Darlan de Castro Silva Filho

Reconhecimento de Caracteres Utilizando Redes Neurais Convolucionais para Auxiliar nas Correções do Sistema Multiprova

Natal – RN

Julho de 2022

Darlan de Castro Silva Filho

Reconhecimento de Caracteres Utilizando Redes Neurais Convolucionais para Auxiliar nas Correções do Sistema Multiprova

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Helton Maia

Universidade Federal do Rio Grande do Norte – UFRN

Departamento de Engenharia de Computação e Automação – DCA

Curso de Engenharia de Computação

Natal – RN

Julho de 2022

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

Silva Filho, Darlan de Castro.
Reconhecimento de caracteres utilizando redes neurais convolucionais para auxiliar nas correções do sistema multiprova / Darlan de Castro Silva Filho. - 2022.
62 f.: il.

Monografia (graduação) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Curso de Engenharia de Computação.
Orientador: Prof. Dr. Helton Maia Peixoto.

1. Reconhecimento ótico de caracteres - Monografia. 2. Inteligência artificial - Monografia. 3. Aprendizado de máquina - Monografia. 4. Redes neurais convolucionais - Monografia. I. Peixoto, Helton Maia. II. Título.

RN/UF/BCZM CDU 004

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

Darlan de Castro Silva Filho

Reconhecimento de Caracteres Utilizando Redes Neurais Convolucionais para Auxiliar nas Correções do Sistema Multiprova

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Helton Maia

Trabalho aprovado. Natal – RN, 22 de Julho de 2022:

Prof. Dr. Helton Maia - Orientador
UFRN

Prof. Dr. Rex Antonio da Costa Medeiros - Convidado
UFRN

Prof. Dr. Orivaldo Vieira de Santana Junior - Convidado
UFRN

Natal – RN
Julho de 2022

Dedico este trabalho à Deus, meus pais, minha família, meus amigos e a todos que me ajudaram durante todo o período da graduação. Grato por tudo.

AGRADECIMENTOS

Ao meu orientador prof. Dr. Helton Maia, pela orientação e apoio durante toda a elaboração deste trabalho.

Aos profs. da equipe Multiprova, pela introdução ao tema do trabalho e pela confiança depositada em mim.

A todos da equipe acadêmica da UFRN que direta ou indiretamente fizeram parte da minha formação.

A todos aqueles que contribuíram, de alguma forma, para a realização deste trabalho.

RESUMO

O software Multiprova foi muito utilizado durante os períodos de ensino remoto na Universidade Federal do Rio Grande do Norte (UFRN), com a volta presencial das aulas o processo de correção de provas precisou ser melhorado para facilitar a adaptação do sistema às aulas presenciais, uma dessas melhorias foi a correção automática de cartões respostas dos alunos. O reconhecimento ótico de caracteres é uma técnica recente muito utilizada para máquinas realizarem a leitura de textos escrito por humanos. O objetivo central do trabalho é desenvolver um processo de reconhecimento de caracteres manuscritos a fim de melhorar e facilitar a correção de provas no software Multiprova. Para essa finalidade foram utilizadas redes neurais convolucionais para realizar essa tarefa. Com a coleta de imagens feitas pelos próprios alunos da UFRN, foram analisados e comparados cenários diferentes com o incremento das configurações das redes com o intuito de gerar redes neurais com as melhores taxas de precisão. Com isso foram obtidos ótimos níveis de acurácia, permitindo alta confiabilidade no software e mais segurança e facilidade na correção das provas.

Palavras-chaves: Reconhecimento ótico de caracteres; Inteligência Artificial; Aprendizado de máquina; Redes neurais convolucionais.

ABSTRACT

The Multiprova software was widely used during remote teaching at the Federal University of Rio Grande do Norte. With the return of classroom classes, the exam correction process needed to be improved to facilitate the adaptation of the system to presential courses, one of these improvements was the automatic correction of student response cards. Optical character recognition is a recent technique widely used for machines to read text written by humans. This academic work aims to develop a handwritten character recognition process to improve and facilitate exam correction in Multiprova software. For this purpose, convolutional neural networks were used to perform this task. With the UFRN student's images, different scenarios were analyzed and compared with the increment of network configurations to generate neural networks with the best accuracy rates. As a result, excellent levels of accuracy were obtained, allowing high reliability in the software and more security and ease in exam correction.

Keywords: Optical Character Recognition; Artificial Intelligence; Machine Learning; Convolutional Neural Networks.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
JSON	*JavaScript Object Notation*
API	*Application Programming Interface*
REST	*Representational State Transfer*
SQL	*Structured Query Language*
NoSQL	*No Structured Query Language*
CNN	*Convolutional Neural Network*
IA	*Inteligência Artificial*
UFRN	*Universidade Federal do Rio Grande do Norte*
ECT	*Escola de Ciências e Tecnologia*
OCR	*Optical Character Recognition*

SUMÁRIO

1 INTRODUÇÃO

De acordo com John McCarthy a Inteligência Artificial - ou IA - pode ser definida como a ciência e engenharia de fazer máquinas inteligentes, especialmente programas de computador inteligentes. Ela está relacionado à tarefa semelhante de usar computadores para entender a inteligência humana, no entanto a IA não precisa se limitar a métodos biologicamente observáveis (MCCARTHY, 2007). Com a Inteligência Artificial é possível obter uma maior produtividade, otimização de processos e uma análise mais confiável em dados complexos.

A IA é subdividida em diversas áreas, entre elas pode-se citar o *Machine Learning* (aprendizado de máquina) em que as máquinas usam algoritmos para tomar decisões e interpretar dados, para assim executar tarefas automaticamente, ou seja, as máquinas não serão programadas para realizar ações específicas. O aprendizado de máquina possui uma abordagem para solução de problemas diferente da programação tradicional. Nos sistemas comuns, são escritas as regras e os fluxos no qual o software deverá seguir. Já para o *Machine Learning*, espera-se que o sistema utilize os dados para criar as próprias regras de funcionamento, invertendo o processo de construção usual.

A técnica de *Deep Learning* ou Aprendizagem Profunda, explicada de maneira simples, consiste em uma subdivisão do *Machine Learning*. Em geral, é um aprendizado de máquina que visa "ensinar" os computadores a agir e interpretar dados de uma maneira análoga ao reconhecimento de padrões realizado pelos humanos. A *Deep Learning* tem como função tratar sobre as oportunidades de aprendizagem através do uso de redes neurais para facilitar e melhorar operação e procedimentos, por exemplo: visão computacional como é tratado no livro *Machine learning in computer vision* (SEBE et al., 2005), reconhecimento de fala visto no *paper Speech emotion classification using machine learning algorithms* (CASALE et al., 2008), classificação de imagens como no artigo *Deep learning for hyperspectral image classification: An overview* (LI et al., 2019), processamento de lingugem natural que está presente no *Evaluation of machine learning methods for natural language processing tasks* (DAELEMANS; HOSTE, 2002) entre outros.

Ao utilizar o método de Aprendizagem Profunda, os desenvolvedores têm conseguido resultados satisfatórios em diferentes contextos de aplicação. Utilizando múltiplas camadas para o processamento de dados e a não linearidade da rede, é possível obter uma representação complexa (de modo a manter uma alta taxa de acerto) e genérica (de modo que possa se adaptar a entradas que não sejam necessariamente iguais ou extremamente parecidas as utilizadas no treinamento) dos dados de forma hierárquica.

1.1 Motivação

A motivação para a realização deste trabalho é a iminente volta às aulas presenciais na Universidade Federal do Rio Grande do Norte (UFRN) e a necessidade do software de avaliação desenvolvido na UFRN, Multiprova, em se adaptar às provas em papel.

O Multiprova nasceu como um sistema de avaliação dos estudantes e ganhou um módulo de resolução de provas online totalmente remoto e foi essencial para a comunidade acadêmica da UFRN durante os dois anos de ensino à distância que foram necessários para a amenização dos efeitos da pandemia do novo Coronavírus (Covid-19).

Com a volta às salas de aula o Multiprova deve se ajustar ao ensino presencial e com isso surge um problema: Como manter o sistema benéfico à comunidade acadêmica sabendo que presencialmente o método de avaliação de provas escritas - prova com correção manual do professor - já funciona bem?

Como solução desse problema foi pensado e construído pela equipe do Multiprova um cartão resposta para provas em papel que pode ser observado na Figura 1. O cartão resposta vai além dos cartões respostas comuns - que só permitem a correção automática de questões de múltipla escolha - e possibilita a utilização de questões do tipo múltipla escolha, associação de colunas e verdadeiro ou falso.

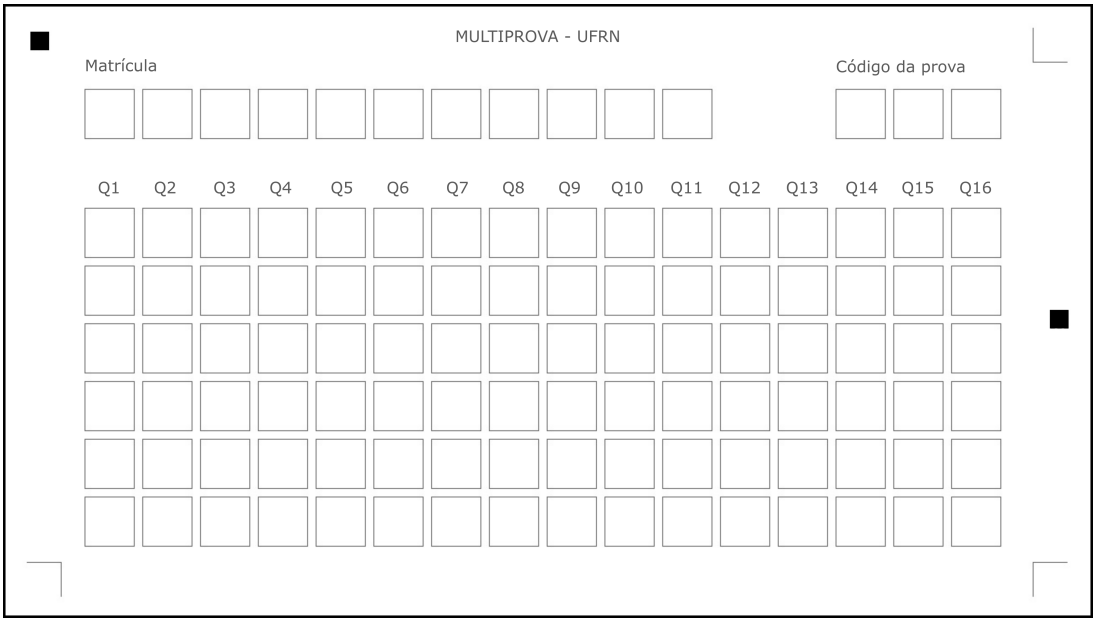


Figura 1 – Novo cartão resposta desenvolvido pela equipe do Multiprova.

Fonte: Acervo do autor.

Visto que serão adicionados novos tipos de questões, a abordagem tradicional de marcar a alternativa correta não funciona (para associação de colunas e verdadeiro ou falso). Portanto, a criação deste cartão genérico o suficiente é necessário para a adaptação

a essas novas opções, bem como também se mantém aberto para novos tipos de questões no futuro.

O problema, então, se resume ao desenvolvimento de uma ferramenta computacional, que tem como objetivo, automatizar o processo de correção dos cartões respostas provenientes do sistema multiprova. Neste tipo de cartão resposta, os alunos escrevem manualmente suas respostas.

1.2 Objetivos

Partindo desse problema é possível destrinchar o processo de resolução em objetivos menores:

- Recuperar a imagem do cartão resposta e isolar os caracteres presentes para posterior classificação dos caracteres pelas redes neurais;
- Desenvolver um processo de reconhecimento dos caracteres isolados escritos no cartão.
- Permitir que o professor tenha a comodidade de corrigir o cartão sem a necessidade de enviá-lo a uma máquina que faça essa correção.

1.3 Estrutura do trabalho

Este trabalho apresenta uma introdução sobre o tema, mostrando os fatores que motivam a implantação da ideia, informações sobre os pontos chave do trabalho, além da justificativa e dos objetivos. Em sequência, o Capítulo 2 apresenta as ferramentas necessárias para um melhor entendimento do projeto como um todo, *softwares* utilizados, pesquisas e livros consultados e uma visão geral da base do trabalho (*Machine Learning* e Redes Neurais Convolucionais). O Capítulo 3, por sua vez, explica a metodologia para desenvolvimento da parte prática do trabalho, com explicações acerca das decisões tomadas durante a elaboração do trabalho. Já o Capítulo 4 expõe os resultados encontrados, as discussões, considerações e interpretações pertinentes acerca desses resultados. Por fim, o Capítulo 5 traz as principais conclusões e contribuições deste trabalho baseados nos objetivos bem como pontos de melhora futura.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Trabalhos relacionados

A utilização de redes neurais convolucionais (CNN's) para o reconhecimento de caracteres manuscritos é um tópico relativamente novo - o conceito de CNN foi apresentado por (LECUN et al., 1998) no artigo *Gradient-Based Learning Applied to Document Recognition* em 1998 - e com muito a desenvolver. Em razão disso não existem muitos trabalhos expressivos para referência já que se trata de uma ramificação desse problema (reconhecer caracteres manuscritos em um cartão-resposta). No entanto, já existiam métodos para a utilização de IA no reconhecimento óptico de caracteres (OCR - *Optical Character Recognition*), ainda que não se utilizassem redes neurais convolucionais (CNN's).

O trabalho *Reconhecimento de Caracteres Numéricos Manuscritos* (VELOSO et al., 1998) é bastante importante já que tem como objetivo realizar o reconhecimento através do treinamento de uma rede neural de perceptrons de multicamadas - que é a rede base para a rede neural convolucional que será utilizada neste trabalho. O trabalho data de 1998, o que também mostrará como esse tópico evoluiu após 24 anos.

O trabalho de reconhecimento de caracteres manuscritos ainda traz pontos interessantes como uma análise topológica de como os caracteres são reconhecidos. Essa análise aponta características importantes dos números como quantidade e posição das cavidades, número de interseções do traçado com o eixo principal e distribuição pictorial.

Para fins de comparação o trabalho *Reconhecimento de Caracteres Numéricos Manuscritos* obteve como precisão final a taxa de 94,92% com um conjunto de treinamento de 4000 amostras distribuídas igualmente entre as classes dos dígitos de 0 a 9.

Considerando as diferenças dos resultados nos trabalhos apresentados, será possível verificar como o reconhecimento de dígitos avançou, principalmente depois da utilização de uma rede neural convolucional (uma forma mais avançada para reconhecimento de imagens baseada em uma rede neural de perceptrons de multicamadas) para o reconhecimento de caracteres.

Outro trabalho importante para o desenvolvimento de *Machine Learning* é o *dataset* EMNIST (COHEN et al., 2017). Esse conjunto de imagem deriva do *NIST Special Database 19* que é um super conjunto de imagens de formas e caracteres manuscritos. As imagens fornecidas tem tamanho de 28 x 28 *pixels*. Existem 6 diferentes tipos de subdivisões para o EMNIST:

- EMNIST ByClass com 814255 amostras e 62 classes não-balanceadas;
- EMNIST ByMerge com 814255 amostras e 47 classes não-balanceadas;
- EMNIST Balanced com 131600 amostras e 47 classes balanceadas;
- EMNIST Letters com 145600 amostras e 26 classes balanceadas;
- EMNIST Digits com 28000 amostras e 10 classes balanceadas;
- EMNIST MNIST com 70000 amostras e 10 classes balanceadas;

Onde as divisões *EMNIST Letters* e *EMNIST Digits* aparentam se encaixar bem como dados de treinamento neste trabalho. Um exemplo das amostras contidas nesse *dataset* pode ser visto na Figura 2.

Figura 2 – Exemplo dos dígitos encontrados no dataset EMNIST.

Fonte: (MNIST. . . , 2022)

2.2 Machine Learning

A utilização do *Machine Learning* tanto na pesquisa quanto na indústria, cresceu muito nos últimos anos. Os algoritmos tem se tornado cada vez mais eficientes, e com a popularização da internet ainda em expansão, um grande volume de informação vem sendo gerado. Desta forma, esses dados podem ser utilizados para alimentar modelos de aprendizagem de máquina com objetivos diversos.

Os métodos mais conhecidos de *Machine Learning* para aprendizagem dos modelos são:

1. Aprendizagem por reforço;
2. Aprendizagem supervisionada;
3. Aprendizagem não supervisionada.

De acordo com (AYODELE, 2010), esses três métodos podem ser explicados como:

Para a aprendizagem por reforço, o algoritmo aprende como se comportar dada uma observação do contexto em que está inserido. Toda ação que é tomada tem um impacto naquele ecossistema e o ambiente fornece uma resposta, positiva ou negativa acerca da ação tomada, para orientar o aprendizado da máquina. Esse tipo de aprendizagem é voltado para redes que tem como objetivo se especializar em tomadas de decisão. Um caso de mundo real que pode se beneficiar desse tipo de abordagem é o tráfego em grandes cidades que, através de uma rede neural especializada em tomar decisões, pode minimizar o tempo de parada dos carros em semáforos ajustando o tempo de espera baseado em métricas analisadas em tempo real como é detalhado no artigo *Learning an Interpretable Traffic Signal Control Policy* (AULT; HANNA; SHARON, 2019).

