemplos de caracteres presentes na base de dados. Originalmente, as imagens têm fundo branco e traços pretos. Em algumas figuras durante este trabalho, as cores foram invertidas, para melhor visualização.

Figura 11 - Exemplo de imagens do *dataset* IRONOFFF



Fonte: Adaptado de Viard-Gaudin (1999)

4.2 *Data Augmentation*

Durante os experimentos, a quantidade de imagens presentes no *dataset* original IRONOFF mostrou-se insuficiente para uma boa generalização das CNNs, como será abordado na seção 5.

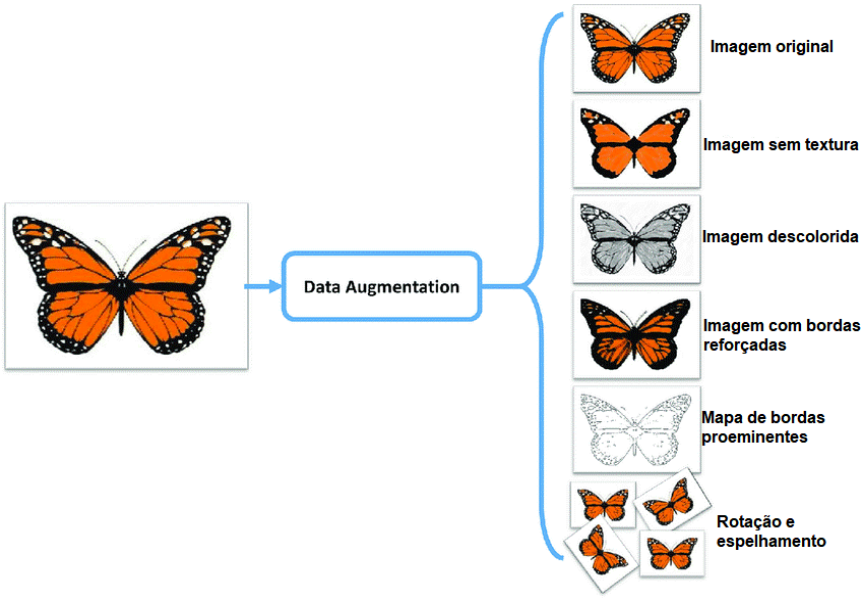
Este fato ocorre porque o número de dados apresentados à rede gerou baixo valor de acurácia, mostrando não ser suficiente para que a mesma aprenda a identificar e generalizar os padrões que caracterizam as classes presentes na base

de dados, e que serão testadas posteriormente. Cada classe presente na base de dados original tem, em média, 410 imagens. Desta forma foi analisado a possibilidade de aplicar *data augmentation* para verificar se o poder de generalização seria melhor em comparação ao uso da base original.

Segundo Goodfellow e Bengio e Courville (2016), a melhor maneira de fazer com que um modelo de *machine learning* generalize melhor é fornecer a ele mais dados para treinamento. Porém, ao se utilizar de um *dataset*, a quantidade de amostras é limitada a partir do momento de sua criação. Para contornar esse problema, a partir dos dados originais cria-se mais entradas, artificiais, e as adicionam no *dataset* de treinamento original. Dessa maneira, é possível mostrar ao algoritmo de treinamento mais exemplos por classe, mantendo característica dos dados reais, já que essa técnica gera novos dados a partir de entradas do *dataset* original. Essa técnica é conhecida como *data augmentation*, ou ampliação de dados.

Quando o *data augmentation* é utilizado em um contexto de processamento de imagens, comumente são aplicadas operações como rotação, *zoom in* e *zoom out*, *shifting*, mudança de paleta de cores e espelhamento. Dessa forma, com uma única imagem pode-se gerar novas imagens, com diferentes características, gerando assim mais entradas para o processo de treinamento da rede. Algumas dessas operações podem ser observadas na Figura 12.

Figura 12 - Exemplos de operações utilizadas no processo de *data augmentation*



Fonte: Adaptado de Ahmad e Muhammad e Baik (2017, p. 6)

A ampliação de dados aplicada a caracteres latinos manuscritos exige atenção e cuidado, pois, ao aplicar alguma das transformações comumente utilizadas no processo, certos caracteres podem-se se transformar em outros, gerando assim confusão no treinamento. Por exemplo, as letras “u” e “c” podem ser confundidas ao aplicar rotação, assim como a letra minúscula “b” pode se transformar em “d” ao aplicar o espelhamento horizontal, em “p” ao aplicar o espelhamento vertical e em “q” ao aplicar espelhamento vertical seguido de um espelhamento horizontal, com pode-se observar na Figura 13.

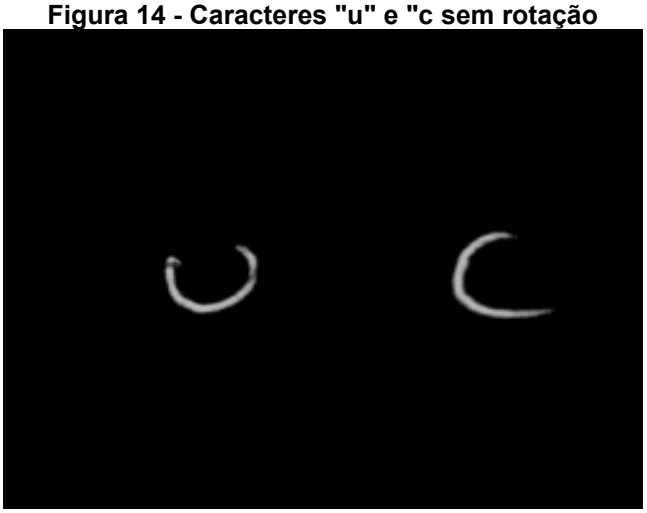
Figura 13 - Exemplo de como uma letra pode ser transformada em outras a partir de operações

Fonte: Adaptado de Viard-Gaudin *et al.* (1999)

Pode-se observar nas Figura 14 e Figura 15 que os caracteres “u” e “c” aos serem rotacionados podem ser confundidos. Na primeira composição, os caracteres “u” e “c” estão posicionados lado a lado, sem nenhuma operação aplicada, além da inversão de cores para facilitar a visualização. Já na segunda composição, os caracteres trocaram de ordem e aplicou-se rotação em ambos: o caractere “c” foi rotacionado em -90 graus e o caractere “u” foi rotacionado em 90 graus.

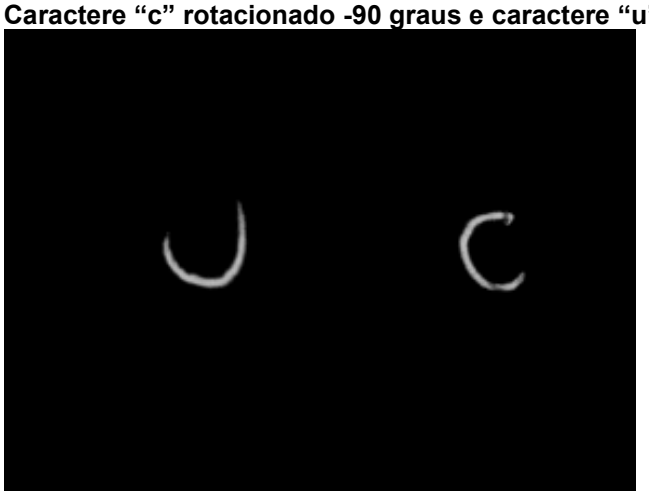
Como as imagens exibem, o resultado final das composições foram praticamente o mesmo. Com isso, utilizando os caracteres como entrada na rede neural, o algoritmo de treinamento não será capaz de distinguir a o caractere “u” do caractere “c”, diminuindo a acurácia do modelo construído.

Figura 14 - Caracteres "u" e "c sem rotação



Fonte: Adaptado de Viard-Gaudin *et al.* (1999))

Figura 15 - Caractere “c” rotacionado -90 graus e caractere “u” 90 graus



Fonte: Adaptado de Viard-Gaudin *et al.* (1999)

Na pesquisa de Rahman et al. (2015), para a realização de data augmentation são aplicadas as operações de *shifts* verticais e horizontais, rotação e zoom de 10 graus. Baseado nessa pesquisa, neste trabalho cada imagem no processo de ampliação de dados foi gerada com translação, e rotações de, no máximo, 15 graus e *shifts* verticais e horizontais e aproximações de, no máximo, 15%. Esse número se mostrou adequados para as amostras contidas no *dataset* IRONOFF.

Para construir o *dataset* com *data augmentation* deste trabalho, foram geradas 13 novas versões de cada sub *dataset* presente na base de dados original, aplicando as transformações citadas.

4.3 *Bounding box*

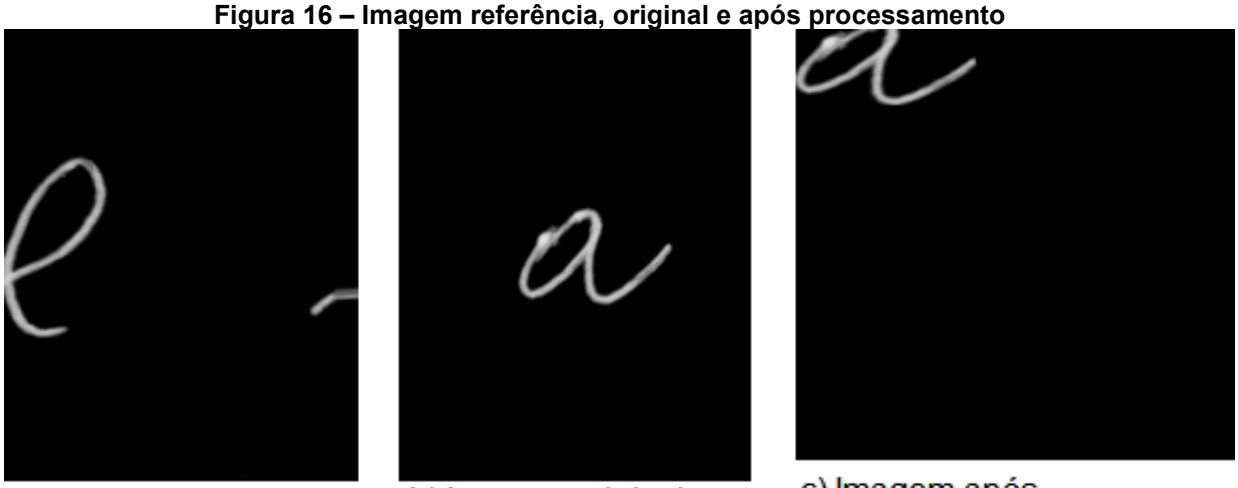
Após a aplicação do data augmentation, foi aplicada a técnica de *bounding box* nas imagens. Essa técnica consiste em encontrar o retângulo de menor área que agrupe todos os pontos de uma determinada imagem.

Ambas as redes neurais deste trabalho, VGG adaptada e LeNet-5, recebem como entrada imagens quadradas. Por conta disso, foi necessário aplicar um pré-processamento nas imagens para transformar entradas retangulares em entradas quadradas. Para que as redes neurais recebessem as imagens padronizadas, foi encontrado o retângulo que agrupa os pontos de todas as imagens da base de dados geradas *após o data augmentation*.

Após encontrada a área de *bounding box*, e recortada a imagem, a área final é expandida, tendo como medida de X e Y o maior valor da área de *bounding box*, para se obter imagens quadradas ao fim do processo.

Por conta dessas características, esse processo acabou aumentando a dimensão da imagem, transformando a imagem original 166x214 em imagens 208x208, pois, como é exibido na Figura 16, há imagens que contém informações em diferentes extremidades da área da imagem.

Figura 16 – Imagem referência, original e após processamento



Fonte: Adaptado de Viard-Gaudin *et al.* (1999)

Esse pré-processamento é ilustrado na Figura 16, na qual a primeira imagem exibe a imagem de referência, aquela que teve o maior valor encontrado, a segunda

imagem exibe a imagem original e a terceira imagem exibe a mesma entrada após a aplicação do *bounding box* e expansão.

4.4 Subconjuntos

Após realizar *data augmentation*, foi realizada uma fase adicional no pré-processamento com o objetivo de transformar as imagens retangulares (167x214) da base de dados em imagens quadradas (208x208) para a entrada das redes. Para isso, identificou-se a maior região com dados (ou seja, com pixel não branco) em todas as imagens e então cada exemplar do dataset foi transformado, seguindo esse valor encontrado.

Finalizado o processo de padronização das imagens, os *datasets*, original e com *data augmentation*, foram divididos em alguns conjuntos: apenas com imagens minúsculas, apenas com imagens maiúsculas e com ambos os tipos. A Tabela 3 mostra os datasets gerados, a quantidade de imagens e qual conjunto cada um representa.

Tabela 3 - Quantidade de imagens por base de dados

Base	Quantidade de imagens	Conjunto
ORIGINAL	21364	Minúsculas e maiúsculas
ORIGINAL_MIN	10685	minúsculas
ORIGINAL_MAI	10679	maiúsculas
AUG	299096	Minúsculas e maiúsculas
AUG_MIN	149590	minúsculas
AUG_MAI	149506	Maiúsculas

Fonte: Autoria Própria (2021)

Para facilitar a compreensão, a partir desta tabela os *datasets* serão representados da seguinte maneira: cada *dataset* que contém “ORIGINAL” representa um subconjunto proveniente da base original; “AUG” representa um subconjunto gerado a partir da base com *data augmentation*; “MIN” e “MAI” representam conjuntos que contêm apenas caracteres minúsculos e caracteres maiúsculos, respectivamente. Para exemplificar, o *dataset* AUG_MIN contém apenas letras minúsculas do *dataset* com *data augmentation*.

4.5 Experimentos

Para realizar o treinamento, validação e teste das redes, cada base foi dividida seguindo a separação 60-20-20: 60% das imagens do *dataset* são usadas para treinamento, 20% para validação e 20% para testes.

