**UNIVERSIDADE FEDERAL DA INTEGRAÇÃO LATINO-AMERICANA**

**FERNANDO JOSÉ ZARDINELLO BATISTTI**

# PROJETO II: CLASSIFICAÇÃO DE DÍGITOS MANUSCRITOS COM REDES NEURAIS CONVOLUCIONAIS

**FOZ DO IGUAÇU**

**2025**

**FERNANDO JOSÉ ZARDINELLO BATISTTI**

# PROJETO II: CLASSIFICAÇÃO DE DÍGITOS MANUSCRITOS COM REDES NEURAIS CONVOLUCIONAIS

## Project II: Handwritten Digit Classification with Convolutional Neural Networks

Relatório Técnico apresentado à disciplina de Tópicos Especiais em Física Experimental (FIS0006), do curso de Física Aplicada da Universidade Federal da Integração Latino-Americana (UNILA).

Orientador: Prof. Dr. Joylan Nunes Maciel

**FOZ DO IGUAÇU**
**2025**

# RESUMO

Este trabalho detalha o desenvolvimento de um modelo de *Deep Learning* para a classificação de dígitos manuscritos. Utilizando uma Rede Neural Convolucional (CNN), implementada com as bibliotecas TensorFlow e Keras, o projeto aborda o *pipeline* completo, desde o pré-processamento de imagens do dataset MNIST até a avaliação e otimização do modelo. A arquitetura base da CNN, inspirada nos modelos clássicos e validada através de validação cruzada estratificada, alcançou uma acurácia de 99,24% no conjunto de teste. Uma etapa de otimização de hiperparâmetros foi conduzida com o KerasTuner para explorar melhorias de performance. O resultado final é um modelo robusto e de alta precisão, demonstrando a eficácia das CNNs para tarefas de reconhecimento de padrões em imagens e contextualizando seu desempenho frente ao estado da arte.

**Palavras-chave:** Redes Neurais Convolucionais; Deep Learning; Classificação de Imagens; MNIST; TensorFlow.

## ABSTRACT

This work details the development of a Deep Learning model for handwritten digit classification. Using a Convolutional Neural Network (CNN), implemented with the TensorFlow and Keras libraries, the project covers the complete pipeline from image preprocessing of the MNIST dataset to model evaluation and optimization. The baseline CNN architecture, inspired by classic models and validated through stratified cross-validation, achieved an accuracy of 99.24% on the test set. A hyperparameter optimization step was conducted using KerasTuner to explore performance improvements. The final result is a robust, high-precision model, demonstrating the effectiveness of CNNs for image pattern recognition tasks and contextualizing its performance against the state-of-the-art.

**Keywords:** Convolutional Neural Networks; Deep Learning; Image Classification; MNIST; TensorFlow.

## LISTA DE FIGURAS

## LISTA DE TABELAS

# LISTINGS

**SUMÁRIO**

# 1 INTRODUÇÃO

O reconhecimento de dígitos manuscritos é uma das tarefas mais fundamentais e canônicas no campo da visão computacional e do aprendizado de máquina. Embora pareça um problema simples para humanos, ensinar uma máquina a classificar a vasta variedade de estilos de caligrafia com alta precisão representa um desafio significativo que impulsionou o desenvolvimento de algoritmos poderosos. A solução para este problema tem aplicações práticas diretas, como a leitura automática de CEPs em envelopes, o processamento de cheques bancários e a digitalização de formulários.

O dataset MNIST e suas variações se tornaram o padrão-ouro para testar e validar algoritmos de classificação de imagem (LECUN et al., 1998). Com o advento do *Deep Learning*, as Redes Neurais Convolucionais (CNNs) emergiram como a abordagem de estado da arte para esta e outras tarefas de visão computacional, devido à sua capacidade de aprender hierarquias de características espaciais diretamente dos dados brutos (pixels).