No caso do método de aprendizagem supervisionada, seu uso é bastante comum para solução por exemplo, de problemas que envolvem classificação - um vez que o objetivo é ensinar a máquina a reconhecer padrões. Mais genericamente esse modelo é recomendado para qualquer problema onde identificar uma classe previamente especificada é útil. Exemplos de sua aplicação são reconhecer e diferenciar imagens de gatos e cachorros como mostra o artigo *Cats and Dogs* (PARKHI et al., 2012), diferenciar jogadas do Pedra, Papel e Tesoura no projeto *Building a rock paper scissors AI using Tensorflow and OpenCV* (ACHARYA, 2020) e a detecção de câncer de pele no artigo *Machine learning on mobile: An on-device inference app for skin cancer detection* (DAI et al., 2019).

Já para a aprendizagem não supervisionada o desafio parece maior, é necessário fazer com que computador aprenda a realizar classificações e agrupamentos de dados ainda que não lhe seja ensinado diretamente apresentando exemplos de entrada e saída. Uma forma de fazer isso é com método da *clustering* (agrupamento), nessa abordagem o objetivo é encontrar similaridades entre os dados analisados e dispor os dados em conjuntos baseados em suas características. Uma aplicação muito comum é a clusterização das flores de Íris, onde se têm um conjunto de dados das características da flores (largura e comprimento da pétala e da sépala) e o objetivo é que as entradas seja categorizadas entre três conjuntos (Versicolor, Setosa e Virginica) que tem características distinta, essa pesquisa pode ser vista no artigo *A study of pattern recognition of Iris flower based on Machine Learning* (YANG, 2013).

2.3 Reconhecimento de caracteres em imagens

O reconhecimento de caracteres é um tipo de problema já bastante trabalhado dentro das áreas de IA e *Machine Learning*. Ainda que seja um problema conhecido desde o século XX, não existe uma solução universal, isto em razão das diferentes abordagens e modelagem do problema.

Normalmente são consideradas duas abordagens para a resolução do problema de reconhecimentos de caracteres em imagens, são elas:

1. Utilização de softwares especializados (já treinados) para leitura de textos de forma genérica em uma imagem;
2. Treinamento de redes neurais artificiais, que serão especializadas no reconhecimento dos caracteres do problema.

A utilização de *softwares* já treinados trás pontos fortes e fracos que podem não se encaixar nas necessidades do projeto. Tomando como exemplo um dos maiores projetos de reconhecimento de caracteres, o *Tesseract*, da Google, trás vantagens para projetos que necessitam de um software capaz de analisar textos em imagens que não necessariamente seguem um padrão definido.

Um exemplo de uma boa aplicação do *Tesseract* é no reconhecimento de placas de carros a partir de imagens de radares ou câmeras. As imagens podem variar em posição, luminosidade, visibilidade, entre outros, e o software é robusto o suficiente para lidar com esse tipo de adversidade.

Já um treinamento especializado para o problema analisado intensifica a ação de pontos fortes e minimiza ou até elimina os pontos fracos. Tomando como exemplo um problema onde os caracteres estão sempre nas mesmas posições, é mais vantajoso realizar a segmentação dessas imagens a fim de classificar imagem a imagem para maximizar a chance de acerto de cada caractere individualmente. Utilizando da segmentação é possível minimizar os pontos fracos como baixa luminosidade, perspectiva não alinhada com a câmera e baixa resolução o que, por um lado, traz menos robustez para trabalhar com imagens menos visíveis, quando por outro prepara melhor o software para receber as imagens com boa visibilidade, que é como se espera receber a imagem, por isso é uma melhor opção.

2.4 Redes neurais convolucionais

> Uma Rede Neural Convolucional (ou *Convolutional Neural Network* - CNN) é uma variação das redes de Perceptrons de Múltiplas Camadas, tendo sido inspirada no processo biológico de processamentos de dados visuais. De maneira semelhante aos processos tradicionais de visão computacional, uma CNN é capaz de aplicar filtros em dados visuais, mantendo a relação de vizinhança entre os *pixels* da imagem ao longo do processamento da rede. (VARGAS; CARVALHO; VASCONCELOS, 2016)

As CNN's são comumente compostas por múltiplas camadas, onde as mais relevantes para este trabalho serão detalhadas a seguir.

A camada convolucional tem como objetivo a filtragem da imagem. Isto é realizado com a aplicação durante um processo de varredura, do filtro (também conhecido como *kernel*) sob a imagem. Além disso, funções de ativação são inseridas nas saídas dos neurônios, provendo características adicionais ao sistema (ALVES, 2018).

> A função de ativação presente em cada neurônio é responsável por aplicar uma transformação nos dados recebidos. Normalmente, utilizam-se funções com algum grau de não-linearidade. A não linearidade das camadas intermediárias permite que as aplicações sucessivas dessas distorções tornem as categorias de saída linearmente separáveis. (VARGAS; CARVALHO; VASCONCELOS, 2016)

Um exemplo de ação de uma camada convolucional com *kernel* (3 x 3 x 3) pode ser observado na Figura 3.

A camada de *pooling* é geralmente utilizada logo após a camada convolucional. Ela tem como objetivo reduzir o tamanho dos dados - conferindo mais agilidade ao processo - enquanto seleciona as características mais expressivas da imagem.

"(...) por exemplo: a entrada é dividida em janelas 4 x 4 e de cada uma é selecionado um valor para os representar. Essa escolha pode ser feita por diversas funções, porém a mais utilizada é a função de máximo"(VARGAS; CARVALHO; VASCONCELOS, 2016).

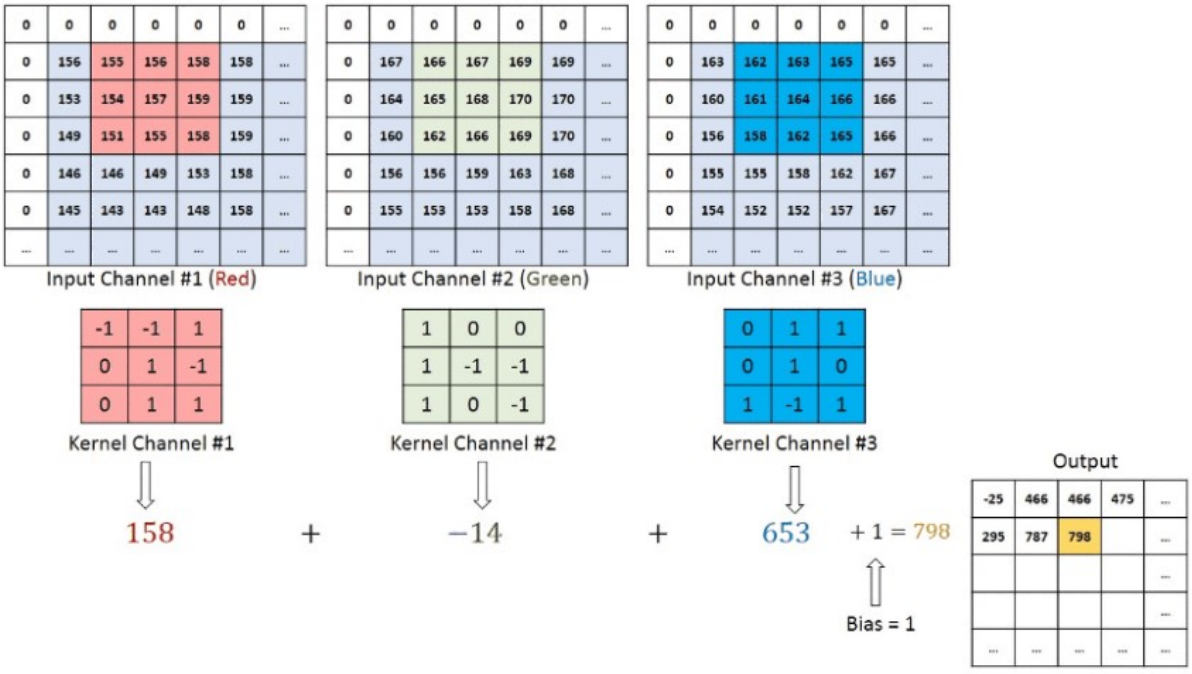


Figura 3 – Camada convolucional agindo sobre uma imagem com 3 canais.

Fonte: (ALVES, 2018)

A Figura 4 mostra uma camada de *pooling* máximo com filtro 2 x 2 aplicada sobre uma imagem 4 x 4.

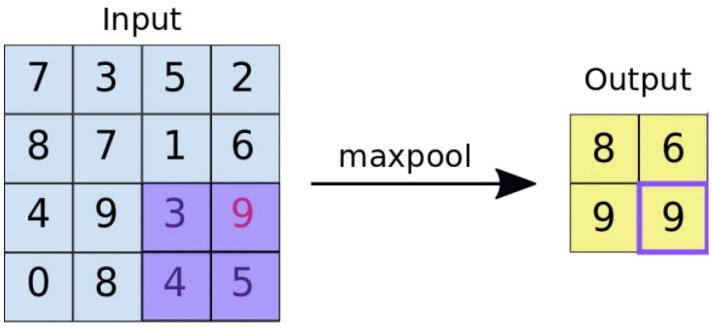


Figura 4 – Camada de pooling máximo agindo sobre uma imagem 4 x 4.

Fonte: (ALVES, 2018)

A camada *flatten* - ou camada de achatamento - nada mais é que uma camada que recebe uma entrada matricial e retorna um vetor unidimensional sem alterar os valores dos dados, somente a forma que essas valores são apresentados. Ela prepara os dados para a camada final que irá fazer efetivamente a predição.

Já a camada totalmente conectada é a camada final de uma CNN, é composta de várias subcamadas (camadas escondidas ou *hidden layers*) de neurônios. Como o seu nome indica, os neurônios presentes nessas subcamadas estão ligados com todos os outros neurônios da subcamada anterior e da subcamada seguinte. Por fim, a última subcamada

tem sempre um número predefinido de neurônios de saída, onde esse número é a quantidade de classes para a classificação da imagem.

A Figura 5 exemplifica uma camada totalmente conectada com 4 classes como saída.

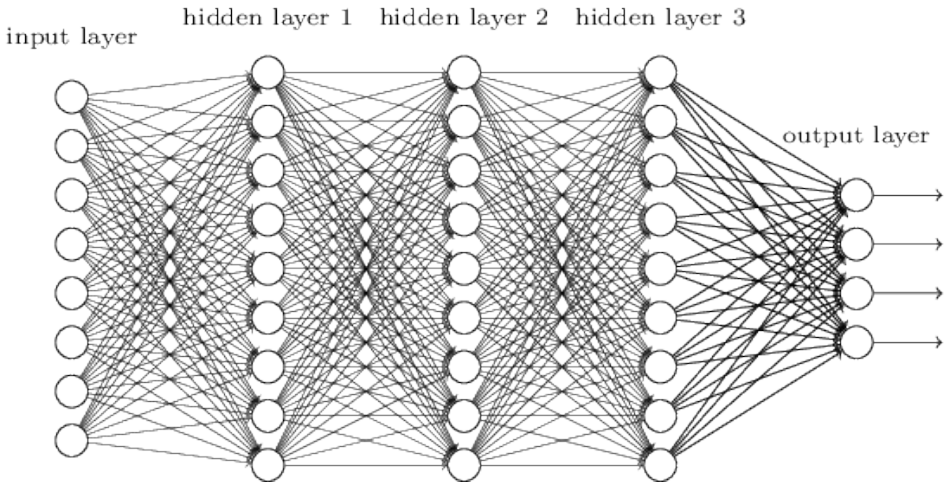


Figura 5 – Camada totalmente conectada.

Fonte: (ACADEMY, 2018)

Na Figura 6, é possível observar a estrutura da CNN em sua completude com todas as camadas sendo utilizadas.

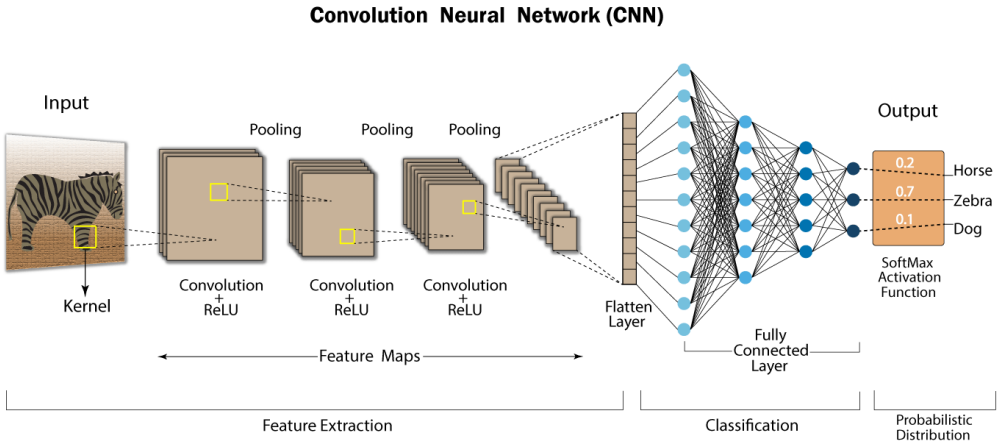


Figura 6 – CNN completa.

Fonte: (KALITA, 2022)

3 METODOLOGIA

3.1 Linguagens de programação

Neste trabalho foram utilizadas duas linguagens de programação. O Python no processamento das imagens dos cartões resposta e na criação e treinamento da rede neural. Já a segunda, a linguagem Javascript, na realização do porte das configurações da rede treinada para o ambiente *mobile* a fim de predizer as respostas dos alunos nos cartões.

Python (ROSSUM; JR, 1995) é uma linguagem de programação de alto nível, interpretada, dinâmica e multiplataforma criada por Guido van Rossum por volta dos anos 90. É uma das linguagens mais populares do mundo e que ganhou fama graças a sua sintaxe simples - o usuário tem uma curva de aprendizado altíssima, sua grande quantidade de bibliotecas nativas e de terceiros - que facilitam e aceleram muito o desenvolvimento, é totalmente grátis e têm atualizações constantes. É utilizada principalmente nas áreas de Inteligência Artificial, análise de dados, desenvolvimento web, *Big Data*, dentre outros.

JavaScript (FLANAGAN, 2006) é uma linguagem de programação de alto nível, interpretada e orientada a objetos que foi inicialmente criada por Brendan Eich em 1995 para a execução de *scripts* no lado do usuário, ou seja, nos navegadores web. Com o passar do tempo novas implementações do JavaScript surgiram e a linguagem saiu somente do campo da web. Um exemplo disso é o V8 - interpretador criado pela Google - que permite que a linguagem seja executada fora dos navegadores, o que confere muito mais liberdade e versatilidade.

3.2 Softwares auxiliares

Criado e mantido pela Google, o TensorFlow (ABADI et al., 2015) é uma biblioteca de código aberto para computação numérica e *Machine Learning*. Essa biblioteca agrupa vários algoritmos e modelos de *Deep Learning*, simplificando seu uso para o usuário final. Devido a sua facilidade de uso e integração com a linguagem Python, o TensorFlow é a ferramenta para o desenvolvimento de *Machine Learning* mais utilizada na atualidade.

Construída sobre o TensorFlow, o Keras (CHOLLET et al., 2015) é uma API de alto nível para criar e treinar modelos de *Deep Learning*. Com elementos fáceis de serem entendidos e aplicados, essa API é utilizada nos projetos de *Machine Learning* do Tensorflow já que ela está presente dentro do seu ecossistema e pode ser importada diretamente do pacote do TensorFlow para o código.

O OpenCV, ou *Open Source Computer Vision* (BRADSKI, 2000), é uma biblioteca de código aberto que foi inicialmente desenvolvida pela Intel no ano de 2000 visando facilitar o desenvolvimento de projetos com o foco em visão computacional. Está implementada em diversas linguagens de programação e é primariamente utilizada para análises e manipulações em imagens e vídeos. É a principal biblioteca para visão computacional pois contem uma abstração boa o suficiente de funções e algoritmos que seriam complexos de se implementar manualmente.

O React Native é uma biblioteca JavaScript criada pelo Facebook para a criação e manutenção de aplicativos mobile (Android e IOS) de forma nativa. A grande vantagem de utilizar uma biblioteca como essa é a facilidade em ter que aprender somente uma linguagem (JavaScript) e desenvolver nativamente para duas plataformas diferentes - o Android que usa Java e o IOS que usa Objective-C e Swift.

3.3 Aplicações web

Uma aplicação web é a implementação de um aplicativo em um navegador. Esse tipo de aplicação tem um servidor central onde os usuários (navegadores) podem fazer pedidos (requisições) com o objetivo de realizar o que o aplicativo se propõe a fazer. De forma resumida, as aplicações web são compostas de um lado do cliente, que é o que o usuário vê e interage, e o lado do servidor, que centraliza o processamento e armazenamento das informações. A comunicação entre cliente e servidor é feita mais comumente através do HTTP (ou *Hypertext Transfer Protocol*).