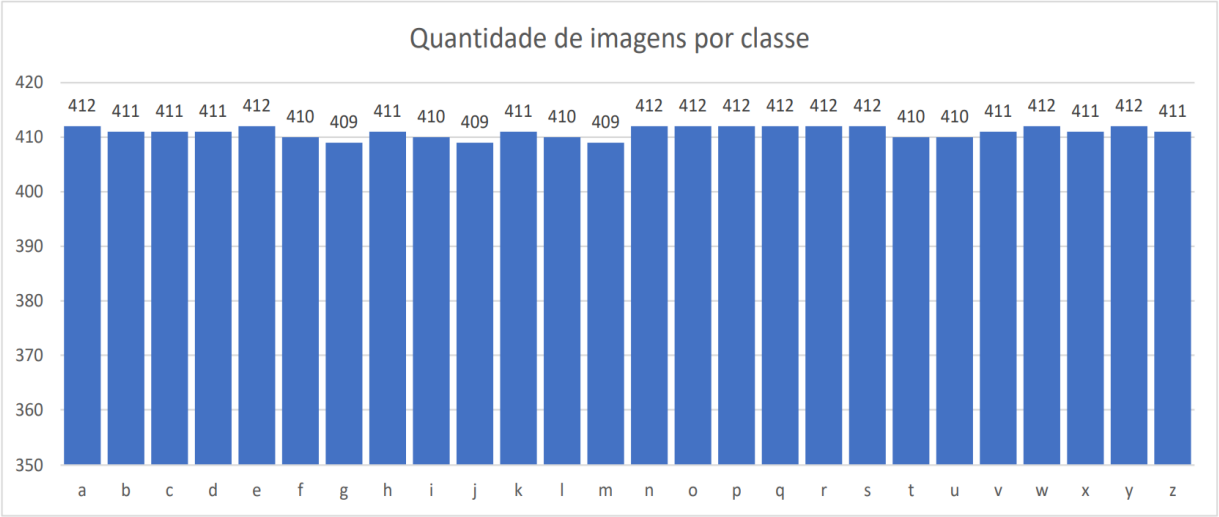
Para garantir que cada caractere tenha, proporcionalmente, a mesma quantidade de imagens para cada um dos treinamentos e testes, antes de cada divisão há um agrupamento por classe. Dessa forma, a proporção 60-20-20 é respeitada para cada classe separadamente.

Cada treinamento foi interrompido após 15 épocas sem melhoria na métrica de acurácia. Como pode-se observar nas figuras da seção 5, o processo de treinamento se estabiliza em dado momento e não há melhoria no valor da acurácia. Portanto, a fim de economizar processamento computacional e, consequentemente, diminuir o tempo de experimento, o treinamento é interrompido. São aguardadas 15 épocas pois, como também é possível observar nas figuras da seção 5, existem quedas na acurácia que precedem um aumento na mesma métrica. Por isso, para finalizar o treinamento é necessário aguardar a estabilização da curva.

Nos Gráfico 1 e Gráfico 2 pode-se observar a quantidade de imagens por classe nos subconjuntos de letras minúsculas e letras maiúsculas, respectivamente, para o *dataset* original. Na base de dados com *data augmentation*, o gráfico mantém a mesma forma, apenas com valores multiplicados por 14, como definido no processo de ampliação.

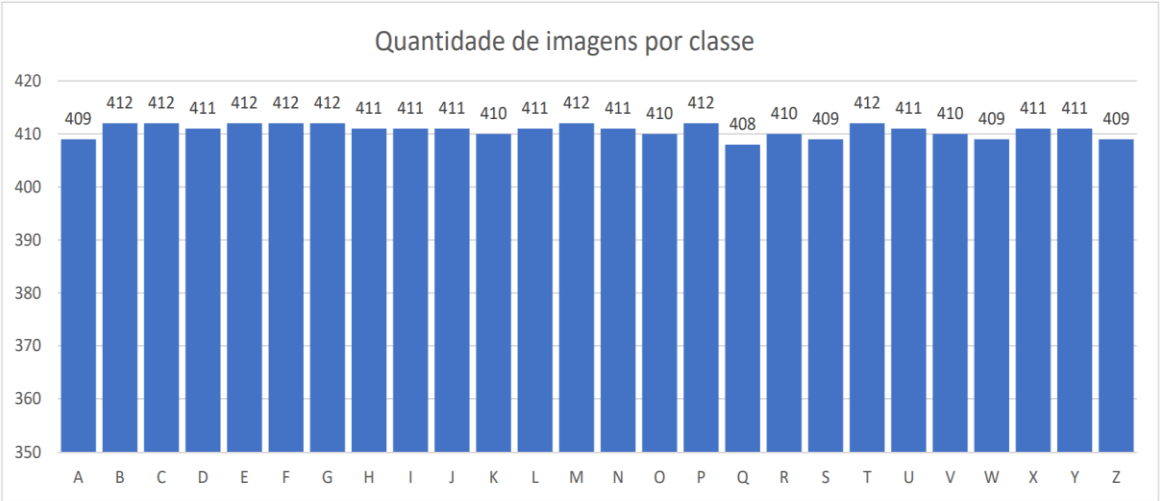
Pode-se perceber que, apesar de diferentes valores, a diferença de imagens entre cada classe é baixa. As classes com menos imagens têm 408 entradas, contra 412 entradas para as classes mais representadas. Portanto, pode-se dizer que essa é uma base equilibrada e, por isso, não foi aplicado nenhum pré-processamento para normalização.

Gráfico 1 - Quantidade de imagens por letra no subconjunto de letras minúsculas



Fonte: Autoria Própria (2021)

Gráfico 2 - Quantidade de imagens por letra no subconjunto de letras maiúsculas



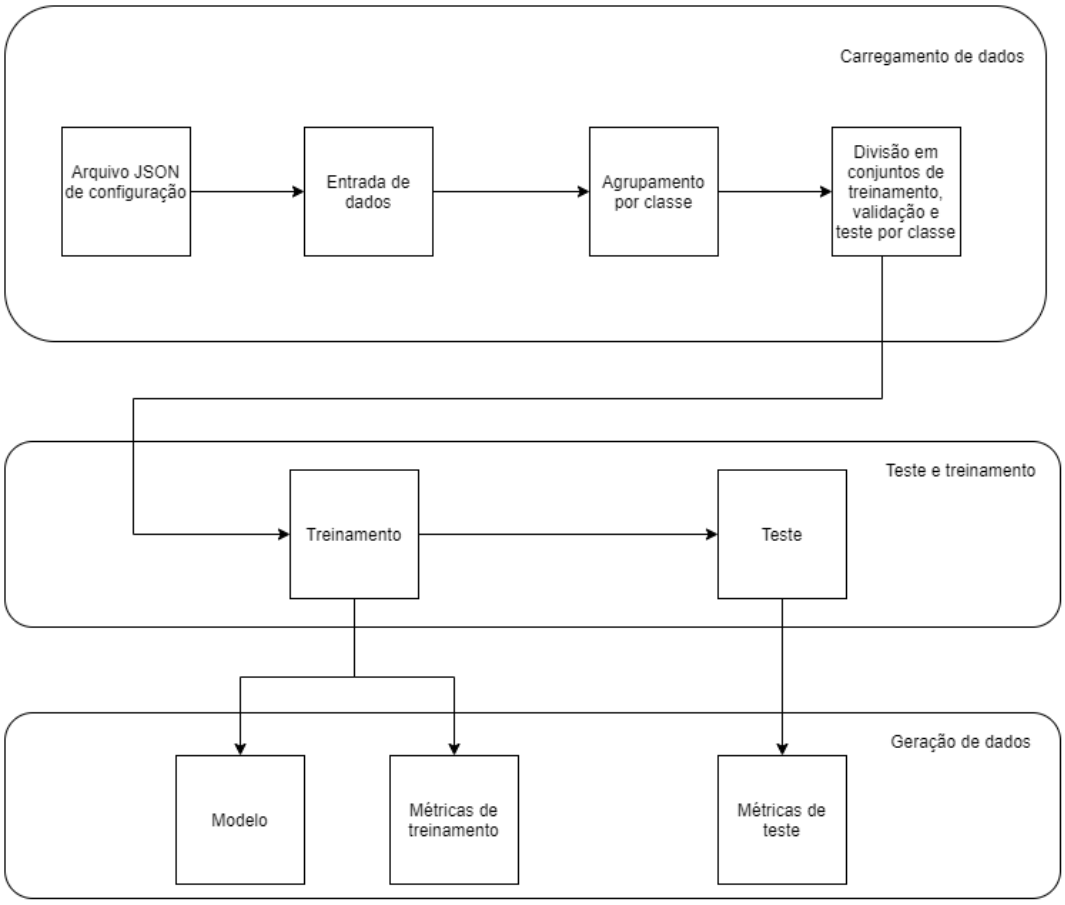
Fonte: Autoria Própria (2021)

Após a entrada de dados, ocorre o agrupamento por classe e então a divisão dos dados entre treinamento, validação e teste para envio para as redes. Como saída, o processo de treinamento e teste gera três artefatos:

1. arquivo contendo valores de acurácia e perda, para treinamento e teste, após cada época;
2. arquivo contendo dados da matriz de confusão;
3. arquivo com o modelo gerado pelo treinamento

Um fluxograma simplificado de todo processo pode ser visualizado na Figura 17, divido em três fases: entrada de dados, treinamento e teste e geração de dados.

Figura 17 - Fluxograma simplificado do processo de treinamento e teste



Fonte: Autoria Própria (2021)

4.6 Redes Neurais Convolucionais

Para este trabalho, foram utilizadas, nos experimentos, duas redes neurais convolucionais: LeNet-5 e uma VGG adaptada.

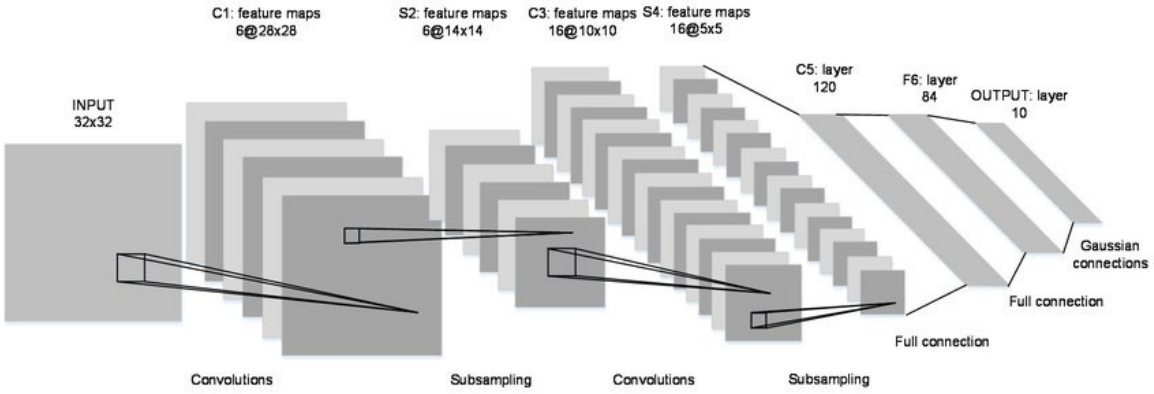
4.6.1 LeNet-5

A LeNet-5 foi criada por LeCun *et al.* (1989) e é conhecida como uma das primeiras redes neurais convolucionais criadas. É considerada uma rede simples, por

conter apenas unidades de processamento básicas de uma CNN: camadas de *pooling*, camadas totalmente conectadas e camadas convolucionais.

Na Figura 18, é possível visualizar a arquitetura de uma rede LeNet-5, que, segundo Tra *et al.* (2017), é uma arquitetura típica de uma rede neural convolucional. Neste trabalho, esta rede foi implementada utilizando as tecnologias citadas na seção 4.7.

Figura 18 - Arquitetura LeNet5



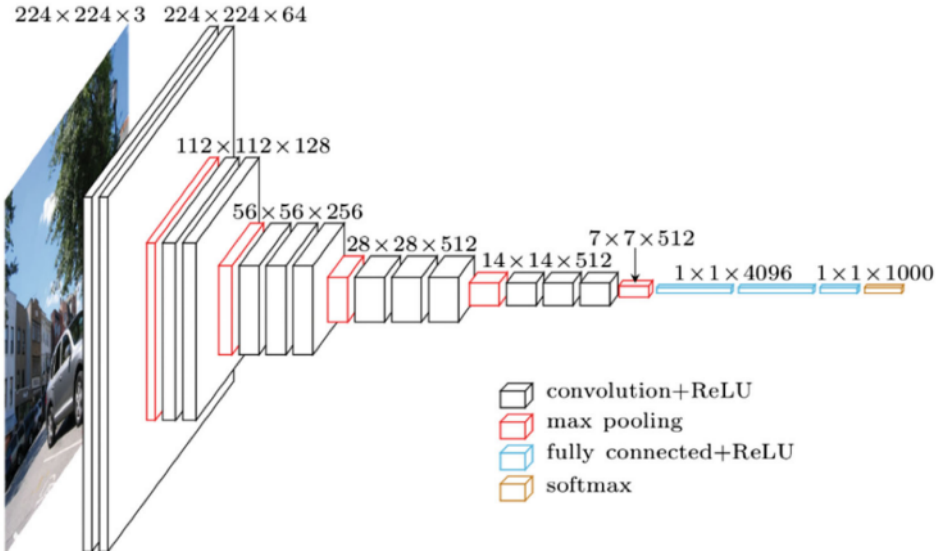
Fonte: (TRA *et al.*, 2017)

4.6.2 VGG adaptada

A rede VGG adaptada tem como ponto de partida a rede VGG-16, uma rede convolucional simples que contém apenas filtros convolucionais 3x3 (SIMONYAN; ZISSERMAN, 2014). Na Figura 19 é possível visualizar a arquitetura de uma rede VGG-16.

A rede VGG adaptada tem como inspiração as primeiras camadas convolucionais da VGG-16, que contém 64, 128, 256 e 512 filtros, aumentando a quantidade de filtros a cada camada. Além disso, há camadas de *max pooling* após cada camada convolucional.