Este trabalho se propõe a construir, treinar e avaliar um modelo de CNN para a classificação de dígitos. A questão de pesquisa que guia o projeto é: é possível desenvolver um modelo de CNN, utilizando ferramentas como TensorFlow e Keras, que atinja um nível de acurácia competitivo, superior a 99%, na classificação de um grande dataset de dígitos manuscritos?

O objetivo principal é, portanto, implementar um *pipeline* completo de *Deep Learning*, incluindo uma fundamentação teórica sobre a evolução das arquiteturas, a validação robusta do modelo e a otimização de seus hiperparâmetros, culminando em um classificador de alta performance contextualizado no cenário atual da área.

# 2 FUNDAMENTAÇÃO TEÓRICA

Esta seção detalha os conceitos de *Deep Learning* que formam a base deste projeto, desde as arquiteturas seminais até os componentes modernos de treinamento.

## 2.1 A Evolução das Redes Neurais Convolucionais (CNNs)

As CNNs são uma classe de redes neurais especializada no processamento de dados com topologia de grade, como imagens. Sua arquitetura é bio-inspirada no córtex visual humano e seu sucesso reside na capacidade de extrair hierarquias de características de forma automática.

### 2.1.1 LeNet-5: A Arquitetura Pioneira

Introduzida por LeCun et al. (1998), a LeNet-5 é considerada a primeira CNN de sucesso e a base para muitas arquiteturas modernas. Ela estabeleceu o padrão de empilhar camadas convolucionais, seguidas por camadas de *pooling* (subamostragem), culminando em camadas totalmente conectadas para a classificação final. Seus componentes principais, ainda hoje utilizados, são:

- **Camada Convolucional (Conv2D):** Aplica um conjunto de filtros (kernels) à imagem para detectar características locais como bordas e texturas.
- **Camada de Pooling (Average/Max Pooling):** Reduz a dimensionalidade espacial dos mapas de características, conferindo invariância a pequenas translações e diminuindo a carga computacional.
- **Camadas Totalmente Conectadas (Dense):** Realizam a classificação com base nas características de alto nível extraídas pelas camadas anteriores.

### 2.1.2 A Revolução da Profundidade e da Escala

O campo progrediu significativamente com o aumento do poder computacional e do volume de dados.

- **AlexNet:** Em 2012, a AlexNet demonstrou o poder das CNNs em larga escala, vencendo a competição ImageNet. Suas inovações incluíram o uso da função de ativação ReLU, a

regularização com *Dropout* e o treinamento em GPUs, permitindo modelos muito mais profundos e largos que a LeNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

- **VGGNet:** Esta arquitetura mostrou que a profundidade era um fator crítico para o desempenho. Ao usar filtros convolucionais muito pequenos e uniformes (3x3), foi possível construir redes de até 19 camadas, alcançando uma performance superior com uma arquitetura mais simples e elegante (SIMONYAN; ZISSERMAN, 2014).

- **ResNet:** As Redes Residuais introduziram as "conexões de atalho"(*skip connections*), resolvendo o problema da degradação (onde redes mais profundas performavam pior). Isso permitiu o treinamento de redes com centenas ou até milhares de camadas, estabelecendo novos recordes de performance (HE et al., 2016). O bloco residual aprende a função $F(x) := H(x) - x$, onde a rede pode facilmente aprender uma identidade ($F(x) = 0$) se a camada não for necessária, facilitando a otimização.

## 2.2 Componentes Essenciais do Treinamento

- **Funções de Ativação:** A função `ReLU` (*Rectified Linear Unit*), popularizada pela AlexNet, é o padrão para introduzir não-linearidade devido à sua simplicidade e eficiência computacional. Na camada de saída para classificação multiclasse, a função `Softmax` converte os logits em uma distribuição de probabilidade.

- **Otimizadores:** O otimizador `Adam` (*Adaptive Moment Estimation*) é um algoritmo de otimização estocástica que adapta a taxa de aprendizado para cada parâmetro, combinando as vantagens do RMSprop e do SGD com momento, sendo amplamente utilizado por sua eficiência e bom desempenho (KINGMA; BA, 2014).