O HTTP é um protocolo de transferência de informações, sua primeira versão documentada é datada de 1995 e sua adoção foi rápida. Este protocolo permite vários tipos de solicitações através de métodos, onde as mais utilizadas podem ser vistas na Tabela 1, bem como seus casos de uso.

Métodos	**Casos de uso**
GET	Solicita a representação de um recurso específico e só deve retornar dados.
POST	Submete uma entidade a um recurso específico, frequentemente causando uma mudança no estado do recurso.
PUT	Substitui todas as atuais representações do recurso de destino pela carga de dados da requisição.
DELETE	Remove um recurso específico.
HEAD	Solicita uma resposta de forma idêntica ao método GET, porém sem conter o corpo da resposta.

Tabela 1 – Métodos HTTP mais comuns e seus casos de uso

Fonte: (CONTRIBUTORS, 2021, tradução livre)

JSON - ou *JavaScript Object Notation* - (PEZOA et al., 2016) é um formato de troca de informações entre sistemas. Tem como estrutura o formato chave-valor, as chaves devem ser *strings* únicas em um mesmo escopo dentro do JSON e os valores podem assumir os tipos: cadeia de caracteres, vetor, objeto, número, boleano ou nulo. É bastante utilizado para a comunicação entre o cliente e o servidor em aplicações web além de ser fácil de ser

entendido por seres humanos. Um exemplo de dados escritos em JSON pode ser observado na Figura 7.

```json
{
  "id": 1,
  "nome": "Beltrano da Silva",
  "idade": 43,
  "endereco": {
    "rua": "Rua 10",
    "bairro": "Centro"
  },
  "dependentes": [
    {
      "nome": "Cicrano da Silva",
      "idade": 16
    }
  ],
  "temHabilitacao": true,
  "numeroCPF": null
}
```

Figura 7 – Exemplo de dados escritos em JSON.

Fonte: De autoria própria.

O *back-end* é toda a infraestrutura que dá suporte a uma aplicação. No contexto das aplicações web o *back-end* é o responsável por receber, processar e responder as requisições feitas pela aplicação que o usuário está utilizando em seu dispositivo. Ainda no contexto das aplicações web existem algumas estruturas que podem ser utilizadas para montar um *back-end*, suas partes podem envolver API's - que processam e garantem a segurança e coerência dos dados e bancos de dados - que guardam os dados persistentemente na memória.

Uma API (*Application Programming Interface*) é somente um conjunto de regras e rotinas preestabelecidas por quem quer que a tenha construído a fim de padronizar o uso de um determinado serviço ou aplicação. Uma API do tipo REST (*Representational State Transfer*) é uma arquitetura que tem alguns pontos chaves em sua definição, os mais importantes são:

- Fornecer os dados em um formato normalizado tendo como base requisições HTTP;
- Utilizar uma comunicação *stateless* - Isso significa que nenhum dado do usuário é armazenado entre solicitações *GET* e todas as solicitações são independentes e desconectadas;
- Ter uma interface uniforme entre os componentes para que as informações sejam transferidas em um formato padronizado.

"Um banco de dados é uma coleção organizada de informações - ou dados - estruturadas, normalmente armazenadas eletronicamente em um sistema de computador."(ORACLE, 2021). Existem vários tipos de bancos de dados onde a grande divisão entre eles se dá em bancos de dados relacionais (SQL) - onde os dados são armazenados em tabelas e mantém relações uns com os outros - e bancos de dados não relacionais (NoSQL) - que são todos os tipos de bancos que não seguem as regras de bancos relacionais.

O MongoDB é o banco de dados NoSQL mais utilizado no mundo. Tem alta performance, com alta capacidade de escalabilidade tanto vertical quanto horizontalmente. É um banco de dados orientado a documentos que permite que os dados sejam inseridos no padrão chave-valor utilizando o JSON como o formato dos dados.

3.4 Sistema Multiprova

O software Multiprova é uma aplicação constituída de duas partes principais:

- Lado do cliente;
- Lado do servidor.

O lado do cliente - ou *front-end* - (interface em que alunos e professores interagem) é composta por duas frentes. A aplicação feita para navegadores - tanto *desktop* quanto *mobile* - e o aplicativo *mobile* nativo.

A aplicação para navegadores é a parte principal onde os discentes e docentes podem executar por completo o fluxo de provas virtuais, desde a criação das provas por parte dos professores até a execução delas pelos alunos. Essa aplicação é feita com React.

Já a aplicação nativa para *mobile* é feita com React Native e tem como função ser uma interface entre as correções de provas presenciais e o lado do servidor do sistema Multiprova.

O lado do servidor (ou *back-end*) é feito com JavaScript utilizando o modelo de API REST e o banco de dados NoSQL MongoDB. O servidor é a parte principal do sistema já que guarda todos os dados de questões e provas desde a sua criação passando por execução e correção.

Os lados do cliente e do servidor se comunicam através do protocolo HTTP.

3.5 Planejamento do fluxo de execução da correção automática

Primeiramente é necessário fazer um tratamento inicial na imagem do cartão resposta para se obter um formato padrão. O tratamento tem como objetivo normalizar a perspectiva da imagem de modo que ela aparente ter sido fotografada paralelamente à câmera. Além disso, deve-se ajustar a sua rotação. Para esses ajustes, podem ser utilizadas as referências de posição presentes no cartão. Na Figura 8 pode-se observar as referências impressas no cartão circuladas em vermelho.

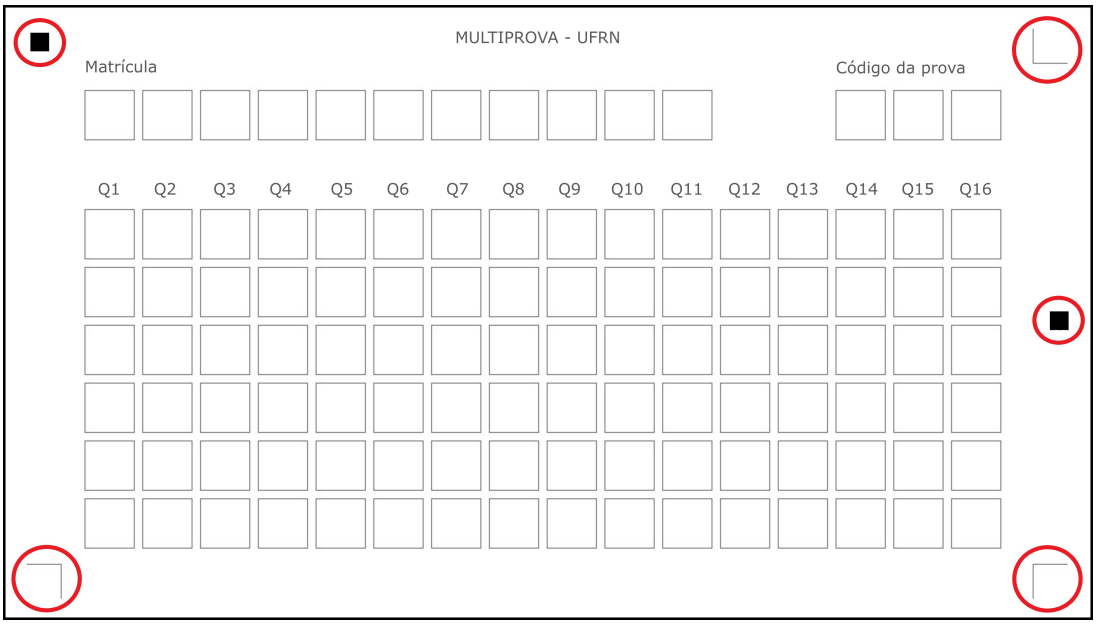


Figura 8 – Destaque das referências de posição do cartão resposta.

Fonte: De autoria própria.

Depois da obtenção da imagem do cartão, agora formatada, são feitos cortes já predefinidos em todas as 110 caixas de texto, onde 11 são correspondentes à matrícula do aluno, 3 ao código da prova e 96 são distribuídas igualmente entre as 16 questões (6 para cada questão). Essas imagens são extraídas e agrupadas para análise posterior.

Tendo em vista que o problema apresentado tem como base fundamental o reconhecimento de caracteres escritos a mão, a utilização de Inteligência Artificial - mais precisamente o reconhecimento de imagens por redes neurais artificiais - é a escolha mais adequada a fim de se desenvolver uma solução para um problema desta natureza e complexidade.

Para isso, é necessário o desenvolvimento de CNN's que consigam identificar, com boa taxa de precisão, letras e números. O método de aprendizagem supervisionada - onde já temos de antemão quais classes queremos reconhecer - se adapta melhor ao caso em questão.

Com as imagens recortadas e as CNN's devidamente treinadas, só restaria a etapa de classificação dessas imagens e o envio dos resultados para o sistema do Multiprova.

3.6 Organização dos dados

O processo de treinamento das CNN's se inicia com a compilação de uma grande quantidade de dados. São necessárias amostras manuscritas de todas as letras maiúsculas e números de dígito único (0 a 9).

3.6.1 Datasets

Para os dois casos (números e letras) foram solicitados que alguns alunos da Escola de Ciências e Tecnologia (ECT), da UFRN, completassem um cartão com exemplos dos caracteres requeridos. Um exemplo desse cartão preenchido pode ser observado a seguir na Figura 9.

Figura 9 – Cartão com amostras feitas pelos alunos.

Fonte: De autoria própria.

Ainda que cada aluno tenha feito uma quantidade igual de caracteres para cada classe, a quantidade de imagens por classe não se manteve igual. Isso aconteceu em razão de uma filtragem feita nas amostras dos caracteres. Se a figura não apresentava uma qualidade boa, ela era descartada, por isso algumas classes possuem mais imagens do que outras.

Para todos os *datasets* (letras, dígitos e V ou F) as quantidades de amostras das classes estavam desbalanceadas e para um melhor treinamento das redes neurais as

quantidades de amostras por classe devem ser balanceadas.

É possível observar o desbalanceamento das classes nas Figuras 10, 12 e 11.

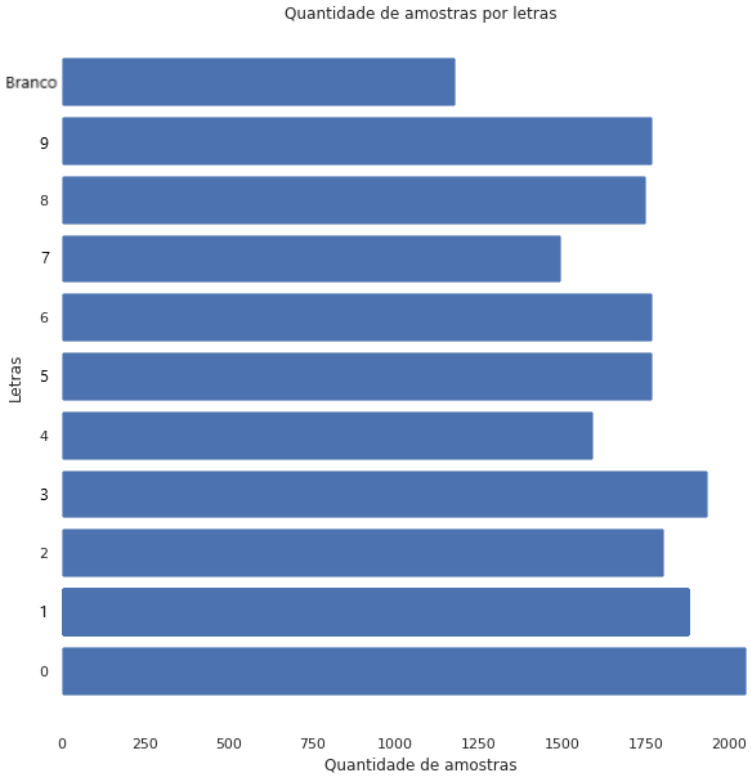


Figura 10 – Número de amostras por classe (dígitos).

Fonte: De autoria própria.

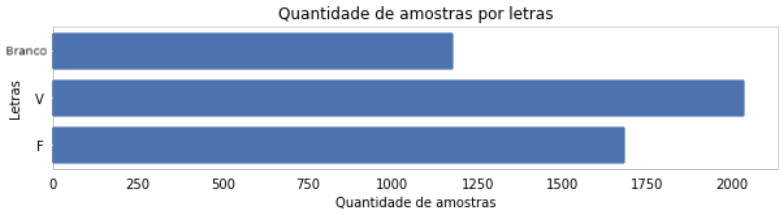


Figura 11 – Número de amostras por classe (V ou F).

Fonte: De autoria própria.

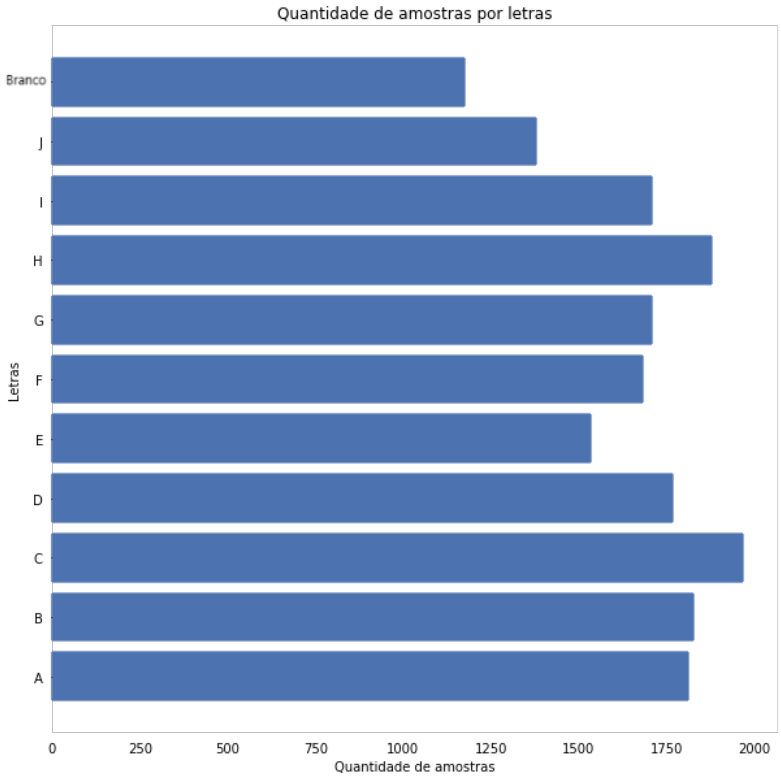


Figura 12 – Número de amostras por classe (letras).

Fonte: De autoria própria.

3.6.2 Balanceamento de dados

Para realizar a distribuição saudável na quantidade de amostras por classe, é recomendável aplicar técnicas de balanceamento de dados a fim de melhorar a qualidade do *dataset*.

Duas técnicas podem ser aplicadas com esse fim, *oversampling* e *undersampling*.

O *oversampling* é útil em situações onde existe uma classe com uma quantidade de dados muito baixa e é necessário equiparar o número de amostras dela com as de outras classes. Na prática, se deve selecionar as imagens da classe com baixo número de amostras e realizar processos de modificação na imagem a fim de gerar imagens novas, levemente diferentes, mas que ainda pertençam aquela classe.

Exemplos de tipos de *oversampling* são: *zoom in*, *zoom out*, espelhamento vertical e/ou horizontal, deslocamento e rotação da imagem. Para o melhor uso dessa técnica e a fim de não descaracterizar as imagens da classe, neste caso, os tipos de *oversampling* mais adequados são, *zoom in*, *zoom out*, translação e rotação da imagem.

Já o *undersampling* ocorre quando uma certa classe tem uma quantidade de dados muito superior às outras. Para resolver esse problema, o modo mais simples é remover

aleatoriamente imagens daquela classe até que ela atinga o número desejado de amostras.

Neste trabalho, foram aplicadas técnicas de *oversampling*, já que não existe nenhuma classe com uma quantidade de amostras muito maior que as outras.

3.7 Arquitetura das CNN's

As CNN's utilizarão camadas convolucionais, de *pooling* máximo, *flatten* e totalmente conectadas na sua construção. Para a criação da estrutura das CNN's, foi utilizado um modelo base, que irá se adaptar à medida que os testes empíricos sejam realizados.

Estrutura base:

- Camada convolucional com 32 filtros (64 para as camadas além da primeira) e *kernel* (3 x 3) com função de ativação relu e dados de entrada no formato (32 x 32 x 3) (imagem 32 x 32 *pixels* com 3 canais de cor);
- Camada de *pooling* máximo com filtro 2 x 2;
- Camada *flatten*;

O *optimizer* utilizado foi o Adam (ou *Adaptive Moment Estimation Algorithm*), proposto em 2014 por Diederik P. Kingma e Jimmy Ba. De acordo com seus criadores, esse *optimizer* é computacionalmente eficiente, requer pouca memória, é invariante ao redimensionamento diagonal dos gradientes e é adequado para problemas que são grandes em dados e/ou parâmetros (KINGMA; BA, 2014).

A única alteração ocorrerá na camada totalmente conectada onde esta é criada com função de ativação Softmax e X neurônios em sua saída - onde X é a quantidade de classes. Para as CNN's de letras e dígitos têm-se 11 classes e para V ou F têm-se 3.