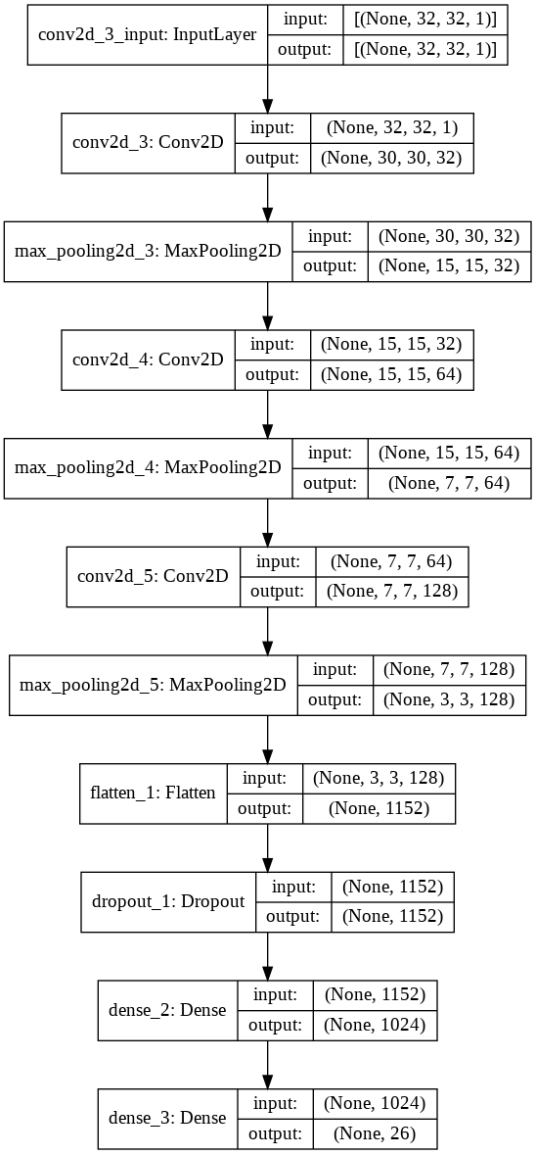
Figura 19 - Arquitetura da rede VGG16



Fonte: (NASH; DRUMMOND; BIRBILIS, 2018)

A arquitetura da rede VGG adaptada tem como *input* uma imagem de 32x32x1 e existem camadas convolucionais, de *pooling* e totalmente conectadas. A camada C1 contém 32 filtros, enquanto C3 contém 64 filtros e C5, 128 filtros. As camadas C2, C4 e C6 são camadas de *Max Pooling* 2x2. Após as camadas convolucionais e de *pooling*, há um *Dropout* de 50%. A seguir existem duas camadas totalmente conectadas: a primeira com 1024 unidades e a última com a quantidade de classes do problema (26 classes ou 52 classes, quando se utiliza as bases minúsculas e maiúscula misturadas). A arquitetura descrita pode ser observada na ilustração a seguir e foi construída através de testes e experimentações realizadas durante o período de desenvolvimento deste trabalho.

Figura 20 - Arquitetura da rede VGG adaptada



Fonte: Autoria própria (2021)

4.7 Frameworks e tecnologias

Para o desenvolvimento deste trabalho, foi utilizada a linguagem Python que, de acordo com Silva et al. (2019), é uma linguagem de programação de alto nível, fácil aprendizagem, robusta e utilizada em diversas aplicações. Entre elas, o aprendizado de máquina.

Para a implementação das redes neurais, utilizou-se a biblioteca TensorFlow que, segundo Ertman *et al.* (2017), é uma biblioteca de código aberto, desenvolvida pela Google, para computação numérica. Além disso, o TensorFlow permite que um

software desenvolvido nesta biblioteca possa ser portado com pouca ou nenhuma alteração para uma grande variedade de aplicações, desde celulares até grandes sistemas distribuídos com GPUs (Graphics Processing Units).

Outras bibliotecas foram utilizadas em caráter de suporte a tarefas de pré-processamento e pós-treinamento, como MatplotLib[1], para geração de gráficos e Numpy[2] e Pandas[3] para manipulação de matrizes e cálculos.

[1] Disponível em: https://matplotlib.org/
[2] Disponível em: https://numpy.org/
[3] Disponível em: https://pandas.pydata.org/

5 RESULTADOS E DISCUSSÕES

Este capítulo apresenta os resultados obtidos no treinamento, validação e testes das redes LeNet-5 e da rede VGG adaptada para os seis *datasets* desta pesquisa.

Após a apresentação dos resultados, serão discutidos os principais pontos de confusão no treinamento, utilizando imagens e matrizes de confusão.

5.1 Resultados das redes neurais

Os principais resultados obtidos nos experimentos estão compilados nas tabelas desta seção, que apresentam os valores de acurácia e perda no treinamento e validação, além dos valores de acurácia nos testes.

Na Tabela 4 são exibidos os valores de acurácia obtidos durante a etapa de teste. Na Tabela 5 são exibidos os valores de perda no treinamento e validação. Na Tabela 6 são exibidos os valores de acurácia no treinamento e validação, para a VGG adaptada e LeNet-5. Além disso, são apresentados e discutidos os gráficos de acurácia e perda no treinamento e validação, para todos os *datasets* e redes neurais e discutidos os dados presentes nesses gráficos.

Tabela 4 - Valor de acurácia no teste para cada sub *dataset*

Base	VGG adaptada	LeNet-5
ORIGINAL	58,08%	45,28%
ORIGINAL_MIN	65,58%	46,13%
ORIGINAL_MAI	75,45%	54,58%
AUG	79,07%	66,16%
AUG_MIN	86,73%	74,01%
AUG_MAI	92,61%	83,05%

Fonte: Autoria Própria (2021)

Tabela 5 - Valor de perda no treinamento e validação para cada sub *dataset*

Base	Rede	Treinamento	Validação
ORIGINAL	VGG adaptada	0,6998	1,4433
ORIGINAL	LeNet-5	1,4830	1,9628
ORIGINAL_MIN	VGG adaptada	0,5363	1,2762
ORIGINAL_MIN	LeNet-5	1,3688	1,7913
ORIGINAL_MAI	VGG adaptada	0,3822	0,9619
ORIGINAL_MAI	LeNet-5	0,9774	1,5670
AUG	VGG adaptada	0,4242	0,5813
AUG	LeNet-5	0,9245	1,0756
AUG_MIN	VGG adaptada	0,1961	0,4265
AUG_MIN	LeNet-5	0,6795	0,8523
AUG_MAI	VGG adaptada	0,0918	0,2528
AUG_MAI	LeNet-5	0,3505	0,5549

Fonte: Autoria Própria (2021)

Tabela 6 - Valor de acurácia no treinamento e validação para cada sub *dataset*

Base	Rede	Treinamento	Validação
ORIGINAL	VGG adaptada	76,29%	58,88%
ORIGINAL	LeNet-5	55,34%	45,98%
ORIGINAL_MIN	VGG adaptada	82,13%	64,30%
ORIGINAL_MIN	LeNet-5	56,94%	47,27%
ORIGINAL_MAI	VGG adaptada	87,33%	73,87%
ORIGINAL_MAI	LeNet-5	68,78%	57,22%
AUG	VGG adaptada	84,01%	79,48%
AUG	LeNet-5	69,68%	66,18%
AUG_MIN	VGG adaptada	93,12%	87,02%
AUG_MIN	LeNet-5	78,25%	73,86%
AUG_MAI	VGG adaptada	96,73%	92,77%
AUG_MAI	LeNet-5	88,66%	83,69%

Fonte: Autoria Própria (2021)

Em todos os experimentos, a rede VGG adaptada apresentou valores superiores aos obtidos na rede LeNet-5. Como pode-se observar na Tabela 4, o melhor resultado para o *dataset* sem ampliação de dados foi obtido na rede VGG adaptada, para o subconjunto ORIGINAL_MAI, com acurácia de 75,45%. O melhor

resultado para a base de dados com *data augmentation* foi obtido também na rede VGG adaptada, para o subconjunto AUG_MAI, com 92,61%.

Em relação à perda, o melhor resultado para o *dataset* sem *data augmentation* também foi obtido para o subconjunto ORIGINAL_MAI, com 0,3822 no treinamento e 0,9619 na validação. Para o *dataset* com *data augmentation*, o mesmo ocorre, com 0,0918 no treinamento e 0,2528 na validação para o subconjunto AUG_MAI.

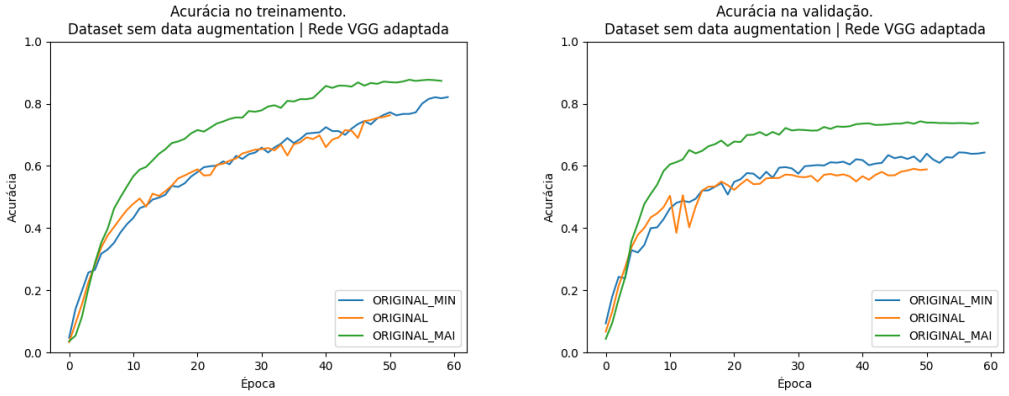
5.1.1 VGG adaptada

A rede VGG adaptada apresentou melhores resultados que a rede LeNet-5, em todos as comparações, para o *dataset* IRONOFF, com acurácia maior e perda menor.

A melhor acurácia de treinamento encontrada ocorre no sub *dataset* AUG_MAI, caracteres maiúsculos com *data augmentation*, de 92,61%. Comparando o *dataset* ORIGINAL_MAI apresentou acurácia de 75,45%, o que mostra a melhoria causada pelo processo de ampliação de dados (*data augmentation*). O mesmo ocorre para demais subconjuntos. Em relação à perda, o melhor valor (menor perda) encontrado durante a validação também foi no subconjunto AUG_MAI, com valor de aproximadamente 0,2528.

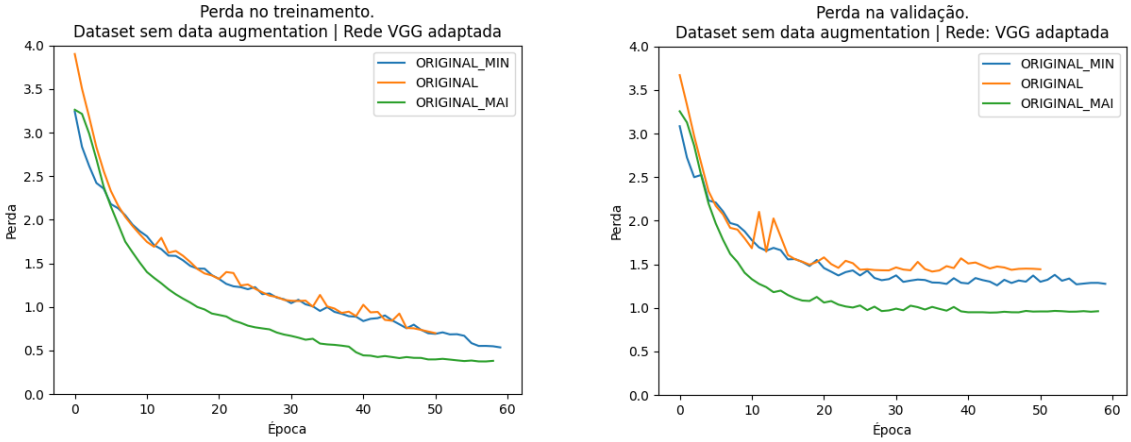
No Gráfico 3, são apresentadas as acurácias em treinamento e validação, para os sub *datasets* da base original. No Gráfico 4, estão os valores de perda para o mesmo conjunto e rede.

Gráfico 3 - Valores de acurácia no treinamento e validação para a base original e rede VGG adaptada



Fonte: Autoria própria (2021)

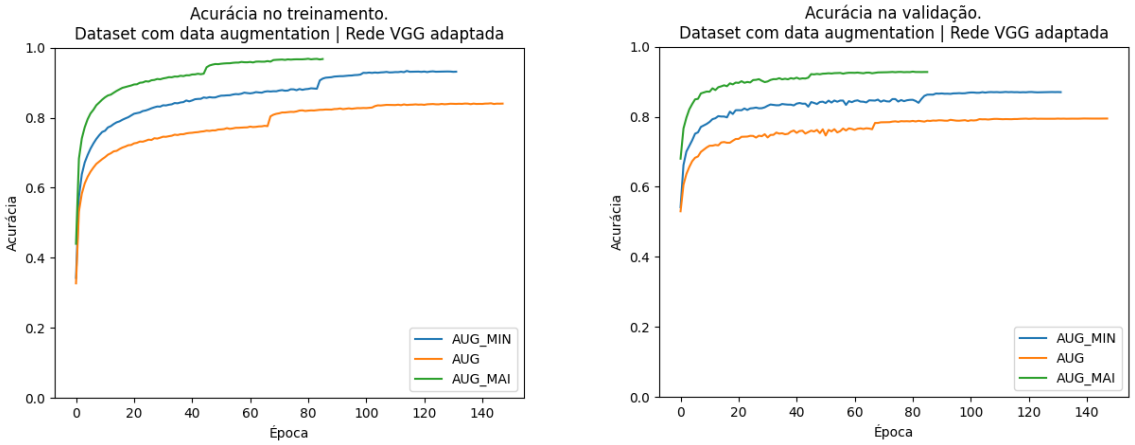
Gráfico 4 - Valores de perda no treinamento e validação para a base original e rede VGG adaptada



Fonte: Autoria própria (2021)

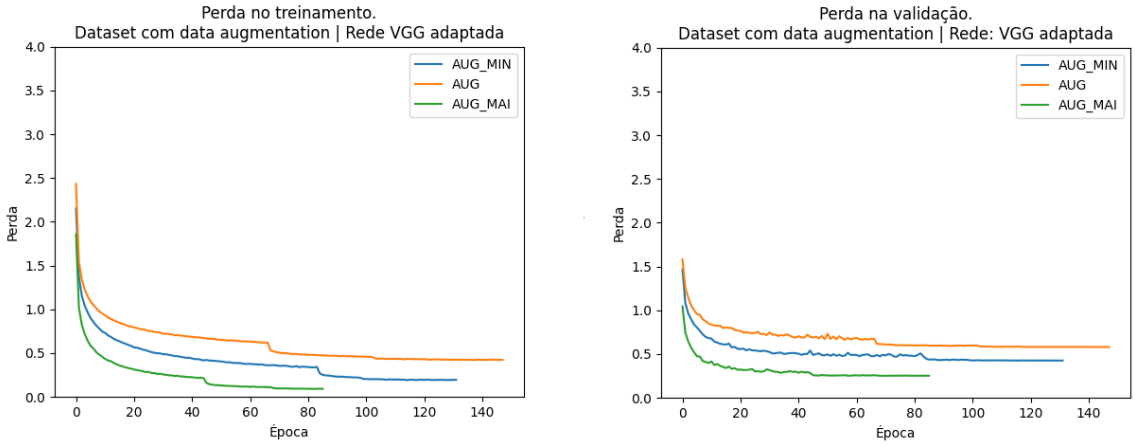
No Gráfico 5, são exibidas as acurácias em treinamento e validação, para os conjuntos com *data augmentation*. No Gráfico 6 são exibidos os valores de perda.