- **Regularização:** Para evitar o sobreajuste (*overfitting*), diversas técnicas são empregadas:

  - **Dropout:** Zera aleatoriamente uma fração das saídas dos neurônios durante o treinamento, forçando a rede a aprender representações mais robustas e distribuídas (SRIVASTAVA et al., 2014).

  - **Batch Normalization:** Normaliza as ativações de uma camada para ter média zero e variância unitária, combatendo o problema do "desvio covariante interno"(*internal covariate shift*). Isso estabiliza e acelera o treinamento, permitindo taxas de aprendizado mais altas e reduzindo a dependência de outras formas de regularização (IOFFE; SZEGEDY, 2015).

- **Função de Perda:** A `categorical_crossentropy` é a função de perda padrão para problemas de classificação multiclasse, medindo a dissimilaridade entre a distribuição de probabilidade prevista e a distribuição real (*one-hot encoded*).

# 3 DESENVOLVIMENTO

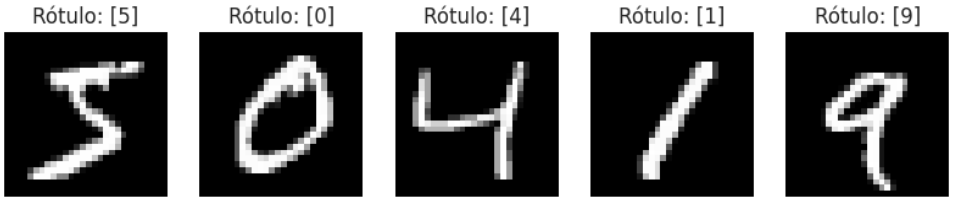
Esta seção detalha o processo prático de construção do modelo, implementado em Python com o ecossistema de bibliotecas TensorFlow e Keras.

## 3.1 Coleta e Pré-processamento dos Dados

O projeto utilizou o dataset MNIST, uma versão reconstruída e verificada do clássico MNIST, contendo 120.000 imagens de treino e 120.000 de teste (YADAV; BOTTOU, 2019), conforme exemplificado na Figura 1. As imagens, com dimensões de 28x28 pixels, foram carregadas e submetidas a três etapas de pré-processamento:

1. **Normalização:** Os valores de pixel, originalmente no intervalo [0, 255], foram escalonados para o intervalo [0, 1] ao serem divididos por 255.0.
2. **Remodelação (*Reshape*):** Os dados foram remodelados para o formato (amostras, 28, 28, 1), compatível com a entrada de uma CNN.
3. ***One-Hot Encoding*:** Os rótulos inteiros (0 a 9) foram convertidos para um formato vetorial binário (categórico).