3.8 Treinamento das CNN's

O desenvolvimento computacional das CNN's foi realizado utilizando Python, TensorFlow e toda a sua extensa API para a facilitação do processo na plataforma do Google Colaboratory.

As especificações da máquina virtual provida pelo Google são:

- CPU: Intel(R) Xeon(R) @ 2.20GHz, 1 *core*;
- Memória RAM: 12,68 GB;

- Disco: 78,19 GB;
- GPU: Nvidia Tesla K80.

Os *datasets* foram construídos utilizando as seguintes quantidades, 70% das amostras, validação, 15% das amostras, e teste, 15% das amostras. Todas as imagens já foram obtidas formatadas para o tamanho da entrada da CNN (32 x 32 x 3) então não foi necessário um tratamento prévio nesse quesito. O treinamento ocorrerá até que a CNN não apresente uma melhora na precisão ou na perda dentro de uma margem de 0,1% (técnica conhecida como *early stopping*).

Com os *datasets* devidamente formatados e as CNN's estruturadas corretamente só resta realizar o treinamento e salvar os pesos das CNN's treinadas para uso posterior.

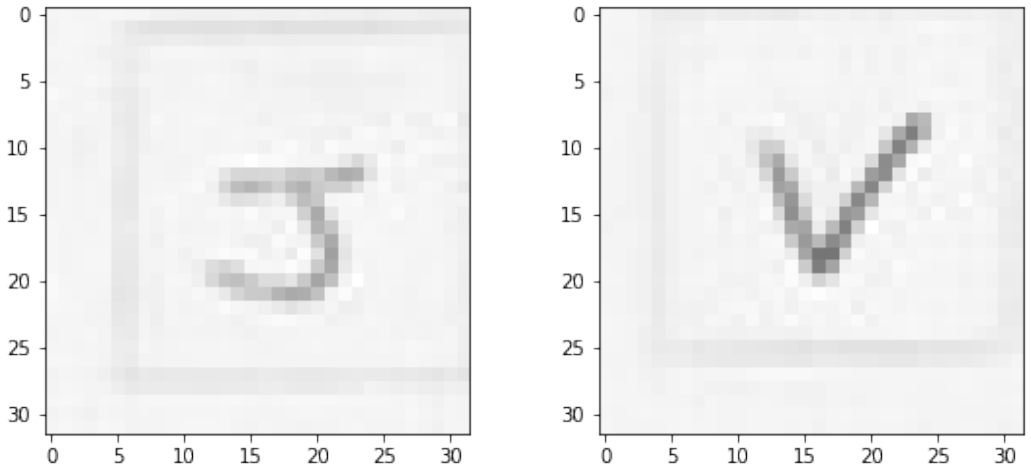
4 RESULTADOS E DISCUSSÃO

4.1 Pré-processamento das imagens

O pré-processamento se faz necessário em razão da necessidade de se reduzir a discrepância na quantidade de amostras. Gerar novas imagens utilizando transformações matemáticas sob as amostras base, é um processo efetivo para o aumento na diversidade dos dados presentes no *dataset*.

Conforme visto no Capítulo 3, Figura 12, é possível observar a grande diferença na quantidade de amostras por classe no *dataset* de letras.

As imagens tem dimensões 32 x 32 x 3 (altura x largura x quantidade de canais de cor). Alguns exemplos de amostras de imagens das classes J e V podem ser observados na Figura 13.



(a) Amostra de imagem do caractere J. (b) Amostra de imagem do caractere V.

Figura 13 – Letras base utilizadas no aumento de dados.

Fonte: De autoria própria.

Após o processo de aumento de dados, as novas amostras podem ser conferidas nas Figuras 14 e 15.

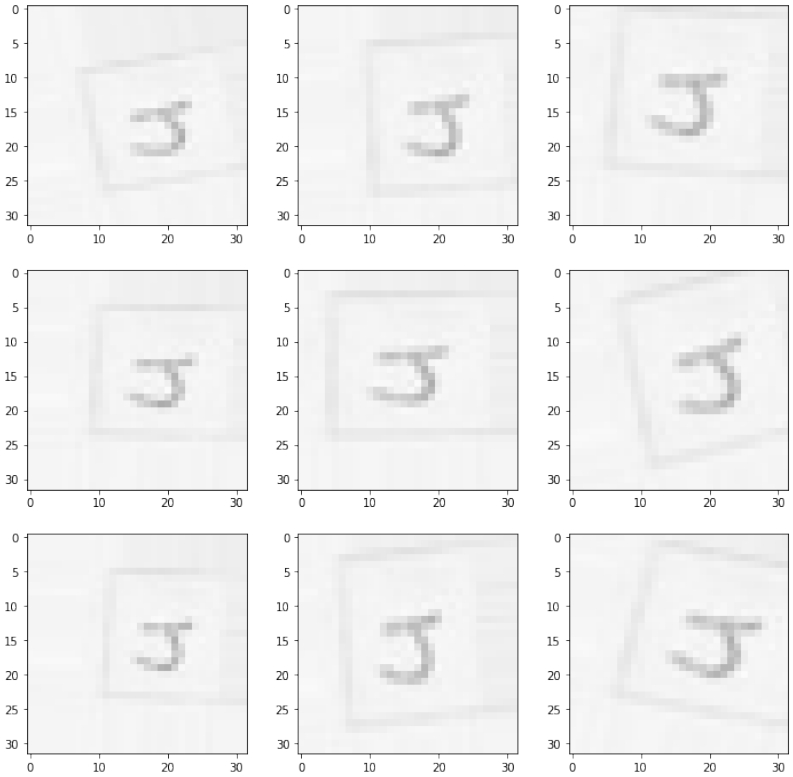


Figura 14 – Amostras de imagens do caractere J após o aumento de dados.

Fonte: De autoria própria.

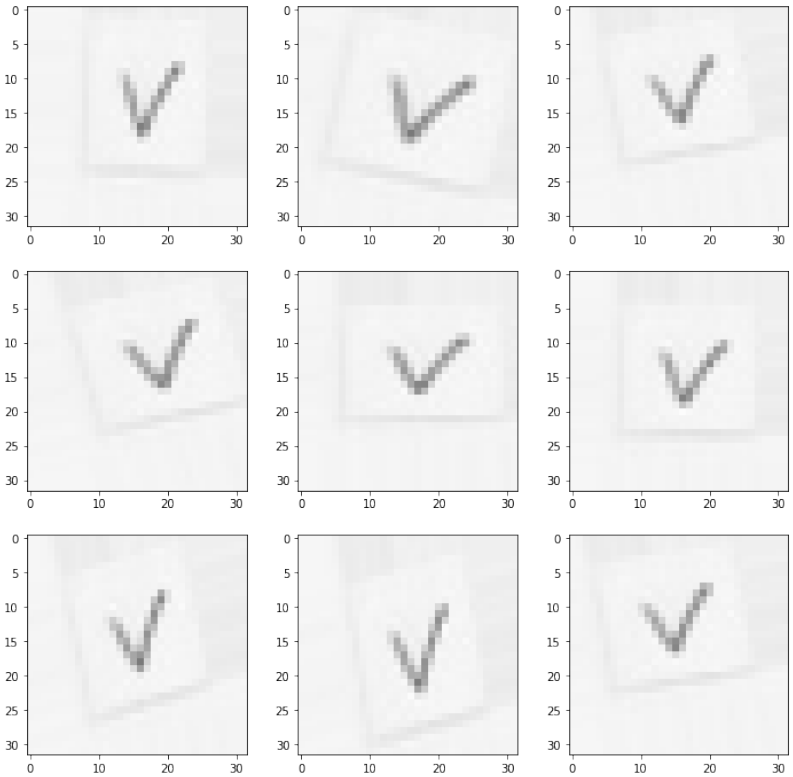


Figura 15 – Amostras de imagens do caractere V após o aumento de dados.

Fonte: De autoria própria.

Na Figura 13 é possível observar um exemplo das imagens bases utilizadas para a geração de novas amostras. Já na Figura 14 e Figura 15 foram compiladas 9 imagens do resultado de cada classe do processo de aumento de dados. São perceptíveis os efeitos de *zoom in*, *zoom out*, rotação e deslocamento se comparadas com as amostras originais.

Por fim, após o balanceamento da quantidade de amostras, juntou-se todas as imagens por classe em um único *dataset* e o resultado pode ser visto na Figura 16.

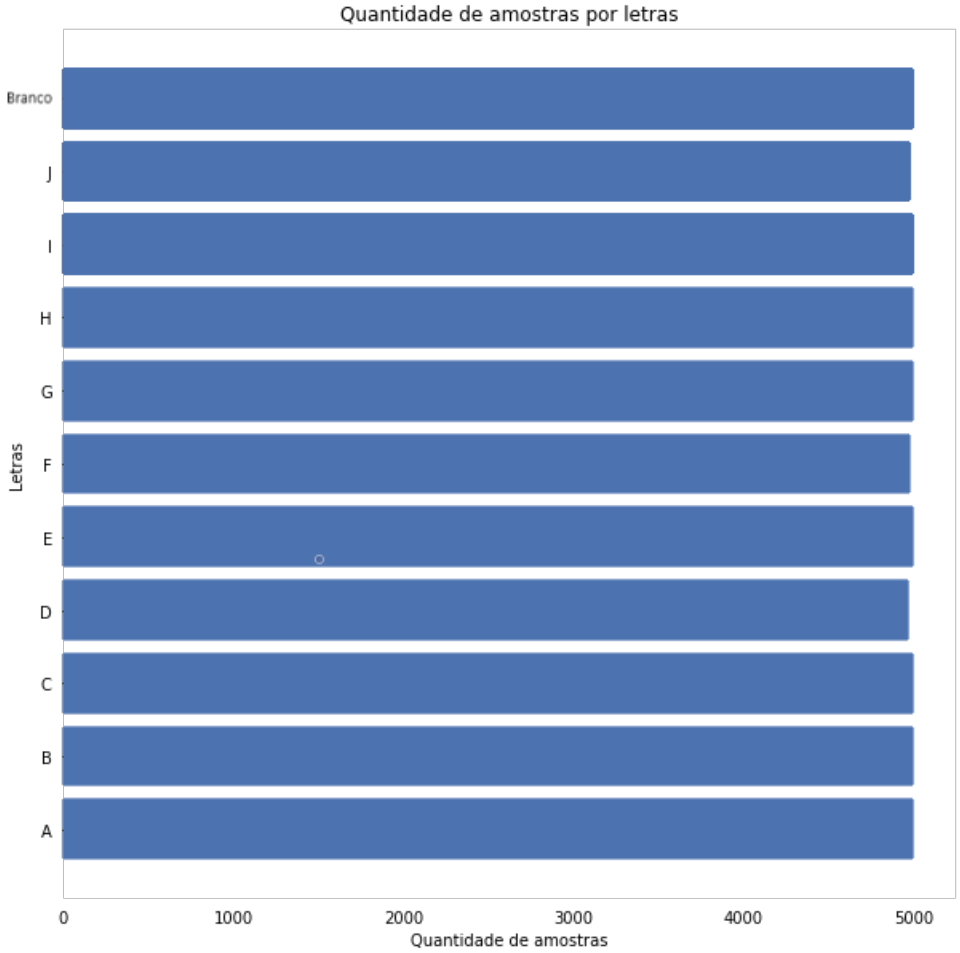


Figura 16 – Número de amostras por classe após o aumento de dados (letras).

Fonte: De autoria própria.

Inicialmente, para o *dataset* de letras, como é possível ver no Capítulo 3, Figura 12, a quantidade de amostras por classe é desigual, variando de aproximadamente 1100 (na classe *Branco*) para 2000 amostras (na classe *C*). Essa variação representa uma diferença de aproximadamente 81% mais imagens na classe *C* do que na classe *Branco*.

Já para o *dataset* de números (Capítulo 3, Figura 10), a diferença máxima na quantidade de amostras por classe ficou em torno de 90%, na classe *Branco* - 1100 amostras - em comparação com a classe *0* - 2100 amostras - e diminuiu para 20% nas classe *7* - 5800 amostras em comparação com outras classes (0, 1, 2, 3, 5, 6, 8 e 9 com 7000 amostras)

(Figura 17).

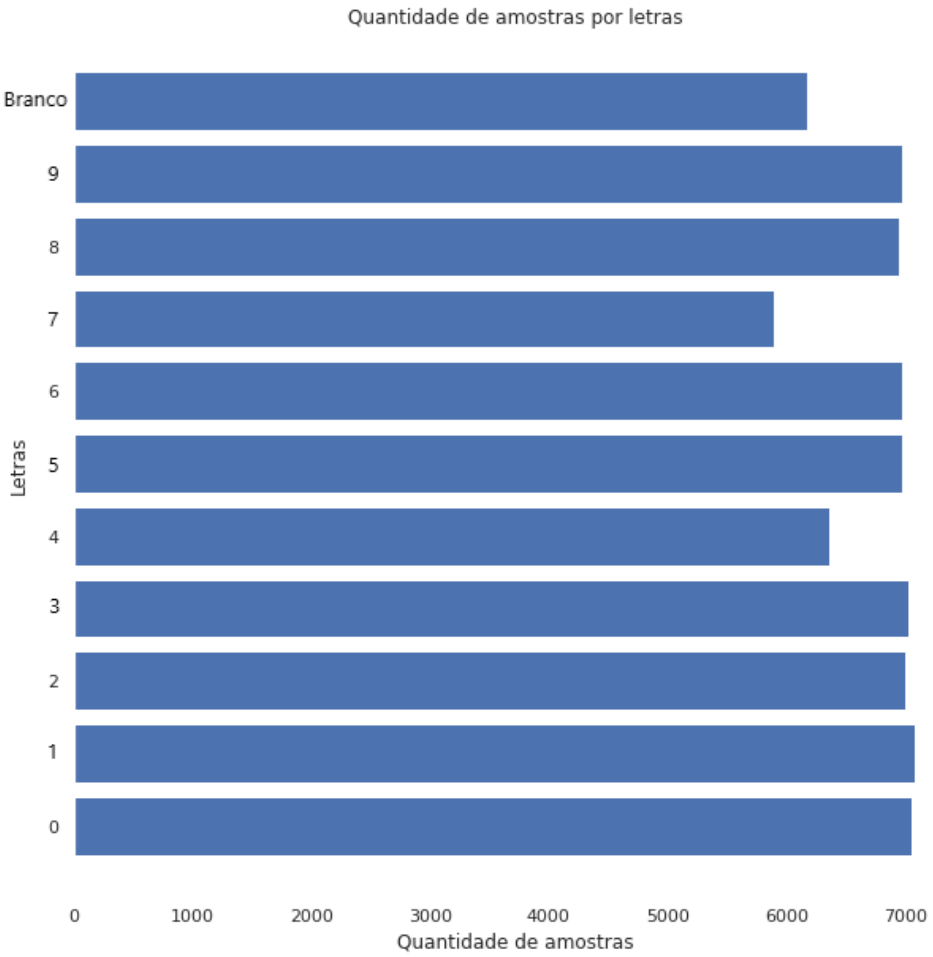


Figura 17 – Número de amostras por classe após o aumento de dados (dígitos).

Fonte: De autoria própria.

Essas divergências nas quantidades de imagens do *dataset*, podem implicar em problemas no treinamento da rede e possíveis erros de classificação, levando a CNN a identificar muito bem as classes com uma grande quantidade de amostras, mas falhar em identificar classes com baixas quantidades de amostras.

Após todo o processo de balanceamento das classes (Figuras 16 e 17), a diferença percentual desses números (agora próximo de no máximo 20%) se mostra melhor para eliminar as diferenças nas quantidade das amostras e por consequência, minimizar possíveis erros.

4.2 Treinamento das CNN's

De modo geral, os treinamentos das CNN's ocorreram adequadamente, permitindo resultados com precisão acima dos 98%, e perda abaixo dos 6% na sua melhor configuração.

Para o reconhecimento das letras e dígitos, estruturas com 4 camadas convolucionais, seguidas de *max pooling* se mostraram mais eficientes. Para o caso do reconhecimento de caracteres v ou f, estruturas com 3 camadas convolucionais, seguidas de *max pooling*, apresentaram os melhores resultados.

4.2.1 Dígitos

Com os dígitos são apresentadas classes com diferenças estruturais na sua escrita, por exemplo, o 1 pode ser escrito com somente uma reta vertical, já o 8 é escrito com dois círculos empilhados. Como cada classe é consideravelmente diferente uma da outra, espera-se que a CNN não apresente dificuldades de adaptação às classes durante o treinamento. Na Figura 17, pode-se observar a quantidade de amostras por classe, utilizadas no treinamento da CNN especialista em dígitos. A quantidade de imagens por classe, se manteve entre 5700 e 7000, com uma média de aproximadamente 6700 amostras por classe.

A seguir serão apresentadas as estruturas utilizadas no treinamento da CNN de dígitos bem como suas configurações e resultados gerados.

- Estrutura 1

Com uma camada convolucional, uma camada de *pooling* máximo, a camada *flatten* e a camada totalmente conectada, a Figura 18 apresenta a estrutura utilizada no para o reconhecimento dos dígitos de 0-1, além da classe *Branco*.