Gráfico 5 - Valores de acurácia no treinamento e validação para a base com *data augmentation* e rede VGG adaptada



Fonte: Autoria própria (2021)

Gráfico 6 - Valores de perda no treinamento e validação para a base com *data augmentation* e rede VGG adaptada



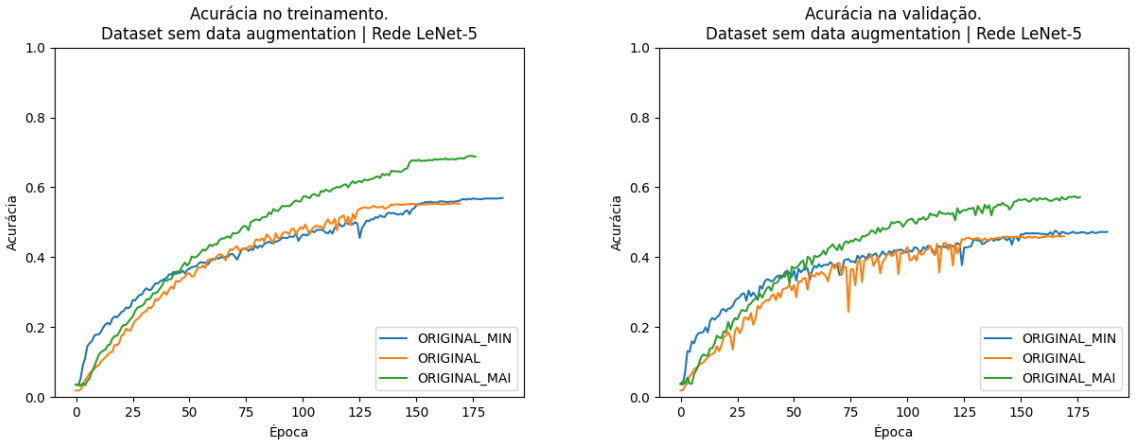
Fonte: Autoria própria (2021)

5.1.2 LeNet-5

A LeNet-5, apesar de apresentar menores valores de acurácia, apresentou os mesmos comportamentos da rede VGG adaptada: a melhor acurácia de treinamento encontrada ocorre no sub *dataset* AUG_MAI, letras maiúsculas com *data augmentation*, de 83,05%. Também há melhoria em relação ao subconjunto ORIGINAL_MAI que apresenta 54,58% de acurácia. Em relação à perda, o melhor valor encontrado também foi no subconjunto AUG_MAI, com valor de aproximadamente 0,5549.

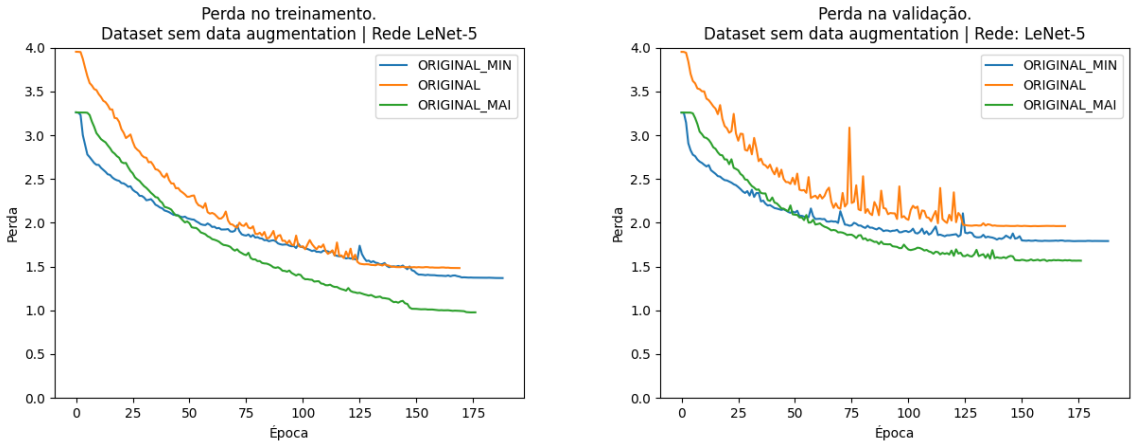
São exibidas as acurácias de treinamento e validação obtidas pela rede LeNet-5 no Gráfico 7 e os valores de perda para o mesmo conjunto e rede são exibidos no Gráfico 8.

Gráfico 7 - Valores de acurácia no treinamento e validação para a base original e rede LeNet-5



Fonte: Autoria própria (2021)

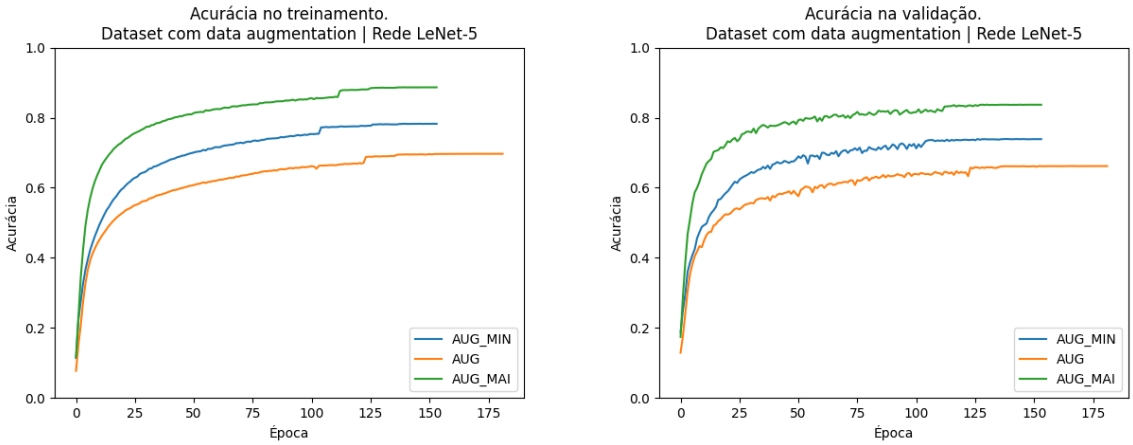
Gráfico 8 - Valores de perda no treinamento e validação para a base original e rede LeNet-5



Fonte: Autoria própria (2021)

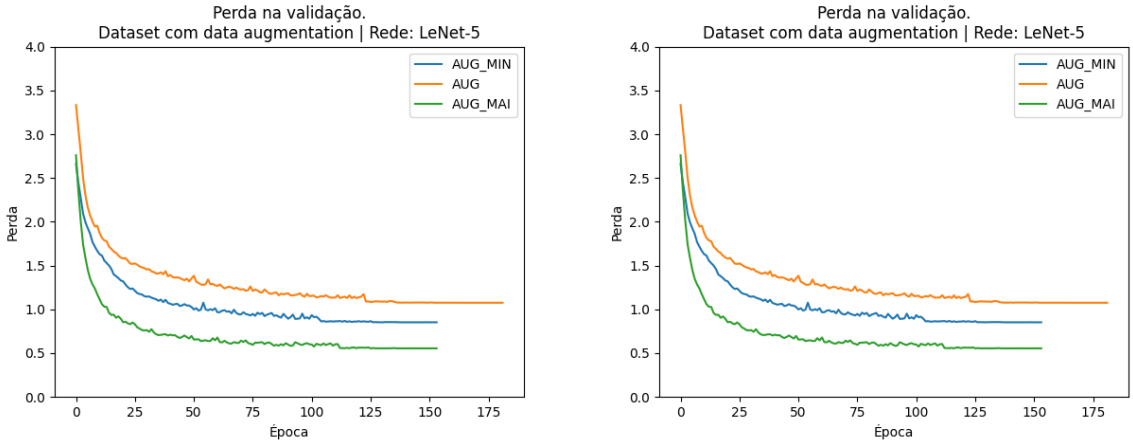
No Gráfico 9 são exibidas as acurácias em treinamento e validação, para os sub *datasets* da base com *data augmentation*. São exibidos os valores de perda no Gráfico 10.

Gráfico 9 - Valores de acurácia no treinamento e validação para a base com *data augmentation* e rede LeNet-5



Fonte: Autoria própria (2021)

Gráfico 10 - Valores de perda no treinamento e validação para a base com *data augmentation* e rede LeNet-5



Fonte: Autoria própria (2021)

Como pode-se observar nos gráficos desta seção, os conjuntos ORIGINAL e ORIGINAL_MIN têm valores similares durante todo o processo de treino e validação, se separando minimamente apenas nas últimas épocas. O conjunto ORIGINAL_MAI apresenta valores superiores de acurácia, e menores de perda, durante todo o treinamento e validação, após algumas épocas iniciais (cerca de 5 épocas na VGG adaptada e 50 na LeNet-5).

Ao analisar os conjuntos com *data augmentation*, percebe-se que o conjunto de maiúsculas, AUG_MAI, também apresenta valores superiores durante a maior parte do teste e treinamento, após algumas épocas iniciais. Entretanto, diferentemente

do que é observado nos conjuntos sem *data augmentation*, os conjuntos AUG_MIN e AUG têm valores diferentes, com maior separação, durante o treino e validação. Isso ocorre, pois, o *data augmentation* fornece à rede melhor capacidade de generalização, dessa forma fazendo com que a rede entenda melhor as classes das letras minúsculas, aumentando assim sua acurácia e diminuindo o valor de perda.

5.2 Discussões

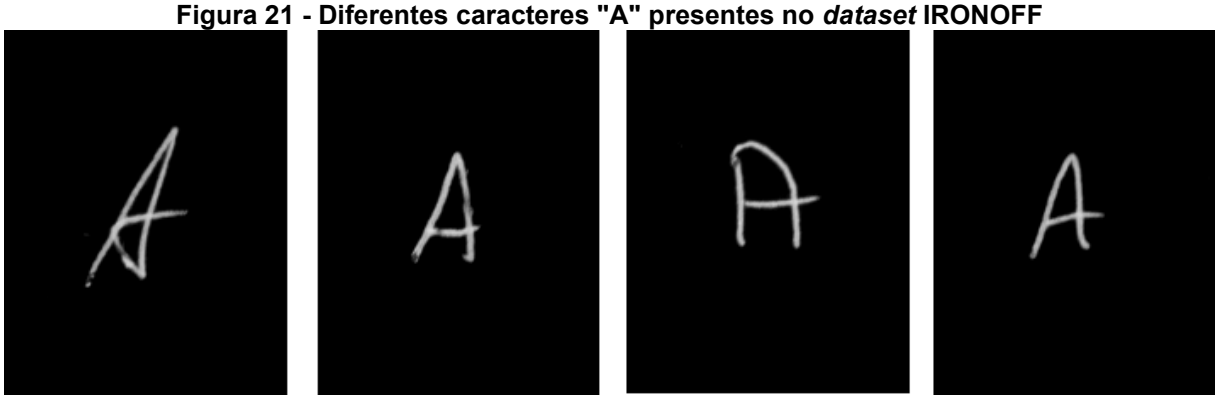
Nesta seção serão discutidos e exemplificados resultados das redes neurais e *datasets*. Ainda, serão abordados pontos de confusão que ocorreram na etapa de testes das redes neurais convolucionais. Para facilitar a compreensão, na discussão dos pontos de confusão serão apresentados dados e gráficos apenas da rede VGG adaptada e com o *dataset* com *data augmentation*. Considera-se relevante essa análise, uma vez, como esse grupo foi o que apresentou melhores resultados, será possível destacar desafios para trabalhos futuros.

5.2.1 *Datasets* com melhores resultados

Como pode-se observar nas Figuras da seção 5.1, os melhores resultados, para os conjuntos com e sem *data augmentation*, foram obtidos nos *datasets* MAI, ou seja, nos conjuntos que contém apenas caracteres maiúsculos. Isso se deve à menor variabilidade de escrita dos caracteres maiúsculos em relação aos caracteres minúsculos. O caractere maiúsculo “A”, por exemplo, tem baixa variabilidade, como ilustrado na Figura 21. Enquanto o mesmo caractere, em sua versão minúsculo, “a”, é escrito de diferentes maneiras, como é exibido na Figura 22.