**Figura 1 – Amostra de cinco dígitos do dataset MNIST.**

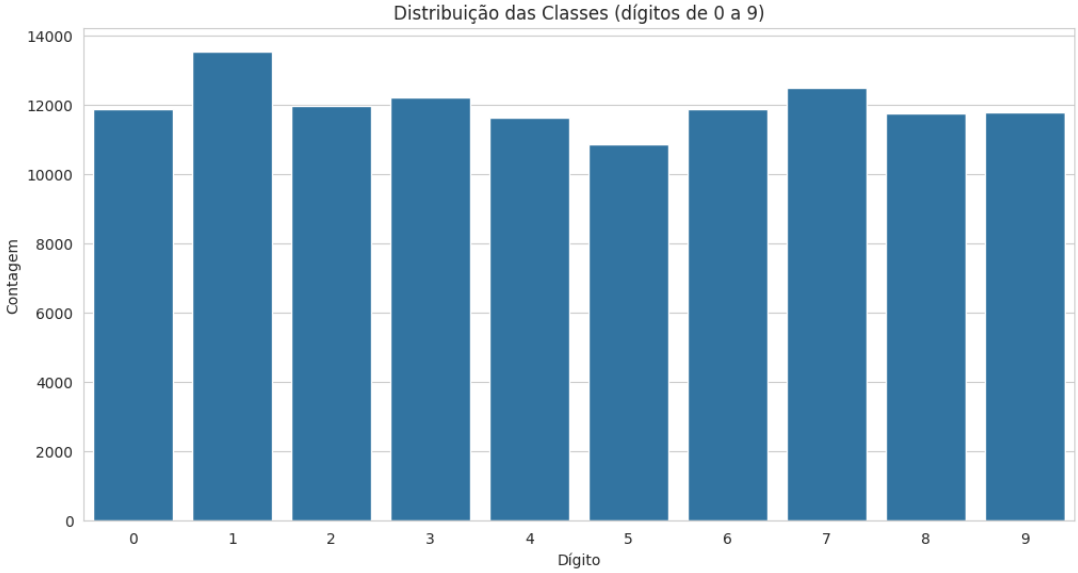



**Fonte: Autor (2025).**

## 3.2 Análise Exploratória e Modelo Base

A análise exploratória confirmou o balanceamento do dataset entre as 10 classes (Figura 2). Uma arquitetura de CNN base, inspirada na LeNet-5, foi proposta e sua robustez foi avaliada através de validação cruzada (‘StratifiedKFold‘) com 3 folds. As curvas de aprendizado (Figuras 3, 4 e 5) demonstraram uma convergência estável com sobreajuste mínimo, atingindo uma acurácia média de **99,19%** nos folds de validação.