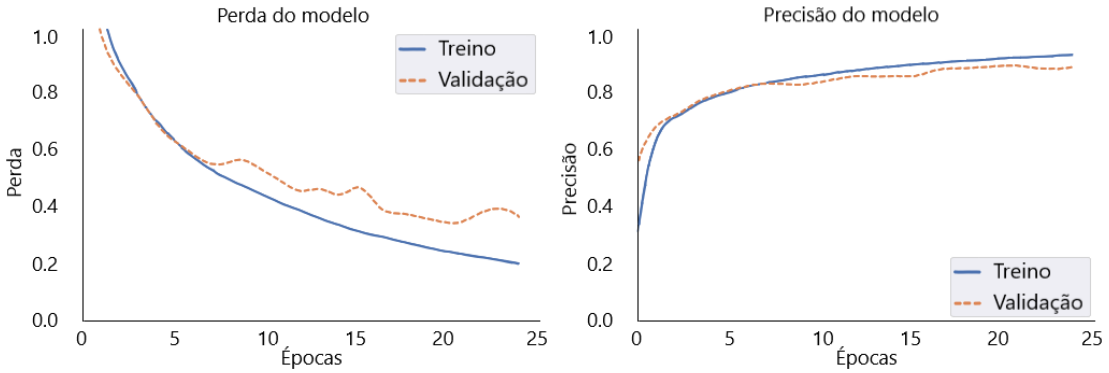
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 flatten (Flatten)           (None, 7200)              0

 dense (Dense)               (None, 64)                460864

 dense_1 (Dense)             (None, 11)                715

=================================================================
Total params: 462,475
Trainable params: 462,475
Non-trainable params: 0
_________________________________________________________________
```

Figura 18 – Estrutura 1 da CNN de dígitos.

Fonte: De autoria própria.

É possível observar que as curvas de erro na Figura 19a se mantém consideravelmente distantes, isso indica que a CNN não se adaptou suficientemente bem para os casos fora do

seu conjunto de treinamento. A curva de perda no conjunto de validação, ainda que tenha evoluído para um erro mais baixo ao longo do tempo, apresentou um perfil oscilatório, enquanto a curva de erro do conjunto de treino evoluiu suavemente, o que mostra que a rede neural teve dificuldades em se adaptar a um conjunto que não foi previamente apresentado a ela. Já as curvas de precisão na Figura 19b se mostram mais suaves, sem muitas variações e com uma discrepância menor entre os dados de treinamento e validação.



(a) Gráfico de evolução do fator de perda durante o treinamento.

(b) Gráfico de evolução do fator de precisão durante o treinamento.

Figura 19 – Gráficos de perda e precisão (dígitos).

Fonte: De autoria própria.

A matriz de confusão apresentada na Figura 20, confirma os resultados apresentados nos gráficos da Figura 19, com apenas 2 classes com valores acimas de 95% observa-se que, as classes apresentam percentuais de erros que chegam até os 12%, um valor alto, com exceção da classe *Branco* em que a rede neural apresentou desempenho excelente no reconhecimento, 100% de acerto e nenhum erro.

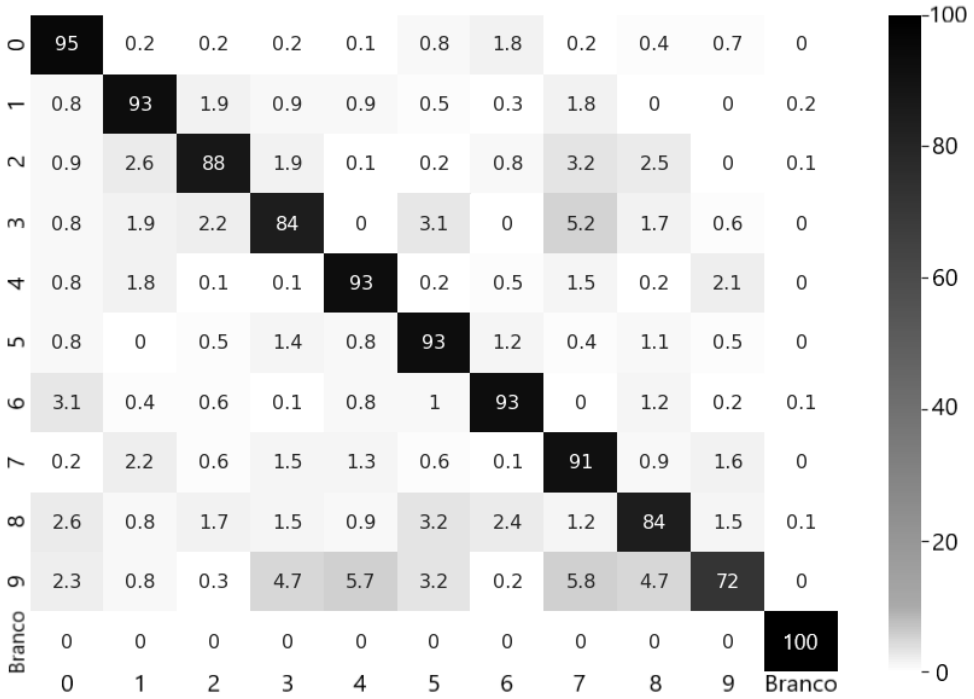


Figura 20 – Matriz de confusão (dígitos).

Fonte: De autoria própria.

Com os resultados da primeira estrutura vê-se que com apenas uma camada convolucional e de *max pooling* a CNN tem dificuldade em reconhecer os detalhes de cada classe, com destaque para as classes 2, 3 e 8 que não atingiram os 90% de precisão e para a classe 9 que atingiu somente 72% de precisão com uma porcentagem de erro de 5,8% - um valor alto - sendo confundida com a classe 7. Para fim de comparação com as outras estruturas a porcentagem de acerto geral foi de 89,41% com a pior taxa de acerto por classe de 72%.

- Estrutura 2

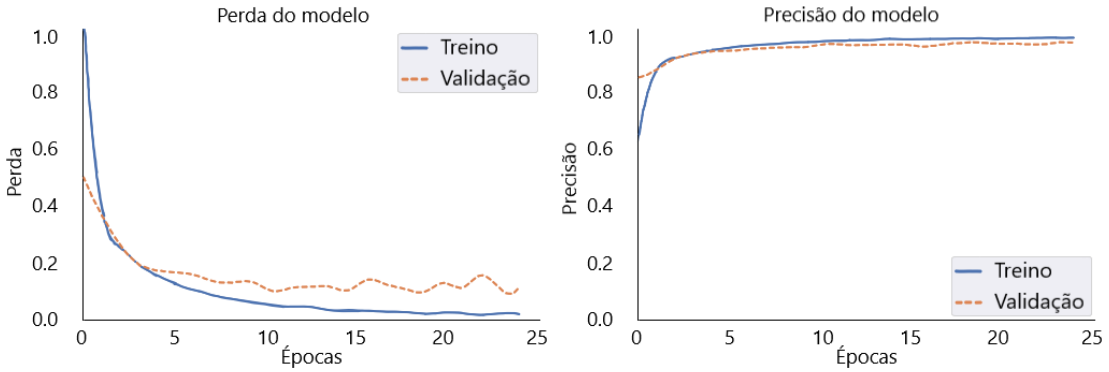
Aumentando uma camada convolucional e uma *max pooling* em relação à estrutura 1, a estrutura 2 (Figura 21) apresentou melhores resultados. Com uma taxa de acerto geral de 95,87%, e a pior classe com 90%, a melhoria é perceptível tanto na matriz de confusão na Figura 23 quanto nos gráficos de treinamento da Figura 22.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_1 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten_1 (Flatten)         (None, 2304)              0

 dense_2 (Dense)             (None, 64)                147520

 dense_3 (Dense)             (None, 11)                715

=================================================================
Total params: 167,627
Trainable params: 167,627
Non-trainable params: 0
_________________________________________________________________
```

Figura 21 – Estrutura 2 da CNN de dígitos.

Fonte: De autoria própria.

As curvas de perda e precisão da Figura 22, mostram uma melhora, comparadas com as curvas da estrutura 1 (Figura 19), nos seus valores de convergência (a precisão mais próxima do 100% e o erro mais próximo de 0%), ainda que a curva de perda tenha mantido a discrepância entre os conjuntos de dados de treinamento e validação.



(a) Gráfico de evolução do fator de perda durante o treinamento.

(b) Gráfico de evolução do fator de precisão durante o treinamento.

Figura 22 – Gráficos de perda e precisão (dígitos).

Fonte: De autoria própria.

Já na matriz de confusão da Figura 23, vê-se menos erros, com o pior deles de 4,4% na linha de classe 8 e coluna 9. A única classe abaixo dos 95% de precisão foi a classe 7 que atingiu somente 90% de acerto.

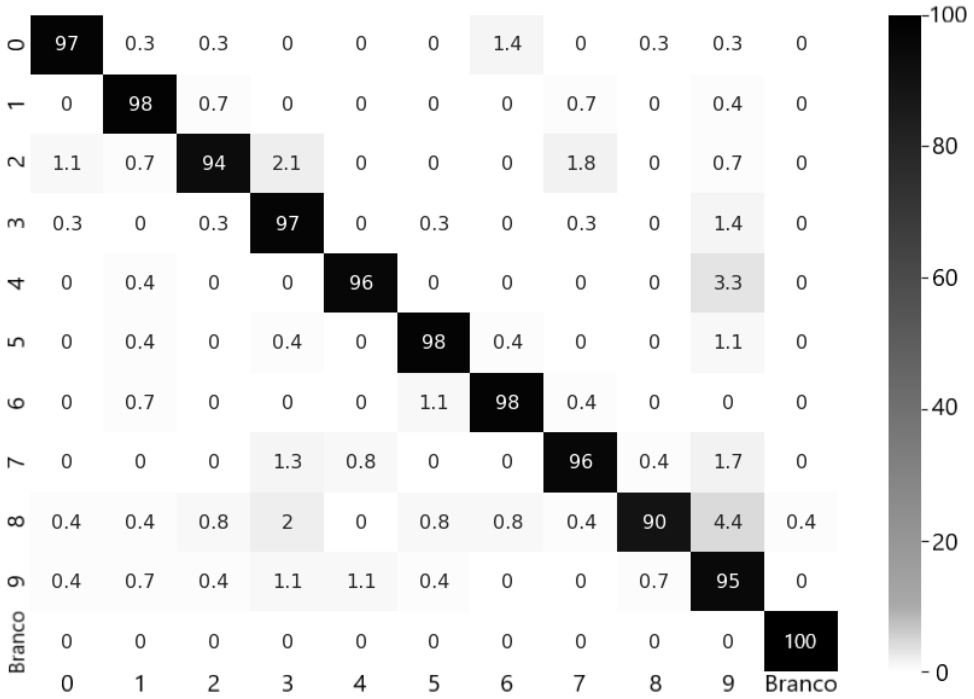


Figura 23 – Matriz de confusão (dígitos).

Fonte: De autoria própria.

- Estrutura 3

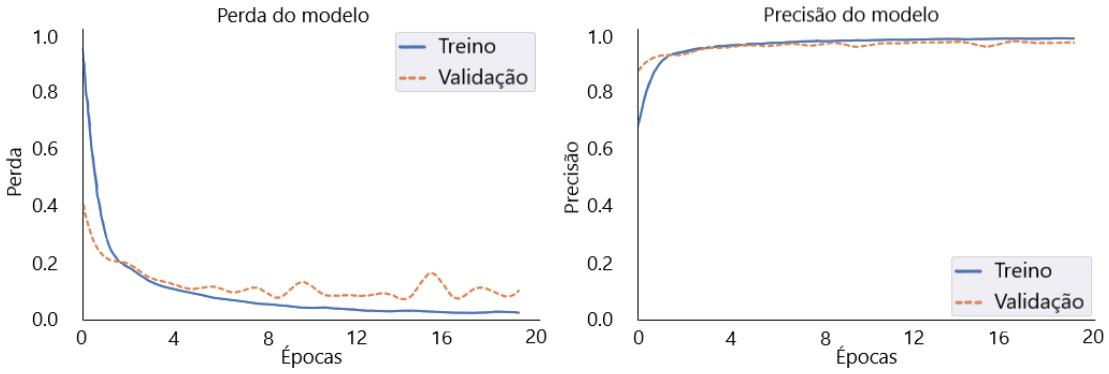
Para a estrutura 3 (Figura 24) é perceptível mais uma melhora - ainda que não tenha sido tão grande quanto da estrutura 1 para a estrutura 2. Com uma média geral de acerto de 97,8% e a pior classe com 95% de acerto, a terceira estrutura conseguiu utilizar suas camada convolucional e de *max pooling* extras para refinar os resultados obtidos.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_6 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_7 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_7 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_8 (Conv2D)           (None, 4, 4, 64)          36928

 max_pooling2d_8 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 flatten_3 (Flatten)         (None, 256)               0

 dense_6 (Dense)             (None, 64)                16448

 dense_7 (Dense)             (None, 11)                715

=================================================================
Total params: 73,483
Trainable params: 73,483
Non-trainable params: 0
_________________________________________________________________
```

Figura 24 – Estrutura 3 da CNN de dígitos.

Fonte: De autoria própria.

Os gráficos da Figura 25 mostram essa melhoria. Com um estreitamento das diferenças entre os conjuntos de treinamento e validação assim como a melhora na convergência das curvas para valores mais adequados (aproximadamente 100% para precisão e 0% para perda).



(a) Gráfico de evolução do fator de perda durante o treinamento.

(b) Gráfico de evolução do fator de precisão durante o treinamento.

Figura 25 – Gráficos de perda e precisão (dígitos).

Fonte: De autoria própria.

A matriz de confusão também apresenta essa diferença na Figura 26. Com um erro máximo de 3,1% a matriz apresenta poucos erros acima de 1% e reforça a boa taxa de

acerto das classes na diagonal principal, com destaque para as classes *0* e *4* que atingiram 99% de acerto.

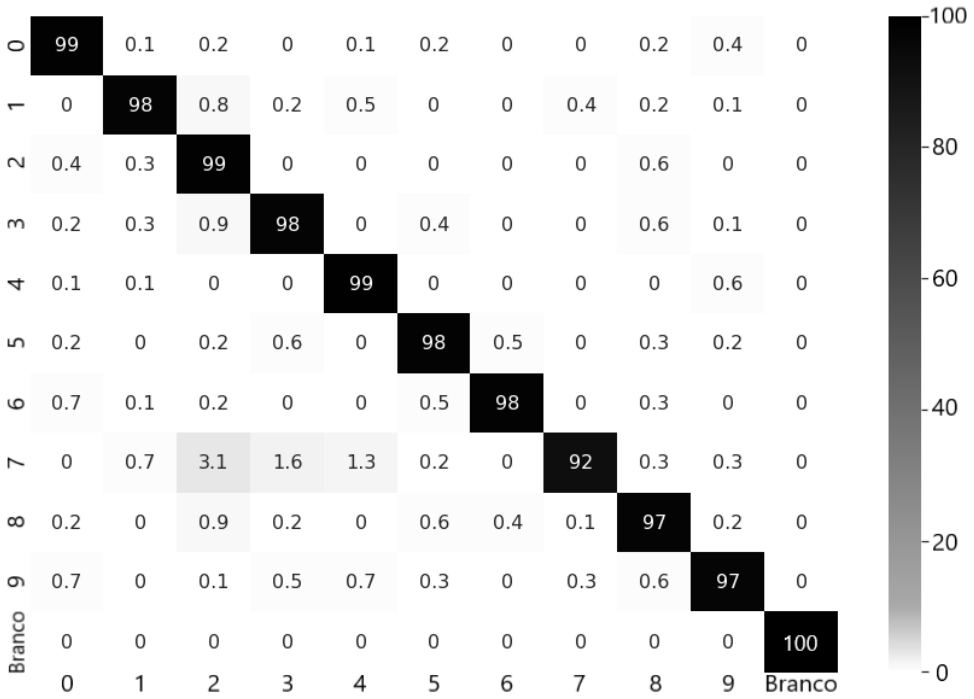


Figura 26 – Matriz de confusão (dígitos).

Fonte: De autoria própria.

- Estutura 4

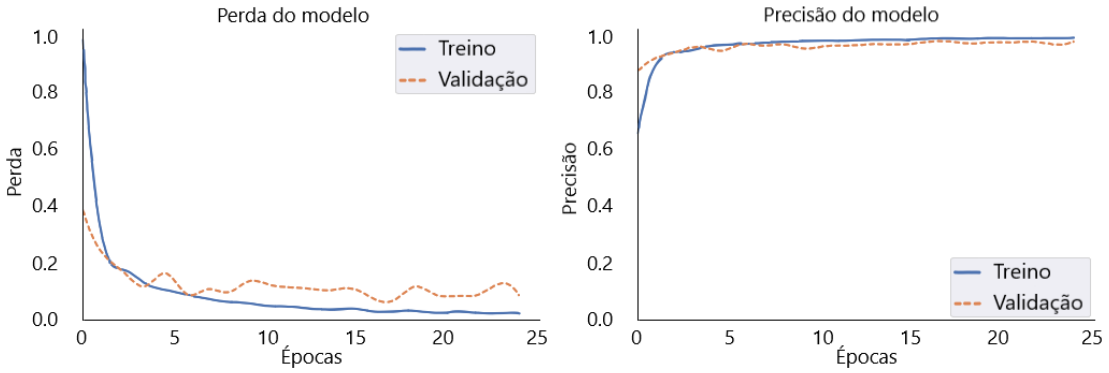
Por fim, a estrutura 4 (Figura 27) indica mais uma melhora pequena em relação à estrutura 3. Com uma média geral de acerto de 98,84% e a pior classe com 98% de acerto, a quarta estrutura apresentou o melhor resultado entre as 4 configurações analisadas.

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_9 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_10 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_10 (MaxPoolin  (None, 6, 6, 64)         0
 g2D)

 conv2d_11 (Conv2D)          (None, 4, 4, 64)          36928

 max_pooling2d_11 (MaxPoolin  (None, 2, 2, 64)         0
 g2D)

 conv2d_12 (Conv2D)          (None, 1, 1, 64)          16448

 max_pooling2d_12 (MaxPoolin  (None, 1, 1, 64)         0
 g2D)

 flatten_4 (Flatten)         (None, 64)                0

 dense_8 (Dense)             (None, 64)                4160

 dense_9 (Dense)             (None, 11)                715

=================================================================
Total params: 77,643
Trainable params: 77,643
Non-trainable params: 0
_________________________________________________________________
```

Figura 27 – Estrutura 4 da CNN de dígitos.