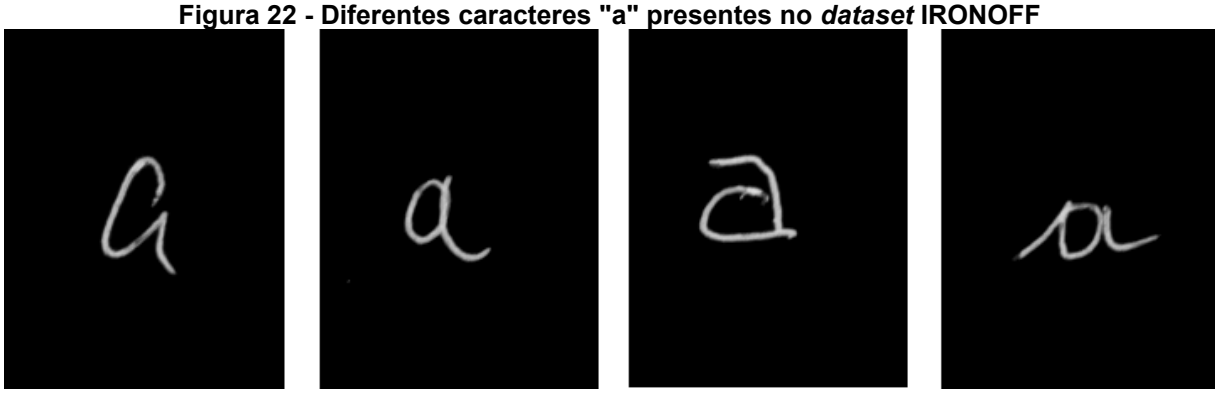
Como pode-se observar nas Figura 21 e Figura 22, o caractere “a” minúsculo apresenta maior variabilidade comparando com o caractere “A” maiúsculo. Além disso, o caractere “a” minúsculo apresenta maior semelhança com outras caracteres do conjunto das letras minúsculas, como “q”, “d” e “o”.

Figura 21 - Diferentes caracteres "A" presentes no *dataset* IRONOFF



Fonte: Adaptado de Viard-Gaudin *et al.* (1999)

Figura 22 - Diferentes caracteres "a" presentes no *dataset* IRONOFF



Fonte: Adaptado de Viard-Gaudin *et al.* (1999)

As amostras foram selecionadas dos mesmos autores. Ou seja, o primeiro caractere da Figura 22 foi escrita pela mesma pessoa do primeiro caractere da Figura 23 e assim sucessivamente.

Como observa-se na Figura 22, o caractere “A” contém baixíssima variação de estilo de escrito. No primeiro deles, o caractere é escrito um pouco mais inclinado, enquanto no terceiro os traços são mais arredondados. Entretanto, as características principais do caractere, que o identificam, permanecem os mesmos.

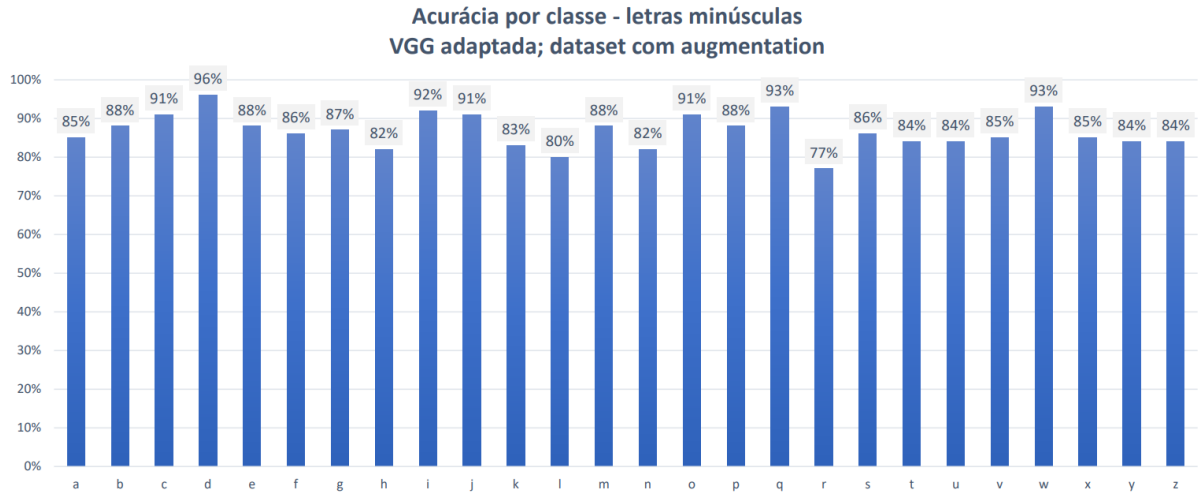
5.2.2 Principais pontos de confusão

Nesta seção, serão discutidos os principais pontos de confusão obtidos nas etapas de treinamento, validação e teste desta pesquisa. O objetivo é analisar a acurácia de cada classe em seus conjuntos e quais os pontos que fazem com que determinada acurácia seja obtida.

5.2.2.1 AUG_MIN e rede VGG adaptada

No Gráfico 11, é possível observar a acurácia de cada caractere para a rede VGG adaptada, com subconjunto AUG_MIN. O caractere com maior acurácia foi o “d”, com 96% de acurácia, enquanto o com menor acurácia foi “r”, com 77% de acurácia.

Gráfico 11 - Acurácia por classe, para subconjunto AUG_MIN e rede VGG adaptada



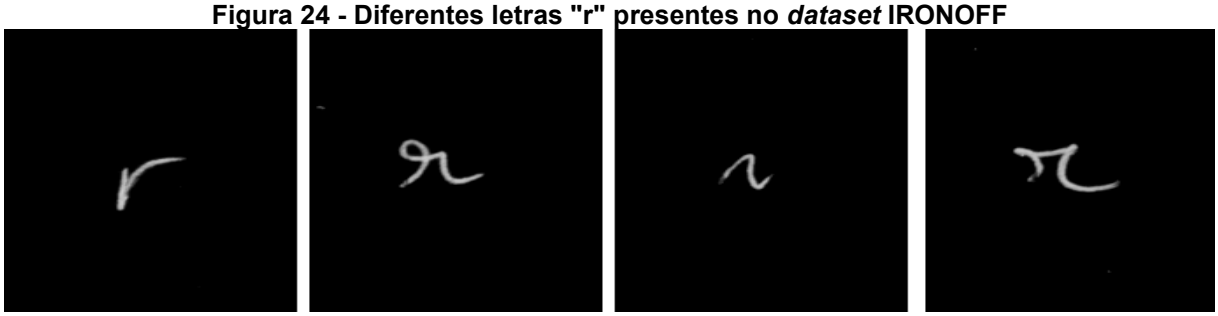
Fonte: Autoria própria (2021)

Figura 23 - Diferentes caracteres "d" presentes no *dataset* IRONOFF

Fonte: Adaptado de Viard-Gaudin *et al.* (1999)

Como pode-se observar na Figura 23, o caractere “d” tem baixa variabilidade de escrita, o que causa a alta acurácia obtida, pela boa generalização no treinamento. Entretanto, como pode-se observar na Figura 24, o caractere “r” pode ser escrito de diferentes maneiras, o que dificulta a generalização. Dessa forma, a rede neural aprende características diferentes, não conseguindo ressaltar características que podem distinguir um do outro, causando confusão entre os caracteres.

Figura 24 - Diferentes letras "r" presentes no *dataset* IRONOFF



Fonte: Adaptado de Viard-Gaudin *et al.* (1999)

Figura 25 - Matriz de confusão do subconjunto AUG_MIN e rede VGG adaptada

Classe correta \ Classe predita	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	
a	985	1	4	9	4	1	11	3	0	0	0	0	2	14	36	2	30	8	3	0	15	1	1	21	0	3	85%
b	1	1015	1	3	3	20	3	45	1	3	14	8	0	1	0	0	1	0	8	14	2	1	2	0	0	5	88%
c	0	0	1051	0	49	0	0	0	2	0	1	2	1	3	15	0	0	10	3	3	4	0	0	3	0	4	91%
d	11	2	1	1104	0	0	1	3	0	9	0	0	0	0	1	0	2	0	6	0	2	1	3	4	0	1	96%
e	18	1	47	0	1021	3	1	0	0	0	0	7	0	6	15	6	2	8	3	2	3	0	0	5	0	6	88%
f	0	12	1	2	2	993	8	5	3	20	11	32	0	0	0	24	1	11	1	13	0	0	0	1	1	7	86%
g	5	3	0	1	2	11	996	3	0	1	0	2	0	3	6	2	26	1	21	1	0	0	1	0	27	34	87%
h	1	38	0	1	2	5	3	947	0	0	92	11	2	9	0	3	2	11	8	8	1	1	2	3	1	0	82%
i	0	1	4	4	2	2	0	2	1055	16	1	37	0	0	0	6	0	1	3	10	0	0	1	0	0	3	92%
j	1	3	0	15	0	13	13	1	10	1047	2	2	0	0	1	1	1	0	10	7	0	0	0	1	10	8	91%
k	1	13	1	2	1	11	0	90	3	0	954	9	0	9	2	1	1	5	1	15	7	5	6	12	2	0	83%
l	0	4	6	0	11	42	4	13	35	4	5	916	0	0	1	62	3	1	7	26	0	2	0	1	1	4	80%
m	3	0	0	0	1	0	1	1	1	0	4	0	1007	77	1	3	2	5	0	0	15	5	14	5	1	0	88%
n	4	2	3	0	2	0	3	13	0	1	3	0	62	942	14	10	5	44	1	1	24	9	4	7	0	0	82%
o	14	0	23	2	13	0	4	0	0	0	1	0	2	10	1048	0	1	2	9	0	8	9	3	4	1	0	91%
p	0	2	0	0	12	19	4	0	1	0	1	36	5	11	2	1017	2	11	4	17	4	0	0	3	1	2	88%
q	27	0	2	0	1	1	8	1	0	0	0	1	1	3	7	5	1077	0	3	0	0	0	1	3	11	2	93%
r	10	1	13	1	8	3	2	7	8	1	0	1	5	73	5	17	3	887	26	20	10	7	0	33	2	11	77%
s	4	10	5	8	7	4	11	5	4	0	1	2	2	10	12	2	2	16	997	7	13	2	0	8	6	16	86%
t	0	25	7	4	8	4	1	6	14	6	7	38	1	0	2	15	1	17	7	966	1	2	2	10	3	1	84%
u	21	2	4	12	6	0	0	3	1	0	5	1	8	25	8	0	1	4	13	0	962	35	16	16	4	1	84%
v	5	1	1	1	2	0	1	0	0	3	2	3	8	7	18	1	1	6	11	7	43	979	29	8	9	5	85%
w	5	1	0	3	1	0	0	0	2	0	5	1	7	3	2	0	0	0	2	0	24	20	1070	8	0	0	93%
x	11	0	2	2	5	2	1	2	1	0	9	1	11	16	6	5	2	21	11	10	16	9	8	981	12	7	85%
y	0	1	0	4	0	3	54	0	0	11	0	2	0	4	2	2	16	8	4	7	5	16	0	15	974	26	84%
z	3	3	6	2	14	14	47	5	3	7	3	1	3	0	2	2	1	6	8	5	2	4	2	26	18	964	84%

Fonte: Autoria própria (2021)

A confusão entre caracteres pode ser analisada ao observar a matriz de confusão, apresentada na Figura 25. Nela, foi possível analisar que os caracteres que

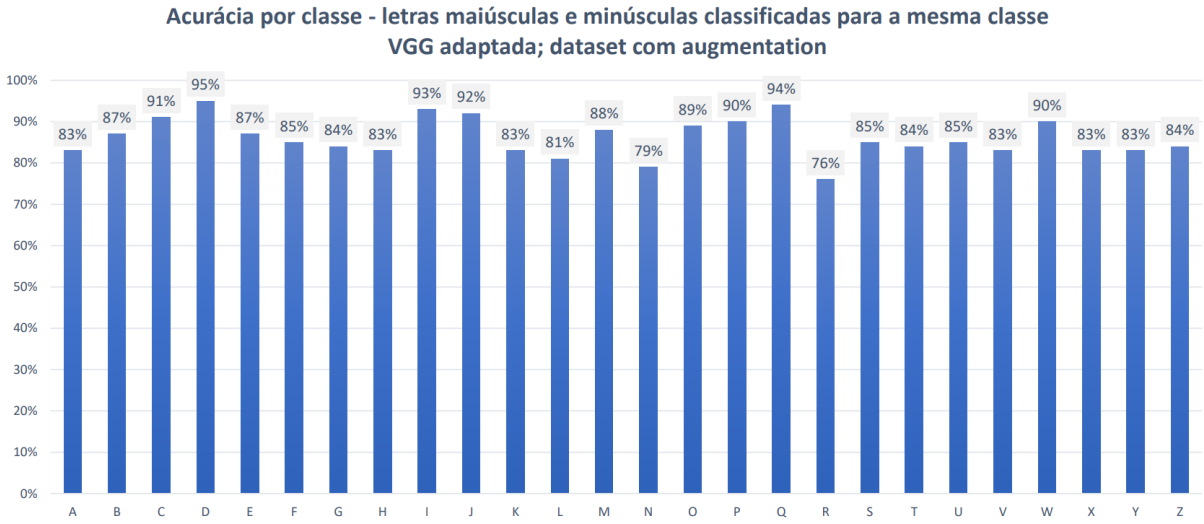
mais foram preditos como “r” de maneira errônea foram “n”, “x” e “s”, com 73, 33 e 26 classificações (quantidade de vezes que essa classe foi predita), respectivamente.

Na matriz de confusão apresentada na Figura 25, observa-se que quanto mais forte o tom de azul, maior o número de classificações ocorrências aquela letra apresentou no teste. A faixa predominante é a diagonal principal, o que mostra que para a maioria das vezes a predição foi correta.

5.2.2.2 AUG_MAI e rede VGG adaptada

No Gráfico 12, é possível observar a acurácia de cada caractere para a rede VGG adaptada, com subconjunto AUG_MAI. Este experimento obteve as melhores taxas de acurácia.

Gráfico 12 - Acurácia por classe, para subconjunto AUG_MAI e rede VGG adaptada



Fonte: Autoria própria (2021)

O caractere com menor acurácia foi o “M”, maiúsculo, com 87%, 10% a mais que o pior caractere do conjunto de teste AUG_MIN, “r”, minúsculo. Isso se deve ao fato de existir menor variabilidade no estilo de escritas de letras maiúsculas, contra uma maior variabilidade em letras minúsculas. Os caracteres que apresentaram maior acurácia foram “T”, “W” e “Z”, com 97%. Esses caracteres sofrem poucas mudanças na escrita de pessoa para pessoa e poucas semelhanças com outros caracteres.