**Figura 2 – Distribuição das classes no dataset MNIST.**

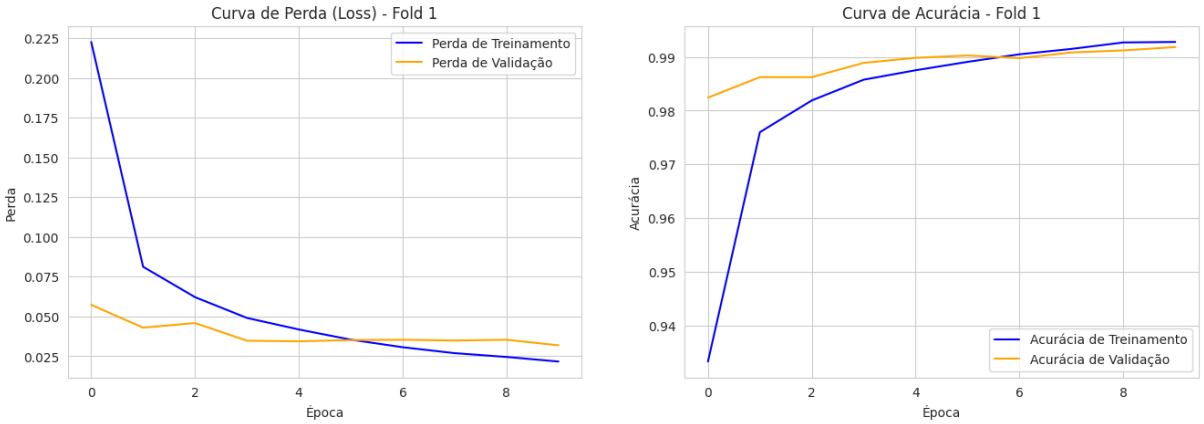



**Fonte: Autor (2025).**

## 3.3 Otimização de Hiperparâmetros

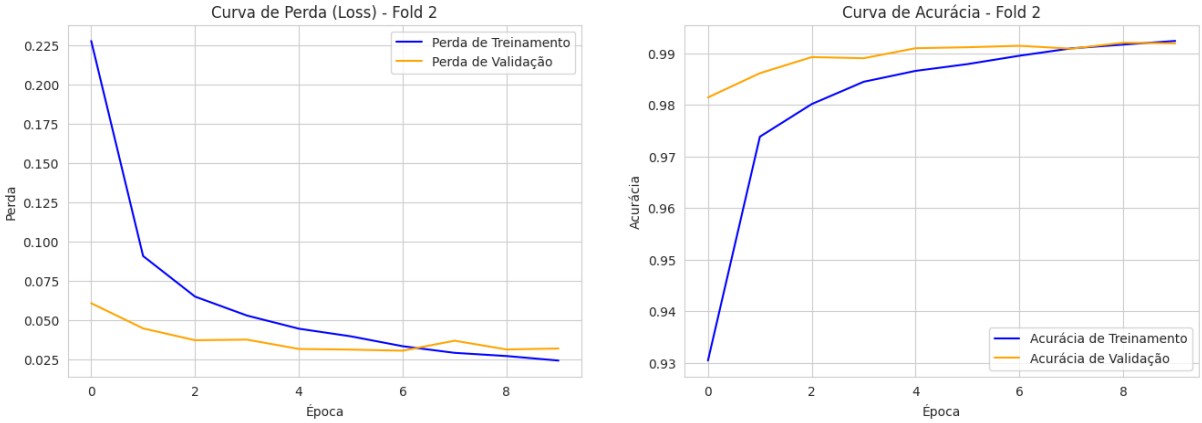
Para investigar a possibilidade de melhorias, foi conduzido um processo de otimização de hiperparâmetros com o KerasTuner (O'MALLEY et al., 2019). Utilizando a técnica 'RandomSearch', foram testadas 5 combinações de hiperparâmetros (número de filtros, tamanho de kernel, unidades densas, etc.) por um número limitado de épocas.

**Figura 3 – Curvas de aprendizado (perda e acurácia) para o Fold 1 da validação cruzada.**



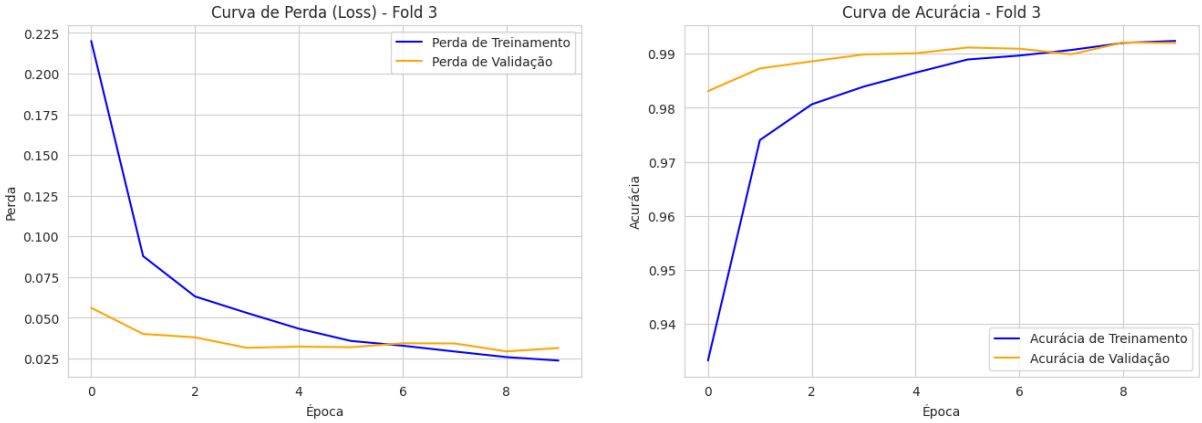

**Fonte: Autor (2025).**

**Figura 4 – Curvas de aprendizado (perda e acurácia) para o Fold 2 da validação cruzada.**




**Fonte: Autor (2025).**

**Figura 5 – Curvas de aprendizado (perda e acurácia) para o Fold 3 da validação cruzada.**




**Fonte: Autor (2025).**

# 4 RESULTADOS E DISCUSSÃO

Neste capítulo, são apresentados e discutidos os resultados quantitativos do modelo base e do modelo otimizado, contextualizando-os com o estado da arte.