Fonte: De autoria própria.

Os gráficos de perda e precisão (Figura 28) não apontam grandes diferenças. As oscilações na curva de validação do gráfico de perda (Figura 28a) ainda se mantém presente. Comparando os gráficos com os da estrutura anterior o incremento de qualidade não é tão perceptível.



(a) Gráfico de evolução do fator de perda durante o treinamento.

(b) Gráfico de evolução do fator de precisão durante o treinamento.

Figura 28 – Gráficos de perda e precisão (dígitos).

Fonte: De autoria própria.

Já na matriz de confusão - Figura 29 - é possível verificar a melhora da qualidade da CNN. Com um erro máximo de 0,9% a matriz apresenta poucos erros acima de 0,5% e reforça a ótima taxa de acerto das classes na diagonal principal (que não foi menor que 98%), com destaque para a classe *6* que atingiu os 100% de acerto.

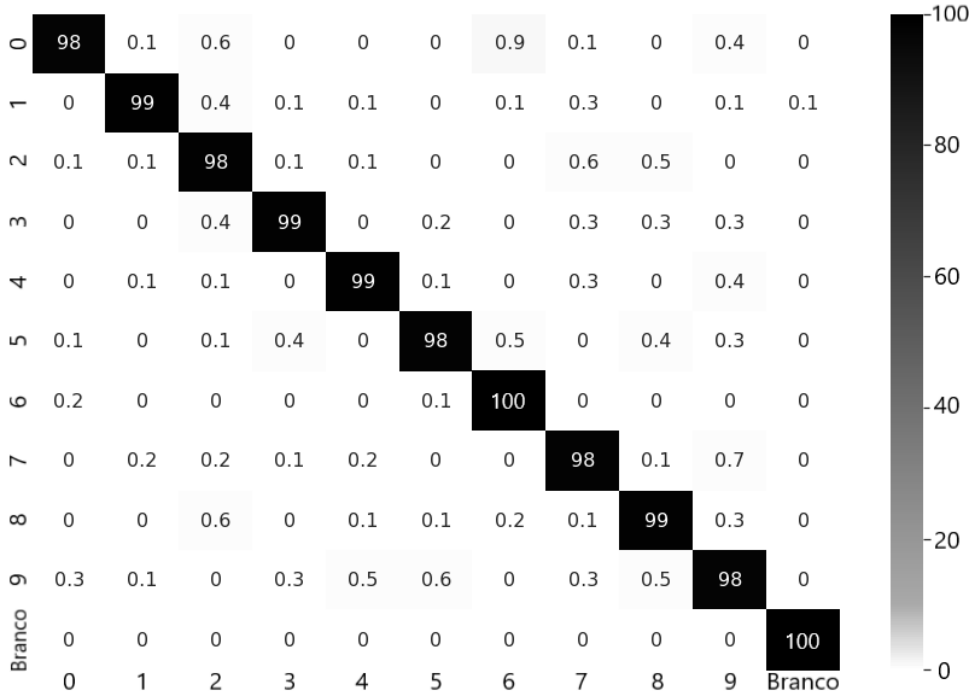


Figura 29 – Matriz de confusão (dígitos).

Fonte: De autoria própria.

Para a CNN que classifica os dígitos foram analisadas 4 estruturas de CNN, onde a que mais se mostrou eficiente foi a quarta (com mais camadas convolucionais e de *pooling* máximo). A quantidade de épocas necessária para habilitar o *early stopping* e parar o treinamento mais cedo se manteve em 24 para as estruturas 1, 2 e 4 enquanto a estrutura 3 convergiu mais rapidamente com somente 19 épocas.

A Tabela 2 sumariza o processo de evolução da precisão e perda da CNN de dígitos.

Estrutura	**Precisão**	**Perda**
Estrutura 1	89,41%	0,351
Estrutura 2	95,87%	0,115
Estrutura 3	97,80%	0,098
Estrutura 4	98,84%	0,064

Tabela 2 – Evolução da CNN de dígitos através das estruturas

4.2.2 Letras

Já com as letras as diferenças estruturais não são tão discrepantes como nos dígitos. Algumas classes compartilham de similaridades na escrita, por exemplo, as classes E e F diferem somente por uma linha reta horizontal na parte inferior e as classes I e J tem como similaridade uma reta vertical e uma reta horizontal na parte de baixo, ainda que a classe I possa ter outra reta horizontal na parte superior grande parte da estrutura dos caracteres são iguais. Em razão disso se espera uma piora na taxa de acerto final em comparação com a CNN de dígitos.

Na Figura 16 pode-se observar a quantidade de amostras por classe utilizadas no treinamento da CNN de letras. O número de imagens por classe se manteve consistentemente em 5000, totalizando aproximadamente 55000 figuras.

- Estrutura 1

Mantendo a mesma configuração da estrutura 1 utilizada no treinamento dos dígitos com uma camada convolucional, uma camada de *pooling* máximo, a camada *flatten* e a camada totalmente conectada, a Figura 30 apresenta a estrutura utilizada no para o reconhecimento das letras de A-J, além da classe *Branco*.

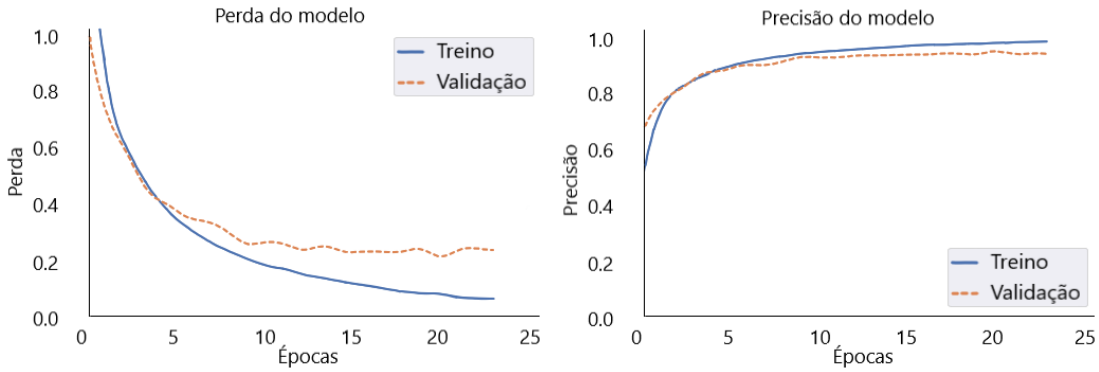
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 flatten (Flatten)           (None, 7200)              0

 dense (Dense)               (None, 64)                460864

 dense_1 (Dense)             (None, 11)                715

=================================================================
Total params: 462,475
Trainable params: 462,475
Non-trainable params: 0
_________________________________________________________________
```

Figura 30 – Estrutura 1 da CNN de letras.

Fonte: De autoria própria.

As curvas de perda da Figura 31a, assim como na estrutura 1 dos dígitos (Figura 19a), apresentam uma distância considerável para os valores de perda finais, indicando dificuldade na adaptação para os dados de validação. No entanto, ao contrário do observado na estrutura 1 dos dígitos, a estrutura das letras apresentou uma convergência melhor para 0 com um

perfil mais suave se sem oscilações consideráveis. Já a Figura 31b apresentou uma boa taxa de precisão (próxima de 1) assim como na Figura 19b.



(a) Gráfico de evolução do fator de perda durante o treinamento.

(b) Gráfico de evolução do fator de precisão durante o treinamento.

Figura 31 – Gráficos de perda e precisão (letras).

Fonte: De autoria própria.

A matriz de confusão - Figura 32 - aponta resultados que confirmam os gráficos vistos anteriormente. Com a classe *Branco*, assim como todo o treinamento da CNN de dígitos, apresentando 100% de precisão, têm-se a pior classe com 90% de acerto (classe E) e as melhores (com exceção da classe *Branco*) com 95% de acerto (classes F, G e H). Com uma precisão média de 93,36% os resultados são bons para a estrutura mais básica.

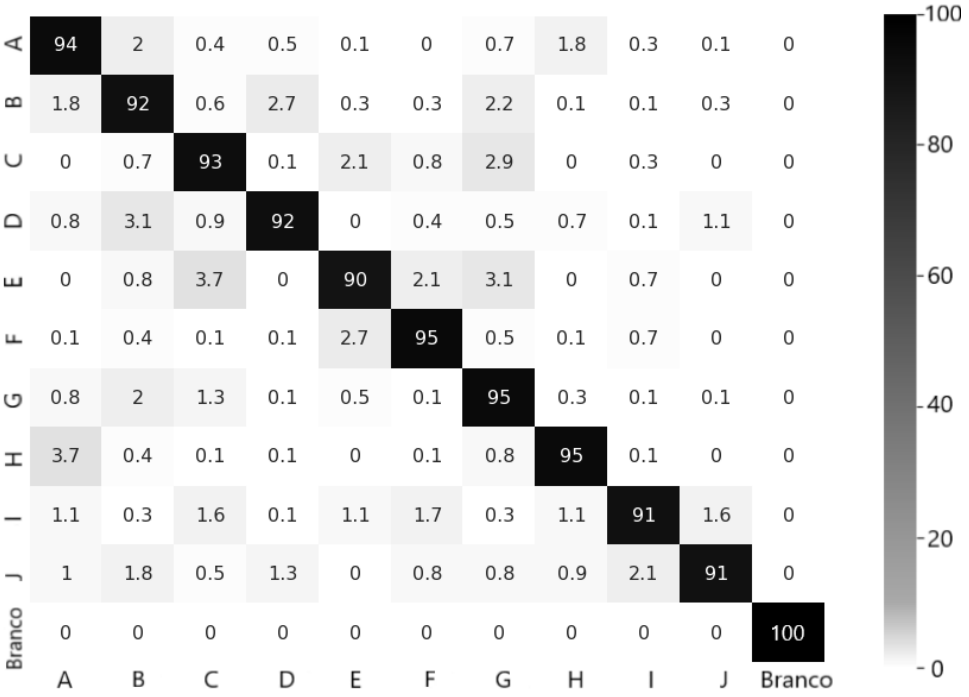


Figura 32 – Matriz de confusão (letras).

Fonte: De autoria própria.

- Estrutura 2

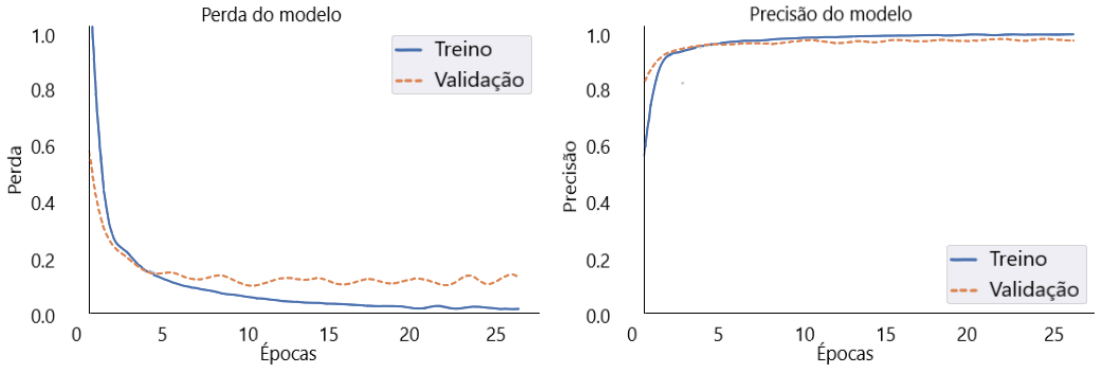
Aumentando uma camada convolucional e uma *max pooling* em relação à estrutura 1, a estrutura 2 (Figura 33) mostra melhores resultados. A melhora é perceptível nas Figuras 34 e 35, com uma taxa de acerto de 97,52% - 4,16% superior à estrutura 1.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_1 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten_1 (Flatten)         (None, 2304)              0

 dense_2 (Dense)             (None, 64)                147520

 dense_3 (Dense)             (None, 11)                715

=================================================================
Total params: 167,627
Trainable params: 167,627
Non-trainable params: 0
_________________________________________________________________
```

Figura 33 – Estrutura 2 da CNN de letras.

Fonte: De autoria própria.

Já a Figura 34 mostra as melhorias conseguidas através do aumento de camadas da estrutura. Com curvas de erro mais próximas do 0 e uma diminuição dos valores finais de erro em treino e validação (ainda que com várias pequenas oscilações na curva de validação) e a curva de precisão cada vez mais próxima de 1, os gráficos indicam um ganho considerável de qualidade com o incremento na quantidade de camadas.



(a) Gráfico de evolução do fator de perda durante o treinamento.

(b) Gráfico de evolução do fator de precisão durante o treinamento.

Figura 34 – Gráficos de perda e precisão (letras).

Fonte: De autoria própria.

Para a matriz de confusão da Figura 35, vê-se menos erros, um gráfico menos marmorizado com os tons de cinza indicando as confusões entre classes indicam que houve melhora no reconhecimento das imagens. Com a pior taxa de confusão sendo 2,7% entre as classes G e B, a pior classe com 94% de precisão (classe G) e as melhores - com exceção da classe *Branco* - chegando à 99% de acerto (classes I e J), o incremento de qualidade foi considerável.

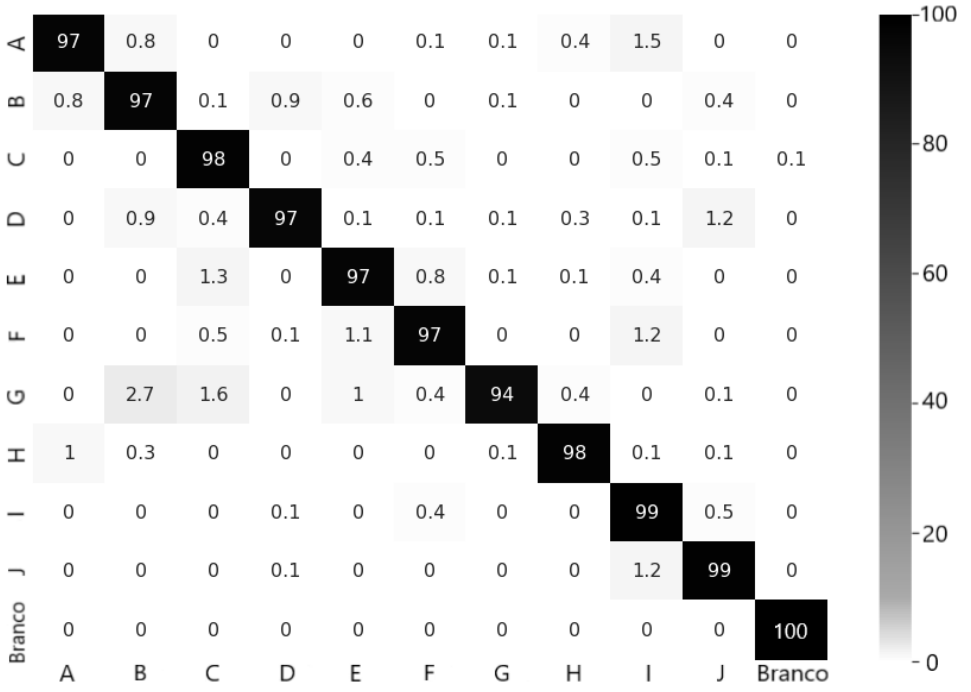


Figura 35 – Matriz de confusão (letras).

Fonte: De autoria própria.