Figura 26 - Matriz de confusão do subconjunto AUG_MAI e rede VGG adaptada

Classe correta \ Classe predita	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
A	1057	8	0	2	1	2	2	13	4	0	2	1	13	8	0	10	0	14	1	1	0	0	1	5	0	1	92%
B	9	1054	1	19	7	2	5	4	0	0	1	0	1	0	3	6	3	17	11	1	0	0	0	1	0	9	91%
C	0	1	1090	0	7	0	11	0	1	0	1	19	1	0	11	0	0	0	0	2	6	0	0	1	0	3	94%
D	4	8	0	1082	1	0	1	0	1	10	0	1	0	0	25	13	2	0	3	0	0	0	0	0	0	0	94%
E	2	3	19	1	1039	28	27	0	3	2	1	4	0	0	1	4	3	3	6	2	0	0	0	1	0	5	90%
F	1	0	1	0	10	1089	1	0	6	2	5	1	0	0	2	22	1	1	3	7	0	2	0	0	0	0	94%
G	5	7	21	0	19	5	1040	0	0	3	0	1	0	1	15	0	24	1	5	1	3	0	0	0	0	3	90%
H	15	0	1	0	0	1	0	1015	1	1	7	1	68	19	1	2	0	1	0	0	8	4	2	1	3	0	88%
I	0	0	1	2	5	1	0	0	1023	67	0	5	1	0	1	1	0	0	8	26	0	1	0	2	0	7	89%
J	0	0	0	4	1	1	2	4	35	1046	0	0	0	1	1	0	0	0	18	28	4	3	0	0	2	1	91%
K	4	0	0	0	2	4	0	11	1	0	1063	2	2	5	0	0	0	15	0	1	4	4	2	27	1	0	93%
L	0	0	29	0	4	0	0	0	21	0	1	1081	0	1	0	0	0	0	0	0	0	4	1	0	1	0	8	94%
M	15	2	0	1	1	2	0	76	0	1	8	0	1000	22	4	2	0	3	0	1	2	2	5	1	4	2	87%
N	4	1	0	3	0	0	0	17	0	2	2	1	8	1059	0	2	0	4	1	0	11	17	14	2	3	0	92%
O	0	2	23	21	5	0	17	0	0	0	0	0	2	0	1062	3	2	0	2	0	9	0	0	0	0	0	93%
P	5	0	1	16	3	48	1	0	1	1	0	0	0	0	2	1055	11	1	4	2	0	0	1	1	1	0	91%
Q	1	1	1	1	0	1	20	0	0	1	1	1	0	0	19	7	1071	10	0	0	2	1	2	0	3	0	94%
R	14	14	0	0	6	2	0	4	0	0	14	2	3	2	1	8	10	1054	1	3	1	0	0	4	0	5	92%
S	1	13	3	3	7	0	7	0	3	19	0	3	0	0	5	7	1	0	1064	2	2	1	0	1	4	0	93%
T	0	4	0	0	2	2	0	0	2	8	0	0	2	0	0	2	1	2	0	1124	0	0	0	1	3	1	97%
U	0	0	2	0	0	0	2	15	0	3	2	10	3	7	6	1	0	0	0	1	1056	36	5	1	1	0	92%
V	0	1	0	1	0	0	2	1	0	5	0	2	0	1	0	1	0	0	1	2	35	1080	2	1	13	0	94%
W	0	3	0	1	0	0	0	2	0	0	2	0	0	16	0	0	1	0	0	0	4	8	1107	1	1	0	97%
X	1	1	0	0	0	1	2	1	1	1	19	2	0	5	0	1	1	2	1	2	0	4	0	1092	10	4	95%
Y	1	1	0	0	0	0	1	0	4	4	0	3	0	2	0	2	0	0	1	3	4	23	0	9	1086	7	94%
Z	1	3	3	1	5	0	2	0	3	4	0	4	0	0	0	2	0	3	1	1	0	0	0	0	0	1113	97%

Fonte: Autoria própria (2021)

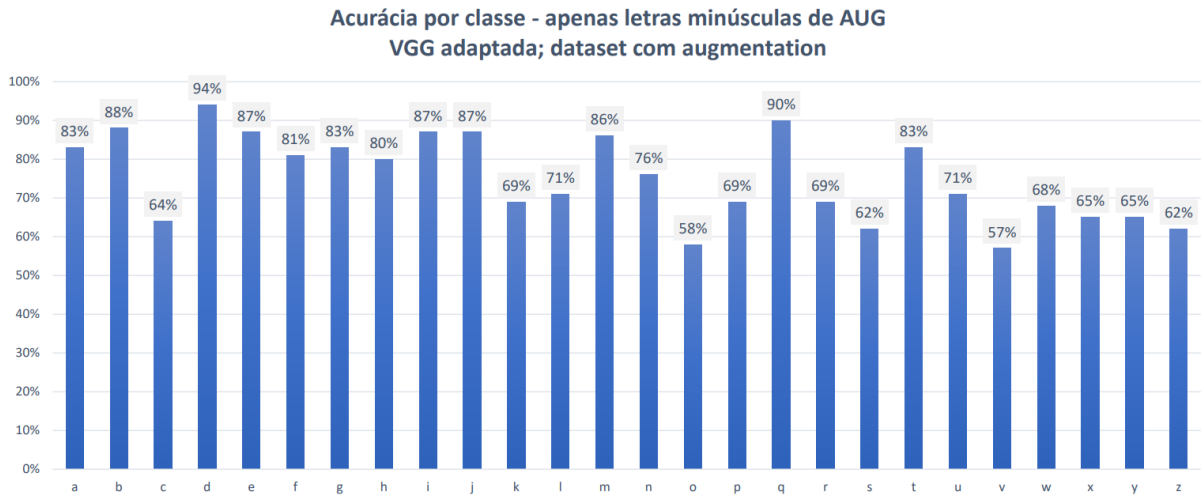
Na matriz de confusão, na Figura 29, é possível observar que a diagonal principal, que representa predições corretas, apresenta a maioria das ocorrências, com apenas 3,27% fora dela. Pode-se dizer, portanto, que a rede apresentou um bom desempenho na classificação dos caracteres do subconjunto das letras maiúsculas

5.2.2.3 AUG e rede VGG adaptada

Em razão da grande quantidade de classes presentes no subconjunto AUG (52), os gráficos serão divididos em letras maiúsculas e minúsculas. No Gráfico 13 é possível observar a acurácia de cada letra minúscula presente no subconjunto AUG, para a rede VGG adaptada. Enquanto no Gráfico 14 pode-se observar a acurácia de cada letra maiúscula.

Como é exibido no Gráfico 13 e Gráfico 14, existe uma grande diferença nos valores de acurácia entre as classes com maior acurácia e classes com menor acurácia.

Gráfico 13 - Acurácia por classe, para letras minúsculas do subconjunto AUG e rede VGG adaptada

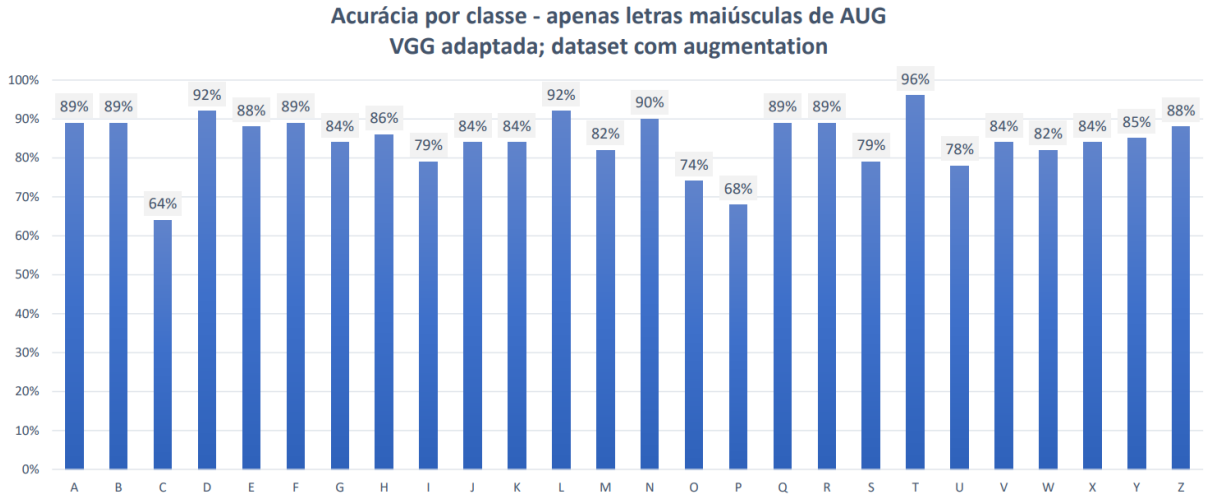


Fonte: Autoria própria (2021)

Como é exibido no Gráfico 13, para as letras minúsculas, o caractere “d” apresentou 94% de precisão, sendo a melhor desse subconjunto. Já o caractere “v” apresentou uma taxa baixa com 57% de acurácia, sendo assim o caractere com menor acurácia desse subconjunto.

Para as letras maiúsculas, como é exibido no Gráfico 14, a letra “T” é a classe com maior acurácia, apresentando 96% de precisão, sendo a melhor desse subconjunto. Já a letra “C” apresentou 64% de acurácia – a pior acurácia desse subconjunto.

Gráfico 14 - Acurácia por classe, para letras maiúsculas do subconjunto AUG e rede VGG adaptada



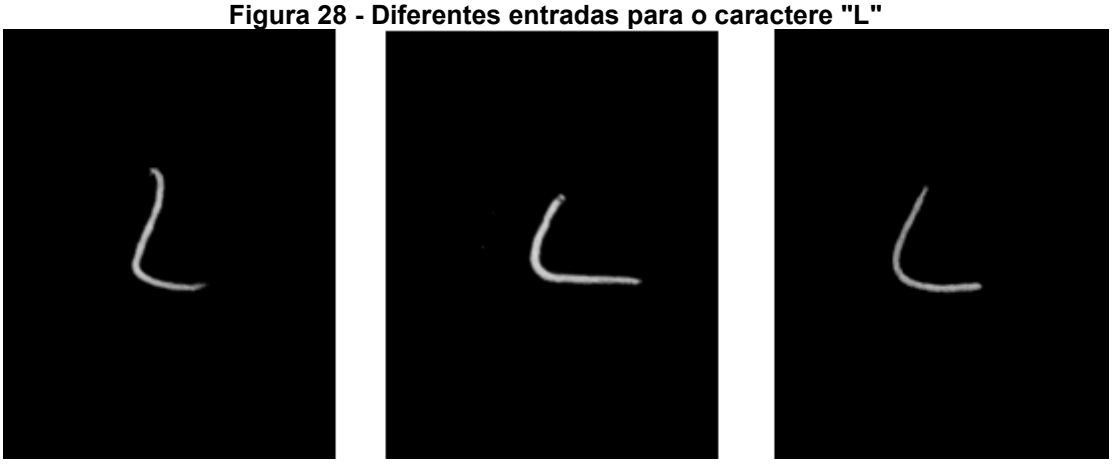
Fonte: Autoria própria (2021)

Como já discutido anteriormente, caracteres com menor variabilidade no estilo de escrita apresentam melhores resultados, como é o caso dos caracteres “T” e “L”, com taxas de acurácia 96% e 92%, respectivamente. Esses caracteres são caracteres maiúsculos que apresentam poucas diferenças na escrita de pessoa para pessoa, como pode se observar na Figura 27 e Figura 28 , respectivamente.

Figura 27 - Diferentes entradas para o caractere "T"

Fonte: Adaptado de Viard-Gaudin *et al.* (1999)

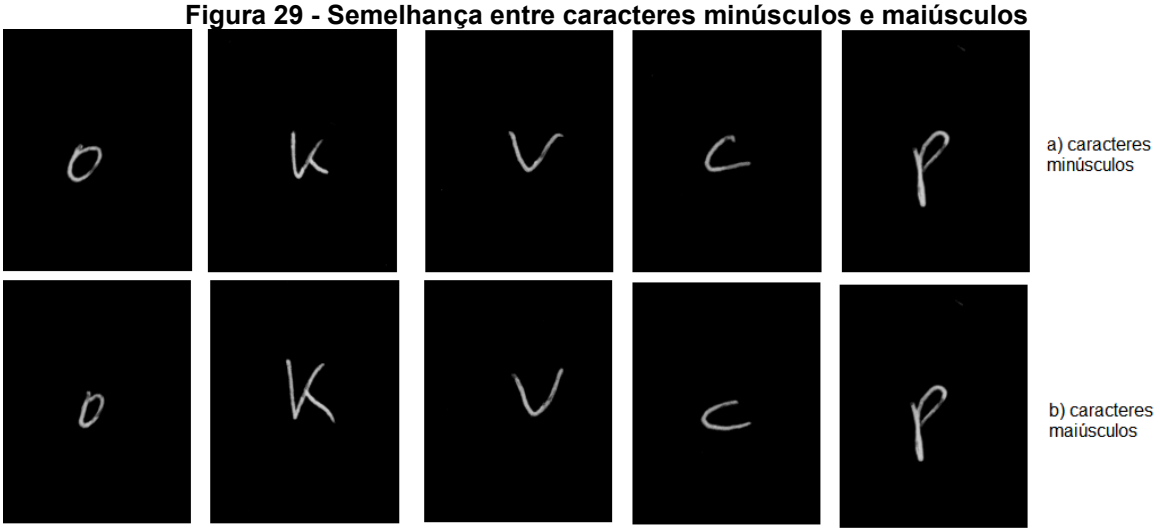
Figura 28 - Diferentes entradas para o caractere "L"



Fonte: Adaptado de Viard-Gaudin *et al.* (1999)

Entretanto, os caracteres “C”, “K”, “O”, “P”, “V”, entre outros, apresentam acurácia baixa nos testes com o *dataset* AUG mesmo como pouca variabilidade na escrita. Esse ponto é explicado pela alta semelhança desses caracteres em suas versões minúsculas e minúsculas, como é mostrado na Figura 29 Dessa forma, ao treinar a rede neural para as 52 classes, há pouca informação que possa ser usada para que a rede diferencie as classes minúsculas das maiúsculas.