## 4.1 Desempenho do Modelo Base

O modelo base, após ser treinado em todo o conjunto de dados de treino por 15 épocas, foi avaliado no conjunto de teste, alcançando uma acurácia final de **99,24%**. O relatório de classificação (Tabela 1) detalha as métricas de precisão, recall e f1-score para cada classe, todas consistentemente altas. A matriz de confusão (Figura 6) ilustra visualmente o baixíssimo número de erros de classificação, demonstrando a alta performance do modelo.

**Tabela 1 – Relatório de classificação do modelo base no conjunto de teste.**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **0** | 1.00 | 0.99 | 1.00 | 2375 |
| **1** | 1.00 | 0.99 | 0.99 | 2707 |
| **2** | 0.99 | 0.99 | 0.99 | 2397 |
| **3** | 1.00 | 0.99 | 0.99 | 2443 |
| **4** | 0.99 | 0.99 | 0.99 | 2324 |
| **5** | 0.99 | 0.99 | 0.99 | 2175 |
| **6** | 0.99 | 0.99 | 0.99 | 2375 |
| **7** | 0.98 | 1.00 | 0.99 | 2499 |
| **8** | 0.99 | 0.99 | 0.99 | 2348 |
| **9** | 0.99 | 0.99 | 0.99 | 2357 |
| **accuracy** | | | 0.99 | 24000 |
| **macro avg** | 0.99 | 0.99 | 0.99 | 24000 |
| **weighted avg** | 0.99 | 0.99 | 0.99 | 24000 |

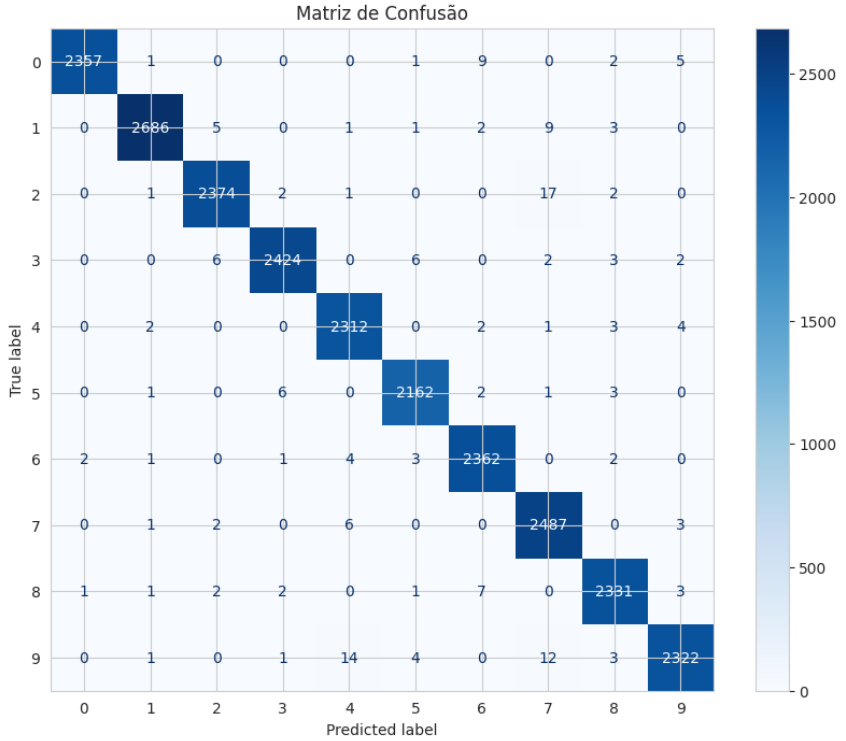
**Fonte: Autor (2025).**

## 4.2 Análise Comparativa e Contextualização

O processo de otimização com KerasTuner, limitado a 5 tentativas, encontrou uma arquitetura que resultou em uma acurácia de **97,55%** no teste, inferior ao modelo base (Tabela 2). Este resultado, aparentemente contraintuitivo, sugere que a busca por hiperparâmetros não foi extensa o suficiente para superar uma arquitetura inicial já bem-sucedida e robusta.