- Estrutura 3

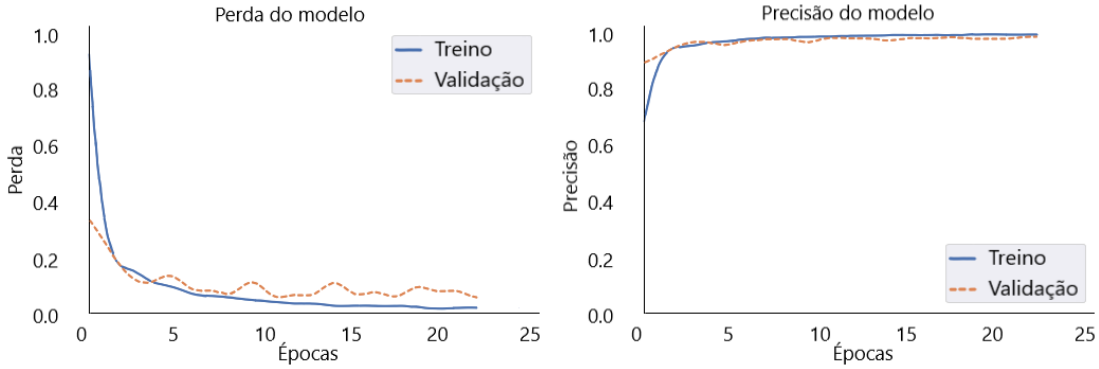
Com 98,1% de precisão - um aumento de 0,58% comparado com a estrutura 2 (uma melhora consideravelmente menor que a atingida entre as estruturas 1 e 2 de 4,16%) - a estrutura 3 fez bom uso das camadas convolucionais e de *pooling* máximo extras, ainda que o incremento de precisão não tenha sido grande.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_6 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_7 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_7 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_8 (Conv2D)           (None, 4, 4, 64)          36928

 max_pooling2d_8 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 conv2d_9 (Conv2D)           (None, 1, 1, 64)          16448

 max_pooling2d_9 (MaxPooling  (None, 1, 1, 64)         0
 2D)

 flatten_3 (Flatten)         (None, 64)                0

 dense_6 (Dense)             (None, 64)                4160

 dense_7 (Dense)             (None, 11)                715

=================================================================
Total params: 77,643
Trainable params: 77,643
Non-trainable params: 0
_________________________________________________________________
```

Figura 36 – Estrutura 3 da CNN de letras.

Fonte: De autoria própria.

O gráfico da Figura 37a mostram uma melhora na convergência da curva de validação. Ainda que as oscilações durante o treinamento se mantenham presentes, a curva de validação se aproximou mais do valor ideal e da curva de treino, o que indica uma melhor adaptação da CNN a dados que ela não foi apresentada. Já o gráfico da Figura 37b mantém a evolução percebida entre as estruturas 1 e 2, as curvas convergindo para valores mais próximos de 1 e a diminuição da diferença entre as curvas de treino e validação.



(a) Gráfico de evolução do fator de perda durante o treinamento.

(b) Gráfico de evolução do fator de precisão durante o treinamento.

Figura 37 – Gráficos de perda e precisão (letras).

Fonte: De autoria própria.

Já na matriz de confusão (Figura 38) a melhora geral é perceptível ainda que o pior ponto de confusão - nas classes A e B - seja pior que na estrutura 2 - aumentou de 2,7% para 3,2%. A precisão da pior classe se manteve nos 94% (na classe A) e as melhores taxas de precisão (com 99% de acurácia) foram as classes B, F, H e J.

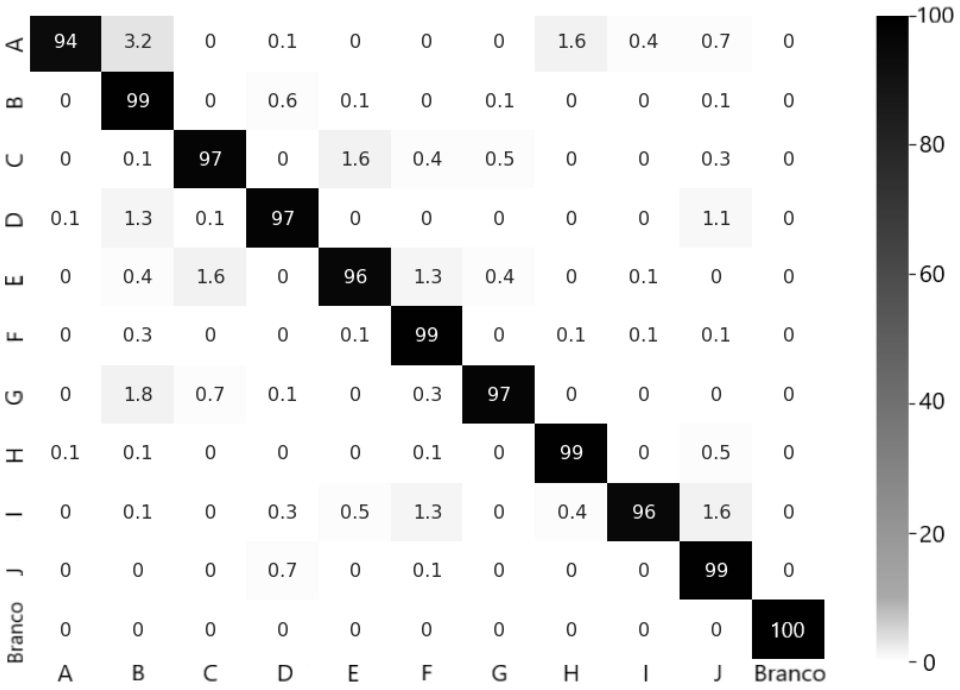


Figura 38 – Matriz de confusão (letras).

Fonte: De autoria própria.

- Estrutura 4

Por fim, a estrutura 4 (Figura 39) indica uma leve melhora em relação à estrutura 3. Com uma média geral de acerto de 98,38% (0,28% melhor que a estrutura anterior) e a

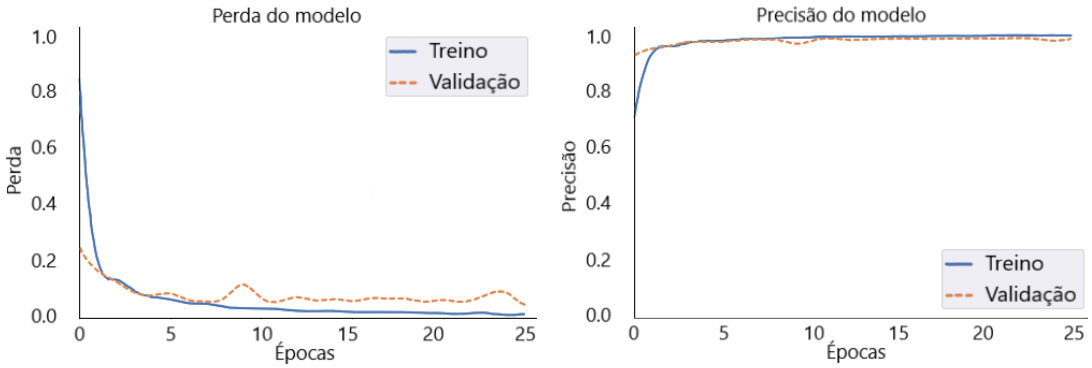
pior classe com 96% de acerto, a quarta estrutura apresentou o melhor resultado entre as 4 configurações analisadas.

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10 (Conv2D)          (None, 30, 30, 32)        896

 max_pooling2d_10 (MaxPoolin  (None, 15, 15, 32)       0
 g2D)

 conv2d_11 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_11 (MaxPoolin  (None, 6, 6, 64)         0
 g2D)

 conv2d_12 (Conv2D)          (None, 4, 4, 64)          36928

 max_pooling2d_12 (MaxPoolin  (None, 2, 2, 64)         0
 g2D)

 flatten_4 (Flatten)         (None, 256)               0

 dense_8 (Dense)             (None, 64)                16448

 dense_9 (Dense)             (None, 11)                715

=================================================================
Total params: 73,483
Trainable params: 73,483
Non-trainable params: 0
_________________________________________________________________
```

Figura 39 – Estrutura 4 da CNN de letras.

Fonte: De autoria própria.

Os gráficos de perda e precisão (Figura 40) não indicam grandes diferenças em razão do incremento ter sido pequeno (0,28% de melhora na precisão). As oscilações na curva de validação do gráfico de perda (Figura 40a) agora aparecem menos e o valor de convergência das curvas de treino e validação se aproximaram. Comparando os gráficos com os da estrutura 3 a melhora na qualidade não é tão perceptível.



(a) Gráfico de evolução do fator de perda durante o treinamento.

(b) Gráfico de evolução do fator de precisão durante o treinamento.

Figura 40 – Gráficos de perda e precisão (letras).

Fonte: De autoria própria.

Na matriz de confusão - Figura 41 - a melhora de qualidade da CNN, comparada com a estrutura 3, não é tão visível. Com um erro máximo de 2,3% a matriz apresenta poucos erros acima de 1% e reforça a boa taxa de acerto das classes na diagonal principal (que não foi menor que 96% na classe *D*).

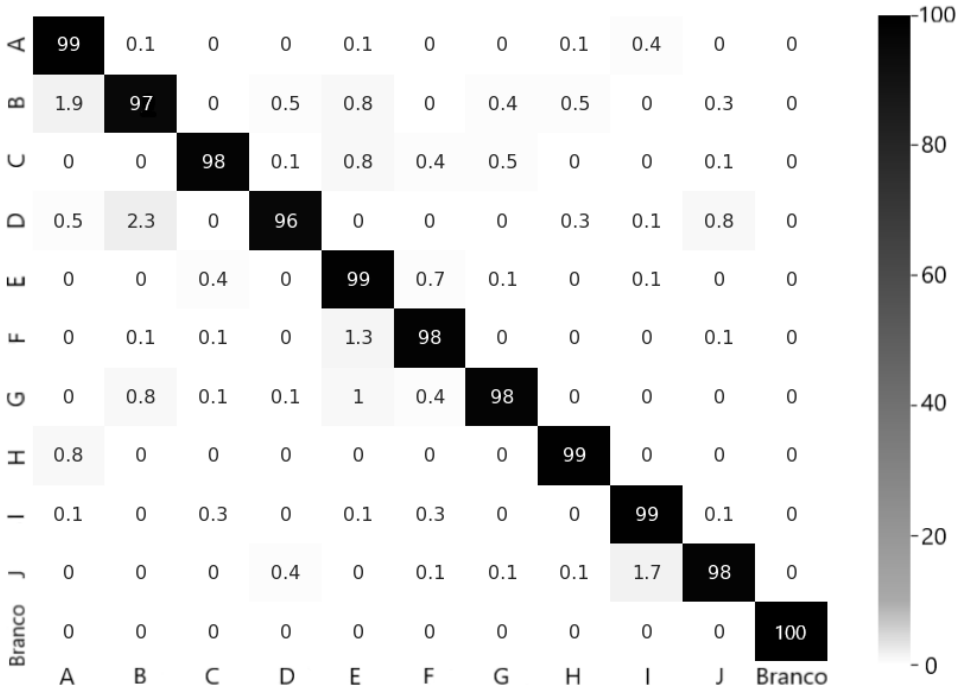


Figura 41 – Matriz de confusão (letras).

Fonte: De autoria própria.

Assim como a rede neural especializada em dígitos a CNN que classifica as letras foi analisada com 4 estruturas diferentes de CNN, onde a que mais se mostrou eficiente foi a quarta. Ainda que os resultados tenham sido bons não chegaram a ser tão bons quanto os resultados da CNN de dígitos. A quantidade de épocas necessária para habilitar o *early stopping* e parar o treinamento mais cedo se manteve entre 22 e 27 épocas, interalo similar ao encontrado na CNN de dígitos.

A Tabela 3 condensa o procedimento de otimização da CNN de letras com a evolução das métricas de precisão e perda por estrutura.

Estrutura	**Precisão**	**Perda**
Estrutura 1	93,36%	0,245
Estrutura 2	97,52%	0,116
Estrutura 3	98,10%	0,068
Estrutura 4	98,38%	0,067

Tabela 3 – Evolução da CNN de letras através das estruturas

4.2.3 Verdadeiro ou falso

Na Figura 42 pode-se observar a quantidade de amostras por classe utilizadas no treinamento da CNN de verdadeiro ou falso com uma média de 5000 amostras por classe.

Com um desafio menor que as CNN's anteriores (classificar somente 3 classes enquanto as CNN's de dígitos e letras classificam 11), é esperado que os resultados sejam melhores e alcançados com mais facilidade (menos épocas durante o treinamento).

- Estrutura 1

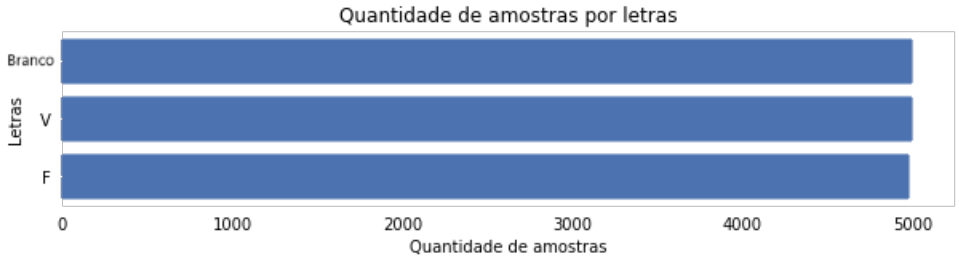


Figura 42 – Número de amostras por classe (verdadeiro ou falso).

Fonte: De autoria própria.

Assim como nas CNN's de dígitos e letras temos a estrutura 1 com uma camada convolucional, uma camada de *max pooling*, uma camada *flatten* e uma camada totalmente conectada. Essa estrutura mais básica já apresenta precisão de 99,02%.

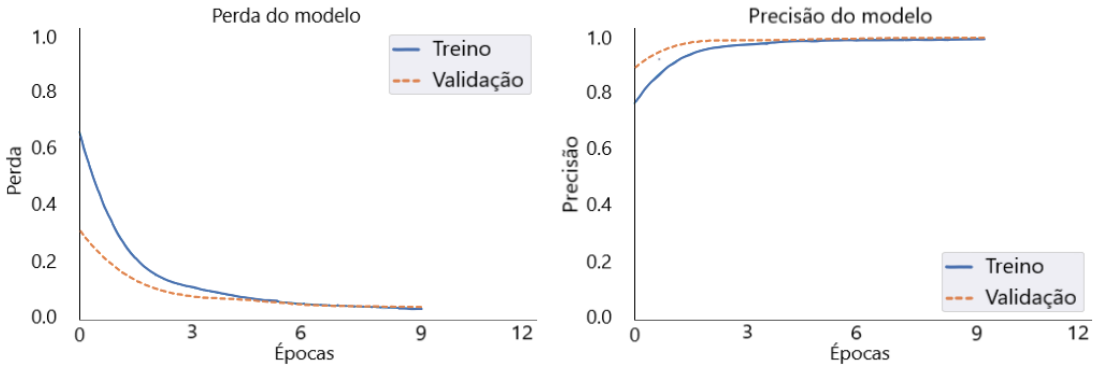
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_2 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 flatten_1 (Flatten)         (None, 7200)              0

 dense_2 (Dense)             (None, 64)                460864

 dense_3 (Dense)             (None, 3)                 195

=================================================================
Total params: 461,955
Trainable params: 461,955
Non-trainable params: 0
_________________________________________________________________
```

Figura 43 – Estrutura 1 da CNN de verdadeiro ou falso.

Fonte: De autoria própria.

Nos gráficos de precisão e perda (Figura 44) já é notável a diferença com as CNN's de dígitos e letras. Os dois gráficos apresentam curvas suaves, que convergem para os

valores esperados (1 para precisão e 0 para perda), e que não apresentam diferenças nos seus valores finais entre treino e validação ou oscilações.



(a) Gráfico de evolução do fator de perda durante o treinamento.

(b) Gráfico de evolução do fator de precisão durante o treinamento.

Figura 44 – Gráficos de perda e precisão (verdadeiro ou falso).

Fonte: De autoria própria.

A matriz de confusão - Figura 45 - mostra o bom resultado da estrutura, com a pior classe com 98% de precisão e o pior erro no valor de 1,9%.

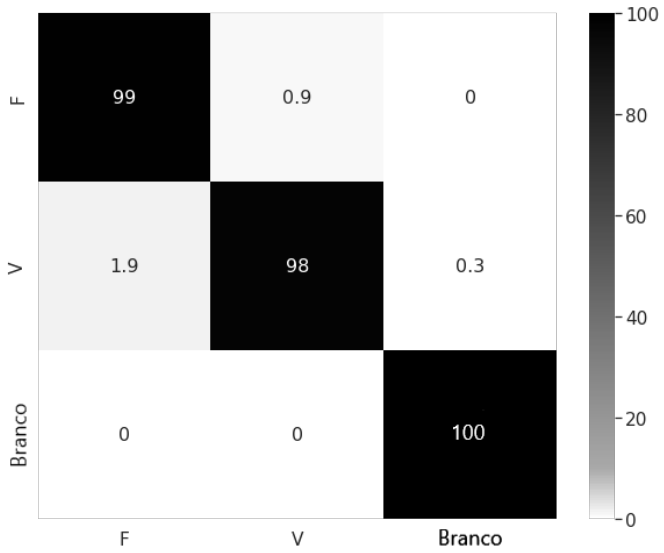


Figura 45 – Matriz de confusão (verdadeiro ou falso).

Fonte: De autoria própria.

- Estrutura 2

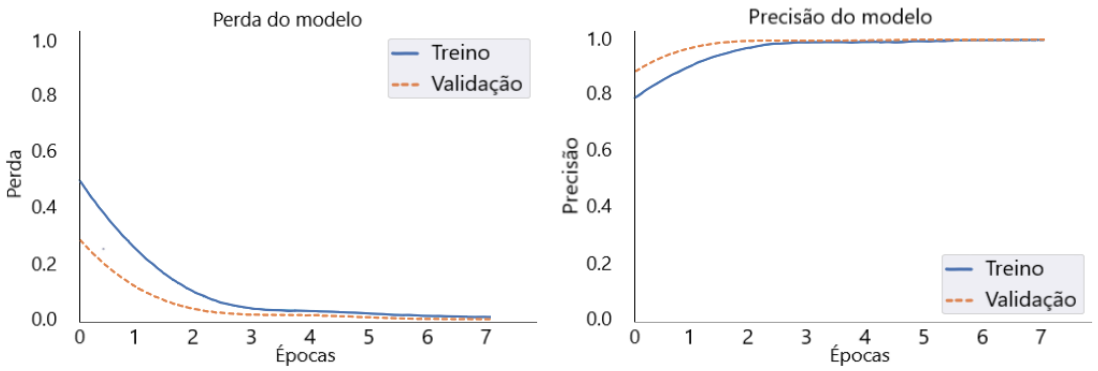
Com a adição de uma camada convolucional e uma camada de *pooling* máximo a CNN respondeu bem e chegou aos 99,68% de precisão - uma melhora de 0,66% comparado com a estrutura 1.

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_5 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_6 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_6 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten_4 (Flatten)         (None, 2304)              0

 dense_8 (Dense)             (None, 64)                147520

 dense_9 (Dense)             (None, 3)                 195

=================================================================
Total params: 167,107
Trainable params: 167,107
Non-trainable params: 0
_________________________________________________________________
```

Figura 46 – Estrutura 2 da CNN de verdadeiro ou falso.