Figura 29 - Semelhança entre caracteres minúsculos e maiúsculos



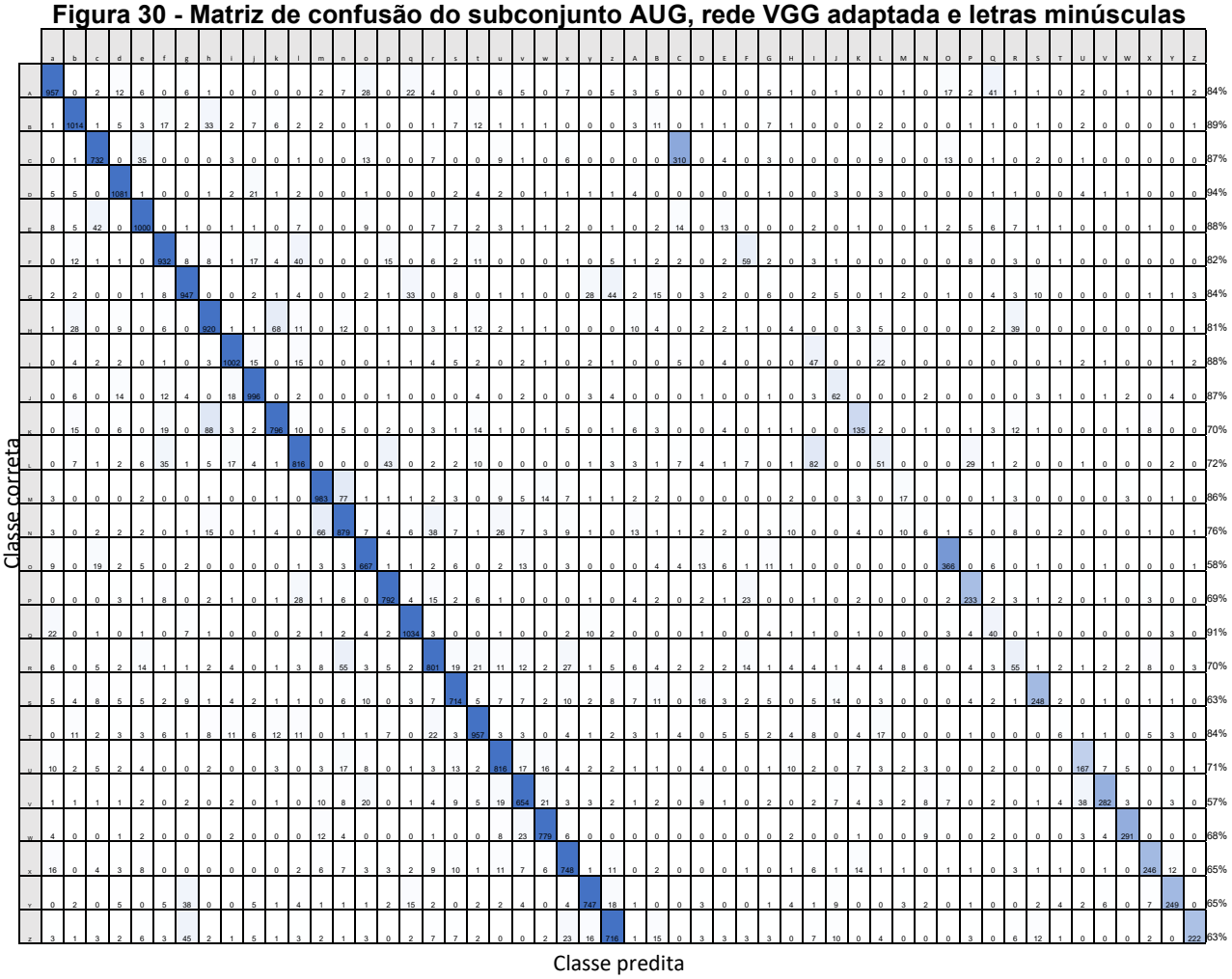
Fonte: Adaptado de Viard-Gaudin *et al.* (1999)

Como pode-se observar na Figura 29, alguns caracteres minúsculos (a) e maiúsculos (b) apresentam alta semelhança. Dessa forma, a rede neural confunde essas classes. Para essa figura foram selecionados caracteres maiúsculos e

minúsculos escritos pela mesma pessoa. Ou seja, quem escreveu “P” é a mesma pessoa que escreveu “p”. Mostrando assim, a baixa diferença de escritas nas versões maiúsculas e minúsculas.

Por conta da dimensão, a matriz de confusão do *dataset* AUG, foi dividida em duas figuras: minúsculas, na Figura 30, e minúsculas, na Figura 31.

Figura 30 - Matriz de confusão do subconjunto AUG, rede VGG adaptada e letras minúsculas

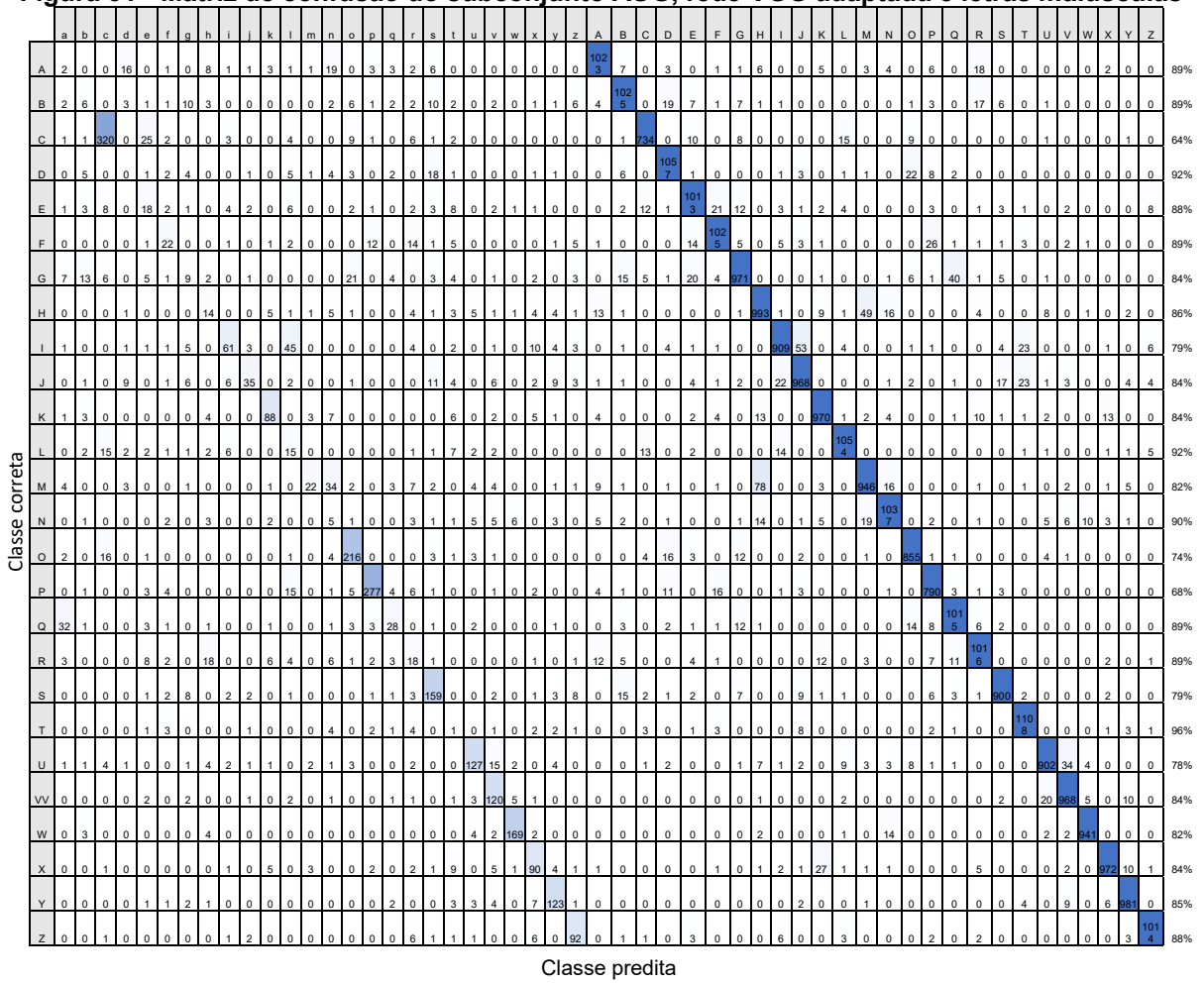


Fonte: Autoria própria (2021)

Pode-se observar nas matrizes de confusão, na Figura 30 e Figura 31, que há duas linhas diagonais paralelas em relação à diagonal principal, que representa a predição correta. Essas diagonais apresentam a confusão entre classes maiusculas e minusculas do mesmo caracter. Por exemplo, a letra “c”, minúscula, para a maioria das entradas é mapeada para a classe correta, mas em outras ocorrências é predita como “C”, maiúscula. Assim como a letra “C”, maiúscula, é mapeada corretamente na maioria das vezes, mas também como “c”, minúscula.

Observa-se na matriz de confusão que 310 vezes o caractere “c”, minúsuculo, foi confudido com o caractere “C”, maiúsculo. Enquanto o caractere “C”, maiúsculo, foi confudido 320 vezes com o caractere "c", minúsculo.

Figura 31 - Matriz de confusão do subconjunto AUG, rede VGG adaptada e letras maiúsculas



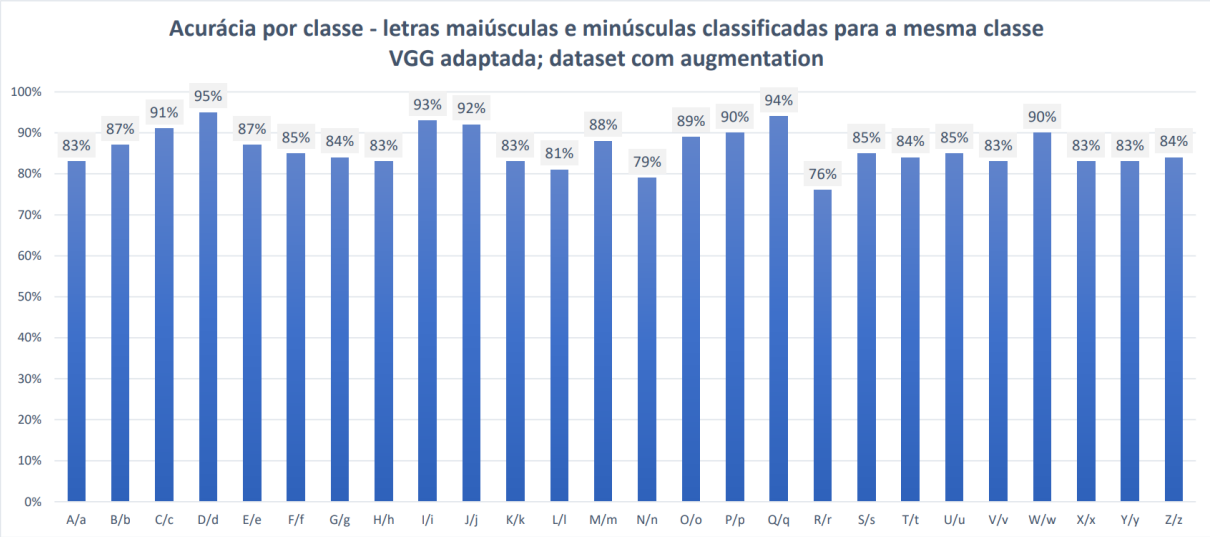
Fonte: Autoria própria (2021)

5.2.2.4 Experimento Extra

Por conta da confusão entre caracteres maiúsculos e minúsculos, foi realizado um novo experimento, com a proposta de classificar caracteres maiúsculos e minúsculos para a mesma classe. Ou seja, o caractere “C”, maiúsculo, é classificado para a mesma classe que o caractere “c”, minúsculo.

Como pode-se observar no Gráfico 15, em relação ao experimento anterior, que classificava letras maiúsculas e minúsculas em classes próprias, houve diminuição e aumento de acurácia em algumas classes.

Gráfico 15 - Acurácia por classe, para letras maiúsculas e minúsculas classificadas para a mesma classe, do subconjunto AUG e rede VGG adaptada



Fonte: Autoria própria (2021)

Para caracteres com características significativamente diferentes em suas versões maiúsculas e minúsculas, por exemplo “b”, sofreram diminuição de acurácia. Isso ocorre porque a rede neural recebe, para a mesma classe, características com poucas similaridades. Dessa forma, há maior dificuldade de generalização.

Já caracteres com características parecidas em suas versões maiúsculas e minúsculas, como “c” e “p”, apresentam melhora no valor de acurácia. Ao contrário da situação anterior, para essas letras a rede recebe mais entradas com alto grau de similaridade, melhorando, assim, a acurácia.

Essa abordagem pode ser aplicada em situações nas quais apenas se deseja saber qual a letra representada, sem considerar se o caractere é maiúsculo ou minúsculo.

Na Figura 32 pode-se observar a matriz de confusão deste experimento. Pode-se observar que, assim como nos experimentos anteriores, a diagonal principal se destaca, com maior quantidade de classificações corretas.