Em comparação com o estado da arte (SOTA) no dataset MNIST, que reporta acurácias

**Figura 6 – Matriz de confusão para o modelo base no conjunto de teste.**




**Fonte: Autor (2025).**

de até 99.67% com métodos mais complexos, o resultado de 99,24% é altamente competitivo para um único modelo de CNN sem augmentation ou ensembles. Isso confirma que a arquitetura e a metodologia adotadas são sólidas.

**Tabela 2 – Comparação de desempenho entre o modelo base e o otimizado.**

| Métrica | Modelo Base | Modelo Otimizado |
|---|---|---|
| Acurácia no Teste | 99.24% | 97.55% |

**Fonte: Autor (2025).**

# 5 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho demonstrou com sucesso a construção de uma Rede Neural Convolucional de alta performance para a classificação de dígitos manuscritos, respondendo afirmativamente à pergunta de pesquisa ao alcançar uma acurácia de 99,24%. O *pipeline* implementado, desde a análise exploratória e fundamentação teórica até a validação cruzada rigorosa, provou ser uma abordagem eficaz para o desenvolvimento de modelos de *Deep Learning*.

O resultado mais notável foi o desempenho superior do modelo base em comparação com o modelo resultante do processo de otimização limitado. Este achado ressalta um ponto importante na prática de *Machine Learning*: a otimização de hiperparâmetros não é uma panaceia e seu sucesso depende de uma busca suficientemente exaustiva no espaço de busca, sendo que uma arquitetura bem projetada pode ser um ponto de partida superior.

Para trabalhos futuros, diversas avenidas promissoras podem ser exploradas. A primeira é a aplicação de técnicas de *data augmentation*, como rotações, translações e, especialmente, distorções elásticas, que são conhecidas por melhorar a robustez em datasets de dígitos. Uma segunda etapa seria conduzir uma busca de hiperparâmetros mais ampla e sistemática. Finalmente, o modelo treinado e salvo (‘.h5‘ ou ‘SavedModel‘) poderia ser implantado como uma API REST utilizando um microframework como o Flask, permitindo sua integração em aplicações reais para inferência em tempo real.

Além disso, a exploração de arquiteturas mais recentes que transcendem as CNNs clássicas, como as *Capsule Networks* (CapsNets) (SABOUR; FROSST; HINTON, 2017), que visam uma melhor compreensão das relações espaciais, ou os *Vision Transformers* (ViT) (DOSOVITSKIY et al., 2020), que aplicam o sucesso dos modelos de atenção à visão computacional, representaria um avanço significativo na pesquisa.

No fim o projeto não apenas atingiu seu objetivo de criar um classificador preciso, mas também forneceu clareza sobre o processo de validação, otimização e contextualização de modelos de *Deep Learning*, além de possibilitar traçar um caminho transparente para futuras melhorias e aplicações baseado em ideias de trabalhos mais desenvolvidos já citados.

# REFERÊNCIAS

DOSOVITSKIY, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. **arXiv preprint arXiv:2010.11929**, 2020.

HE, K. et al. Deep residual learning for image recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. 2016. p. 770–778.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **arXiv preprint arXiv:1502.03167**, 2015. Disponível em: https://arxiv.org/abs/1502.03167.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014. Disponível em: https://arxiv.org/abs/1412.6980.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in Neural Information Processing Systems 25**. 2012. p. 1097–1105.

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

O'MALLEY, T. et al. **KerasTuner**. GitHub, 2019. `https://github.com/keras-team/keras-tuner`.

SABOUR, S.; FROSST, N.; HINTON, G. E. Dynamic routing between capsules. In: **Advances in Neural Information Processing Systems 30**. 2017. p. 3856–3866.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014. Disponível em: https://arxiv.org/abs/1409.1556.

SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, 2014.

YADAV, C.; BOTTOU, L. Cold case: The lost mnist digits. In: **Advances in Neural Information Processing Systems**. 2019. v. 32. Disponível em: https://proceedings.neurips.cc/paper/2019/hash/c3a690be96c20da8d35359c4a443bc6e-Abstract.html.