Fonte: De autoria própria.

Os gráficos de perda e precisão (Figura 47), mais uma vez, apontam resultados bons e consistentes. Com curvas de treinamento e validação com convergência suave para valores ideias e sem oscilações ou diferença nos valores finais entre treino e validação.



(a) Gráfico de evolução do fator de perda durante o treinamento.

(b) Gráfico de evolução do fator de precisão durante o treinamento.

Figura 47 – Gráficos de perda e precisão (verdadeiro ou falso).

Fonte: De autoria própria.

A melhora na precisão é perceptível na matriz de confusão (Figura 48), com a pior classe em 99% de precisão e a maior taxa de confusão em 0,7%, os resultados reforçam a boa taxa de acurácia geral.

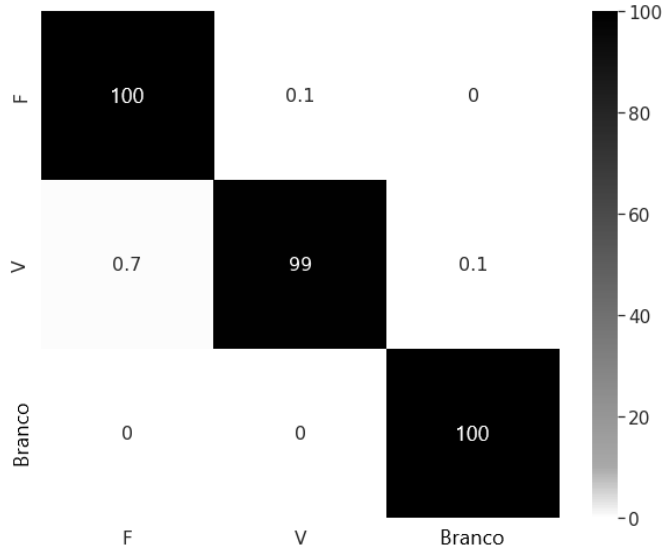


Figura 48 – Matriz de confusão (verdadeiro ou falso).

Fonte: De autoria própria.

- Estrutura 3

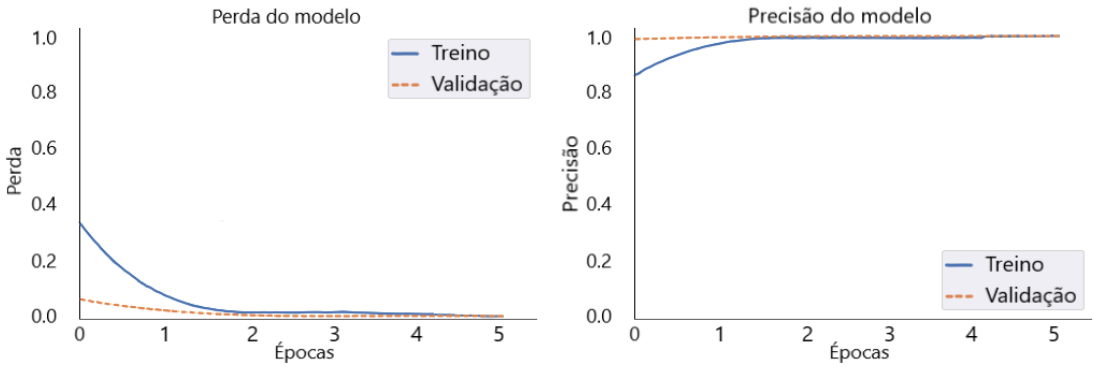
A estrutura 3 (Figura 49), consegue incrementar a sua qualidade com as camadas convolucional e de *max pooling* a mais, chegando à taxa de precisão de 99,89%.

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_9 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_10 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_10 (MaxPoolin  (None, 6, 6, 64)         0
 g2D)

 conv2d_11 (Conv2D)          (None, 4, 4, 64)          36928

 max_pooling2d_11 (MaxPoolin  (None, 2, 2, 64)         0
 g2D)

 flatten_6 (Flatten)         (None, 256)               0

 dense_12 (Dense)            (None, 64)                16448

 dense_13 (Dense)            (None, 3)                 195

=================================================================
Total params: 72,963
Trainable params: 72,963
Non-trainable params: 0
_________________________________________________________________
```

Figura 49 – Estrutura 3 da CNN de verdadeiro ou falso.

Fonte: De autoria própria.

Os gráficos da Figura 50, mais uma vez reforçam as altas taxas de precisão. Com valores de convergência ideias para perda (0) e precisão (1). As curvas convergem rapidamente (somente 5 épocas necessárias) e não apresentam desvios ou oscilações.



(a) Gráfico de evolução do fator de perda durante o treinamento.

(b) Gráfico de evolução do fator de precisão durante o treinamento.

Figura 50 – Gráficos de perda e precisão (verdadeiro ou falso).

Fonte: De autoria própria.

Por fim, a matriz de confusão da Figura 51 mostram uma matriz quase limpa, com somente uma pequena taxa de 0,6% de confusão entre as classes V e F. A pior classe é a V com 99% de precisão enquanto as classes F e *Branco* chegam aos 100%.

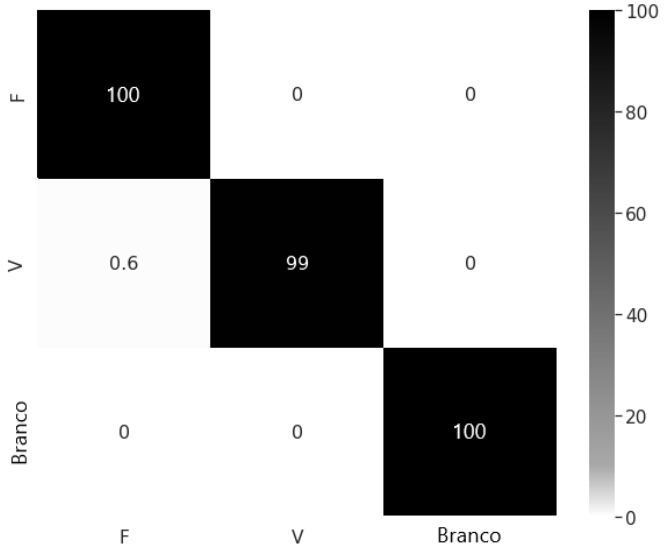


Figura 51 – Matriz de confusão (verdadeiro ou falso).

Fonte: De autoria própria.

Ao contrário das classificações de dígitos e letras foram analisadas somente 3 estruturas de CNN para a classificação de verdadeiro ou falso. Isso se deve ao fato de que a terceira estrutura já apresentou um resultado excelente (99,89% de precisão) e não foi necessário um quarto modelo. Outro ponto diferente das CNN's de dígitos e letras é a

quantidade de épocas necessárias para o *early stopping*, enquanto nas outras CNN's era em torno de 20 épocas a CNN de verdadeiro ou falso teve um mínimo de 5 e um máximo de 9 épocas.

Por fim, a Tabela 4 sintetiza o processo de otimização da CNN de verdadeiro ou falso com a evolução das métricas de precisão e perda.

Estrutura	**Precisão**	**Perda**
Estrutura 1	99,02%	0,039
Estrutura 2	99,68%	0,015
Estrutura 3	99,89%	0,008

Tabela 4 – Evolução da CNN de V ou F através das estruturas

5 CONCLUSÃO

Esse trabalho se propôs a desenvolver um método de reconhecimento de caracteres para facilitar a correção de cartões resposta do software Multiprova por meio do desenvolvimento de um fluxo de análise dos cartões resposta que culminasse no reconhecimento das letras escritas pelos alunos e na correção automática das provas.

Para isolar e identificar as respostas escritas nos cartões resposta foram utilizadas técnicas de segmentação das imagens baseando-se em marcas fixas impressas no cartões.

Já para o reconhecimento das letras e números escritos nos cartões foram criadas e treinadas três redes neurais convolucionais (para dígitos, letras e verdadeiro ou falso).

Os resultados conseguidos (98,84% de precisão para CNN de dígitos, 98,38% de precisão para CNN de letras e 99,89% de precisão para CNN de verdadeiro ou falso) apontam uma ótima taxa média de acerto, ainda que exista espaço para melhorias.

Com os resultados obtidos, só é necessário exportar os pesos das CNN's já treinadas e os inserir no aplicativo do Multiprova (Multiprova Corretor). O aplicativo é feito com React Native e por isso tem suporte às bibliotecas do TensorFlow que permitem a importação de redes neurais já treinadas.

O aplicativo recebeu o arquivo contendo os pesos do treinamento da CNN de dígitos a fim de fazer o reconhecimento das matrícula do aluno e o código da prova. Com matrícula e código da prova validados, essa informação e a imagem do cartão são enviados ao *back-end* do sistema Multiprova, onde foi construído um serviço a parte a fim de realizar as predições das imagens dos cartões resposta. Esse serviço contém as três CNN's desenvolvidas neste trabalho.

O aplicativo Multiprova Corretor está disponível na Play Store. Isto permite que os professores façam *download* e utilizem a aplicação nos seus próprios aparelhos e não dependam de meios de terceiros (máquina de correção de cartões como era feito na versão anterior do Multiprova) para efetuar a correção automática de suas provas. A aplicação está disponibilizada través do link (https://play.google.com/store/apps/details?id=br.ufrn.multiprova).

Após os pesos serem configurados no aplicativo e as redes neurais integradas ao fluxo de correção, o professor tem a comodidade de realizar a correção de suas provas de forma simples, rápida e com pouco esforço.

A Figura 52 demonstra como as redes neurais estão dispostas na estrutura do sistema Multiprova.

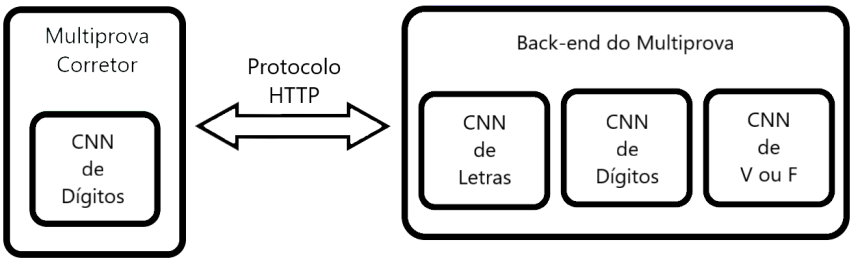


Figura 52 – Situação das redes neurais no projeto Multiprova.

Fonte: De autoria própria.

Ainda que as redes tenham apresentado altas taxas de precisão, elas ainda são passíveis de cometer erros ao classificar as respostas. Pensando nisso foi estipulado um alto nível de rigorosidade para considerar uma predição como confiável.

O processo de classificação retorna os índices de certeza do modelo para todas as classes da CNN utilizada. Qualquer classificação com menos de 99,9% de certeza é validada com o próprio professor através do aplicativo.

A fim de sempre buscar melhorar a aplicação é importante pontuar os seguintes aperfeiçoamentos:

- Ajustes finos nas configurações das CNN's;
- Recuperação e reutilização de imagens com baixos níveis de precisão para retreinamento das redes;
- Expansão do *dataset* de letras com imagens de novas classes (A a Z) para treinamento de uma CNN de letras mais abrangente.

Todos os *notebooks* do Google Colab utilizados para limpeza e aumento de dados, criação dos *datasets* e treinamento das CNN's estão disponibilizados no seguinte link (https://github.com/Gduodq/CNN-Training) e podem ser utilizados publicamente.

REFERÊNCIAS

ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>.

ACADEMY, D. S. *Introdução as Redes Neurais Convolucionais*. 2018. Online. Disponível em: <https://www.deeplearningbook.com.br/introducao-as-redes-neurais-convolucionais-/>. Acesso em: 07 jan. 2022.

ACHARYA, M. *Building a rock paper scissors AI using Tensorflow and OpenCV*. Towards Data Science, 2020. Disponível em: <https://towardsdatascience.com/building-a-rock-paper-scissors-ai-using-tensorflow-and-opencv-d5fc44fc8222>.

ALVES, G. *Entendendo Redes Convolucionais (CNNs)*. 2018. Online. Disponível em: <https://medium.com/neuronio-br/entendendo-redes-convolucionais-cnns-d10359f21184>. Acesso em: 07 jan. 2022.

AULT, J.; HANNA, J. P.; SHARON, G. Learning an interpretable traffic signal control policy. *arXiv preprint arXiv:1912.11023*, 2019.

AYODELE, T. O. Types of machine learning algorithms. *New advances in machine learning*, InTech Rijeka, Croatia, v. 3, p. 19–48, 2010.

BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

CASALE, S. et al. Speech emotion classification using machine learning algorithms. In: IEEE. *2008 IEEE international conference on semantic computing*. [S.l.], 2008. p. 158–165.

CHOLLET, F. et al. *Keras*. GitHub, 2015. Disponível em: <https://github.com/fchollet/keras>.

COHEN, G. et al. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

CONTRIBUTORS, M. *HTTP request methods*. 2021. Online. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>. Acesso em: 06 jan. 2022.

DAELEMANS, W.; HOSTE, V. Evaluation of machine learning methods for natural language processing tasks. In: EUROPEAN LANGUAGE RESOURCES ASSOCIATION (ELRA). *3rd International conference on Language Resources and Evaluation (LREC 2002)*. [S.l.], 2002.

DAI, X. et al. Machine learning on mobile: An on-device inference app for skin cancer detection. In: IEEE. *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*. [S.l.], 2019. p. 301–305.

FLANAGAN, D. *JavaScript: the definitive guide*. [S.l.]: "O'Reilly Media, Inc.", 2006.

KALITA, D. *Basics of CNN in Deep Learning*. 2022. Disponível em: <https://www-.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/>.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Ieee, v. 86, n. 11, p. 2278–2324, 1998.

LI, S. et al. Deep learning for hyperspectral image classification: An overview. *IEEE Transactions on Geoscience and Remote Sensing*, IEEE, v. 57, n. 9, p. 6690–6709, 2019.

MCCARTHY, J. What is artificial intelligence? 2007.

MNIST database. Wikimedia Foundation, 2022. Disponível em: <https://en.wikipedia-.org/wiki/MNIST_database>.

ORACLE. *O Que É um Banco de Dados?* 2021. Online. Disponível em: <https://www-.oracle.com/br/database/what-is-database/>. Acesso em: 06 jan. 2022.

PARKHI, O. M. et al. Cats and dogs. In: IEEE. *2012 IEEE conference on computer vision and pattern recognition*. [S.l.], 2012. p. 3498–3505.

PEZOA, F. et al. Foundations of json schema. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. *Proceedings of the 25th International Conference on World Wide Web*. [S.l.], 2016. p. 263–273.

ROSSUM, G. V.; JR, F. L. D. *Python reference manual*. [S.l.]: Centrum voor Wiskunde en Informatica Amsterdam, 1995.

SEBE, N. et al. *Machine learning in computer vision*. [S.l.]: Springer Science & Business Media, 2005. v. 29.

VARGAS, A. C. G.; CARVALHO, A. M. P.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. *Proceedings of the xxix conference on graphics, patterns and images*, v. 1, n. 4, 2016.

VELOSO, L. R. et al. Reconhecimento de caracteres numéricos manuscritos. Universidade Federal de Campina Grande, 1998.

YANG, Y. A study of pattern recognition of iris flower based on machine learning. Turun ammattikorkeakoulu, 2013.