Figura 32 - Matriz de confusão do subconjunto AUG, rede VGG adaptada e letras maiúsculas e minúsculas classificadas para a mesma classe

Classe correta \ Classe predita	A/a	B/b	C/c	D/d	E/e	F/f	G/g	H/h	I/i	J/j	K/k	L/l	M/m	N/n	O/o	P/p	Q/q	R/r	S/s	T/t	U/u	V/v	W/w	X/x	Y/y	Z/z	
A/a	962	1	9	14	8	0	8	6	0	2	0	1	3	8	39	2	41	6	3	0	18	4	1	13	1	4	83%
B/b	1	1005	3	2	2	24	5	45	0	6	8	14	0	3	1	1	0	0	6	21	1	1	0	0	0	2	87%
C/c	2	1	1051	0	40	2	0	0	1	0	0	2	0	3	28	0	0	11	2	0	6	0	0	1	0	1	91%
D/d	8	3	0	1093	0	4	1	3	3	7	6	2	0	0	6	0	0	1	5	5	1	1	1	0	0	1	95%
E/e	7	4	55	0	1009	0	4	0	0	0	1	13	0	4	16	8	0	4	7	4	4	1	0	2	2	9	87%
F/f	0	12	2	3	2	978	10	7	2	12	8	47	0	0	0	25	1	9	3	16	0	0	0	0	1	10	85%
G/g	8	5	0	1	1	18	963	3	1	3	3	0	0	1	10	0	35	0	10	2	0	2	0	0	29	51	84%
H/h	1	40	0	1	3	14	2	953	0	1	90	14	1	6	0	2	0	3	0	14	2	2	0	1	0	1	83%
I/i	0	4	8	0	0	0	0	3	1065	17	1	25	0	0	0	2	0	5	4	7	2	2	0	2	0	1	93%
J/j	1	5	0	12	0	15	6	0	15	1052	2	7	1	0	2	0	0	0	5	8	0	0	0	0	5	10	92%
K/k	1	16	2	8	2	14	2	80	3	2	960	8	1	7	0	1	0	8	3	10	5	3	5	9	1	0	83%
L/l	0	10	4	0	9	48	3	3	40	7	3	934	0	0	0	42	2	1	3	31	0	4	0	1	1	2	81%
M/m	0	0	0	0	0	0	0	3	0	0	1	0	1008	78	1	1	1	5	1	0	9	10	22	6	0	0	88%
N/n	6	3	6	0	4	0	1	13	0	0	13	0	65	911	12	9	7	47	11	5	17	7	6	10	1	0	79%
O/o	10	1	33	2	8	0	1	0	0	1	0	2	3	12	1023	2	0	3	11	0	9	22	1	6	1	3	89%
P/p	2	2	1	0	3	22	2	0	3	0	1	30	1	11	2	1037	6	12	6	6	0	1	0	4	0	2	90%
Q/q	16	0	1	1	2	1	8	1	0	0	0	0	0	7	11	2	1088	2	1	0	0	0	0	6	7	0	94%
R/r	5	1	15	2	11	7	1	13	4	0	4	3	7	80	3	15	4	879	22	18	12	11	2	24	4	7	76%
S/s	1	9	6	5	8	8	16	2	8	5	2	2	4	9	14	3	3	19	985	3	9	3	1	15	2	12	85%
T/t	0	18	3	6	4	10	1	7	17	7	8	32	0	2	1	7	0	21	4	969	4	8	0	6	9	4	84%
U/u	16	2	6	4	6	0	0	2	1	0	4	1	6	17	6	0	2	6	9	5	977	34	31	10	1	2	85%
V/v	2	1	1	1	2	1	1	0	4	1	1	3	3	10	17	0	1	6	12	5	50	958	50	5	13	3	83%
W/w	1	2	1	0	2	0	0	1	2	0	1	0	20	3	1	0	0	3	1	0	20	39	1041	14	2	0	90%
X/x	15	1	8	1	5	4	0	0	1	2	9	1	7	15	3	7	2	18	18	8	10	22	7	960	14	13	83%
Y/y	0	2	0	6	2	5	51	1	2	11	0	4	2	3	3	2	20	3	6	11	5	16	1	13	956	29	83%
Z/z	4	6	6	1	11	6	49	2	2	6	0	3	3	1	4	1	1	10	17	1	2	0	0	19	25	971	84%

Fonte: Autoria própria (2021)

6 CONCLUSÃO

Neste trabalho, foram apresentados resultados para a classificação dos caracteres manuscritos presentes no *dataset* IRONOFF. A base de dados IRONOFF foi utilizada em sua versão original e com *data augmentation*. Essas bases foram divididas em 6 subconjuntos, dividindo ambas as versões em três: apenas com caracteres maiúsculos, apenas com caracteres minúsculos e com ambos.

Para realizar a classificação dos caracteres presentes na base de dados IRONOFF, foi implementa a rede LeNet-5, por encontrar vasto conhecimento sobre a mesma na literatura e considera-la uma rede importante na história das redes neurais convolucionais. Também foi implementada uma rede VGG adaptada, com inspiração na VGG-16 e que apresenta como principais características o número crescente de filtros e a presença de camadas de pooling, após cada camada convolucional.

A rede VGG adaptada apresentou os melhores resultados, chegando a 92,61% para o subconjunto AUG_MAI (maiúsculas com *data augmentation*), 86,65% para o subconjunto AUG_MIN (minúsculas com *data augmentation*) e 79,67% para o subconjunto AUG (ambas com *data augmentation*).

Foi apresentado neste trabalho o impacto do processo de *data augmentation* para a classificação do *dataset* IRONOFF. O melhor resultado para o *dataset* sem ampliação de dados apresentou acurácia de 74,16%, enquanto o melhor resultado para a base de dados com *data augmentation* obteve acurácia de 92,61%. Por isso, entende-se essa etapa como fundamental.

Como discutido, alguns caracteres apresentaram maior dificuldade de generalização para o subconjunto AUG_MIN, como “r”, e outros apresentaram baixa acurácia no subconjunto AUG, por conta da alta similaridade de suas versões maiúsculas e minúsculas, como “p” e “c”.

Por fim, através dos dados de acurácia e das matrizes de confusão, observa-se que a rede VGG adaptada neste trabalho alcançou um bom resultado na classificação dos caracteres manuscritos do *dataset* IRONOFF, realizando a predição correta na maioria dos testes e cumprindo assim o objetivo deste trabalho.

6.1 Trabalhos futuros

Ainda há muito o que explorar sobre o assunto proposto neste trabalho. Existem questões em aberto e novos experimentos devem ser realizados com o objetivo de comparar a rede VGG adaptada com outras redes presentes na literatura e melhorar o resultado para o subconjunto AUG. Esta seção apresenta algumas propostas para a continuação deste trabalho.

A primeira proposta consiste em realizar a comparação com outras redes neurais presentes na literatura, como a VGG-16 e AlexNet. Dessa forma, busca-se comparar a rede VGG adaptada com outras redes já desenvolvidas, comparando valores de acurácia e perda dessas redes com a VGG adaptada.

A segunda proposta consiste em implementar uma solução mista, como alguns dos trabalhos relacionados realizam, combinando a VGG adaptada com redes do tipo RNN. Dessa forma, obtendo dados para comparar se a acurácia e perda melhoram ou não com essa abordagem.

A terceira proposta consiste em avaliar a rede VGG adaptada com outras bases de dados utilizadas na literatura, como a MNIST.

Por fim, propõe-se utilizar a VGG adaptada para classificar caracteres reconstituídos e compará-la com outras redes neste contexto.

REFERÊNCIAS

AHMAD, J; MUHAMMAD, K; BAIK, S. Data augmentation-assisted deep learning of hand-drawn partially colored sketches for visual search. **PLOS ONE**. v. 12, ago. 2017

AIRES, S. B. K. **Reconhecimento de caracteres manuscritos baseados em regiões perceptivas**. 2005. 97f. Dissertação (Mestrado em Informática Aplicada) - Pontifıcia Universidade Católica Do Paraná, Curitiba, 2005.

COPPIN, B. **Inteligqência Artificial**. 1. ed. Rio de Janeiro: Grupo GEN, 2010.

DEO, R. C. Machine Learning in Medicine. Circulation, Dallas (TX), v. 132, n. 20, nov. 2015.

ELSAWY, A; LOEY, M; EL-BAKRY, H. Arabic Handwritten Characters Recognition using Convolutional Neural Network. **WSEAS TRANSACTIONS on COMPUTER RESEARCH**. v. 5, p. 11-19, jan. 2017

FREDJ, H. B.; *et al.* Parallel implementation of Sobel filter using CUDA. In: PROCEEDINGS. INTERNATIONAL CONFERENCE ON CONTROL, AUTOMATION AND DIAGNOSIS (ICCAD). 2017, Hammamet. **Anais…** Hammamet: 2017. p. 209-212.

GOODFELLOW, I; BENGIO, Y; COURVILLE, A. **Deep Learning**. Cambridge: The MIT Press, 2016.
GRAUPE, D. **Principles of Artificial Neural Networks**. 2. ed. World Scientific, 2007.

GRAVES, A; *et al.* A Novel Connectionist System for Unconstrained Handwriting Recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence,** v.31, n. 5, p. 855-868, 2000

HAN, L. Y**. Recuperação da trajetória online de caracteres latinos utilizando deep learning**. 2020. 95f. Dissertação de Mestrado em Ciência da Computação — Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2020.

HAYKIN, S. O. **Neural Networks and Learning Machines**. 3. ed. Pearson, 2009.

JABBAR, H. K.; KHAN, R. Z. Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study). In: PROCEEDINGS. Computer Science, Communication and Instrumentation Devices. 2014, Kochi. **Anais…** Kochi: 2014. p. 163-172

LANGLEY, P; SIMON, H. A. Applications of machine learning and rule induction. **Communications of the ACM,** v. 38, n. 11, p. 54-64, nov. 1995

LECUN, Y; BENGIO, Y; HINTON, G. Deep Learning. **Nature,** v. 521, p. 436-444, 2015

LECUN, Y; *et al.* Backpropagation applied to handwritten zip code recognition. **Neural Computation**, p. 541-551. 1989

MAALEJ, R.; KHERALLAH, M. Convolutional Neural Network and BLSTM for Offline Arabic Handwriting Recognition. **International Arab Conference on Information Technology**, p. 1-6, 2018.

MARQUES FILHO, O; VIERA NETO, H. **Processamento Digital de Imagens**, 1 ed. Rio de Janeiro: Brasport, 1999.

NAM, N. T; HUNG, P. D. Padding Methods in Convolutional Sequence Model: An Application in Japanese Handwriting Recognition. In: Proceedings of the 3rd International Conference on Machine Learning and Soft. 2019, Nova Iorque, Estados Unidos. **Anais…** Nova Iorque: 2019, p. 138–142.

NASH, W.; DRUMMOND, T.; BIRBILIS, N. A review of deep learning in the study of materials degradation. ***npj Mater Degrad*.** v. 2, p. 1-12, 2018

NORVIG, P; RUSSEL, S. **Inteligência Artificial**. 3. ed. Grupo GEN, 2013

O'SHEA, K; NASH, R; An Introduction to Convolutional Neural Networks. **arXiv e-prints,** nov. 2015

PAL, A.; SINGH, D. Handwritten English Character Recognition Using Neural Network, **International Journal of Computer Science & Communication**. v. 1, n. 2, p. 141-144, 2010

PERWEJ, Y.; CHATURVEDI, A. Neural Networks for Handwritten English Alphabet Recognition. **International Journal of Computer Applications,** v. 20, n. 7, abr. 2011

PHAM, V; *et al.* Dropout Improves Recurrent Neural Networks for Handwriting Recognition. In: PROCEEDINGS. 14TH INTERNATIONAL CONFERENCE ON FRONTIERS IN HANDWRITING RECOGNITION. 2014, Crete. **Anais…** Crete: 2014. p. 285-290.

PLAMONDON, R; SRIHARI, S. N. Online and off-line handwriting recognition: a comprehensive survey. **IEEE Transactions on Pattern Analysis and Machine Intelligence,** v. 22, n. 1, p. 63-84, 2000

PRADEEP, J; SRINIVASAN, E; HIMAVATHI, S. Neural network based handwritten character recognition system without feature extraction**. International Conference on Computer, Communication and Electrical Technology**, p. 40-44, 2011.

RABBY, A. S. A; *et al*. BornoNet: Bangla Handwritten Characters Recognition Using Convolutional Neural Network. **Procedia Computer Science**. v. 143, p. 528-535, jan. 2018

RAHMAN, M; *et al.* Bangla Handwritten Character Recognition using Convolutional Neural Network**. International Journal of Image, Graphics and Signal Processing**, v 7, p. 42-49, jul. 2015

RAWAT, W; WANG, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. **Neural Computation,** v. 29, n. 9, p. 2352-2449, set. 2017

SILVA, F.M.D.; *et al.* **Inteligência Artificia**l. 1. ed. Porto Alegre: Grupo A, 2019

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. **arXiv 1409.1556,** set. 2014

SRIVASTAVA, N; *et al.* Dropout: A Simple Way to Prevent Neural Networks from Overfitting. **The Journal of Machine Learning Research,** v. 15, n. 1, p. 1929–1958, jan. 2014

SUTHA, J; RAMARAJ, N. Neural Network Based Offline Tamil Handwritten Character Recognition System. In: COMPUTATIONAL INTELLIGENCE AND

MULTIMEDIA APPLICATIONS, INTERNATIONAL CONFERENCE. 2007, Sivakasi, India. **Anais…** Sivakasi: 2007, p. 446-450.

TAPPERT, C. C.; SUEN, C. Y; WAKAHARA, T. The state of the art in online handwriting recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence,** v. 12, n. 8, p. 787-808, 1990

TRA, V.; *et al.* Bearing Fault Diagnosis under Variable Speed Using Convolutional Neural Networks and the Stochastic Diagonal Levenberg-Marquardt Algorithm. **Sensors**. v. 17, dez. 2017

VELHO L.; FRERY, A. C.; GOMES, J. **Image processing for computer graphics and vision**, 2 ed. Londres: Springer, 2009.

VIARD-GAUDIN, C. P.; *et al.* The IRESTE On/Off (IRONOFF) dual handwriting database. In: PROCEEDINGS OF THE FIFTH INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION. 1999, **Anais…** 1999. p. 455-458.

XU, Q; PAN, G. SparseConnect: regularising CNNs on fully connected layers. **Electronic Letters**, v. 53, n. 18, p. 1246-1248, ago. 2017

YANMEI, H; BO, W; ZHAOMIN, Z. An improved LeNet-5 model for Image Recognition. In: PROCEEDINGS OF THE 2020 4TH INTERNATIONAL CONFERENCE ON ELECTRONIC INFORMATION TECHNOLOGY AND COMPUTER ENGINEERING. 2020, Xiamen, China**. Anais…** Xiamen: 2020, p. 444-448.

YUAN, A.; *et al.* Offline handwritten English character recognition based on convolutional neural network. **10th IAPR International Workshop on Document Analysis Systems**, p. 125-129, 2012.

ZHANG, C.L.; *et al.* In Defense of Fully Connected Layers in Visual Representation Transfer. **Advances in Multimedia Information Processing,** v. 10736, p. 807-817, 2017.