INSTITUTO LATINO-AMERICANO DE CIÊNCIAS DA VIDA E DA NATUREZA (ILACVN)
PROGRAMA DE PÓS-GRADUAÇÃO EM FÍSICA APLICADA

UNIVERSIDADE FEDERAL DA INTEGRAÇÃO LATINO-AMERICANA - UNILA

Projeto apresentado na disciplina FIS0006 - Tópicos Especiais em Física Experimental

Projeto II: Classificação de Dígitos Manuscritos com Redes Neurais Convolucionais

Discente:
Fernando José Zardinello Batistti

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 TensorFlow e Keras, o projeto aborda o *pipeline* completo, desde o pré-processamento de um dataset de 120.000 imagens, conhecido como MNIST-120k e derivado do QMNIST, até a sua avaliação e otimização. A arquitetura base da CNN, validada através de validação cruzada, alcançou uma acurácia de 99,24% em um conjunto de teste de 24.000 imagens (separado a partir de uma divisão 80/20 do total). Uma etapa de otimização foi conduzida com o KerasTuner, e o modelo final demonstrou a eficácia das CNNs para tarefas de reconhecimento de padrões, com seu desempenho contextualizado frente ao estado da arte.

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

Abstract

This work details the development of a Deep Learning model for handwritten digit classification. Using a Convolutional Neural Network (CNN), implemented with TensorFlow and Keras, the project covers the complete pipeline, from the preprocessing of a 120,000-image dataset, known as MNIST-120k and derived from QMNIST, to its evaluation and optimization. The baseline CNN architecture, validated through cross-validation, achieved an accuracy of 99.24% on a 24,000-image test set (separated from an 80/20 split of the total). An optimization step was conducted using KerasTuner, and the final model demonstrated the effectiveness of CNNs for pattern recognition tasks, with its performance contextualized against the state-of-the-art.

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

Sumário

Lista de Figuras

Lista de Tabelas

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 [7]. 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. [7], 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.

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 [6].

- **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 [10].

- **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 [3].

2.2 Componentes Essenciais do Treinamento

Para construir e treinar o modelo, diversos componentes são necessários:

- **Funções de Ativação:** A função `ReLU` (*Rectified Linear Unit*) é o padrão para introduzir não-linearidade. Na camada de saída, a função `Softmax` converte os logits em uma distribuição de probabilidade.
- **Otimizadores:** O otimizador `Adam` (*Adaptive Moment Estimation*) é um algoritmo eficiente que adapta a taxa de aprendizado para cada parâmetro do modelo [5].
- **Regularização:** Para evitar o sobreajuste (*overfitting*), técnicas como *Dropout* [11] e *Batch Normalization* [4] são essenciais.
- **Função de Perda:** A `categorical_crossentropy` mede a dissimilaridade entre a previsão do modelo e o rótulo verdadeiro.

3. Desenvolvimento

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

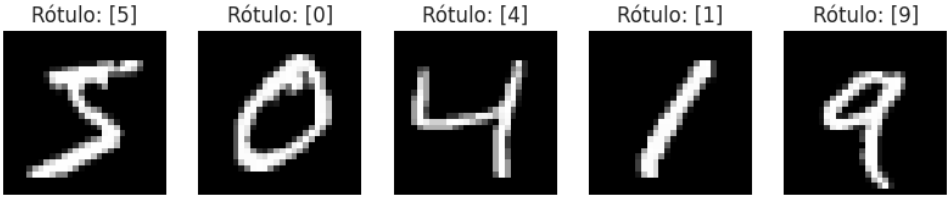
3.1 Coleta e Pré-processamento dos Dados

O projeto utilizou um conjunto de dados obtido na plataforma Kaggle, popularmente conhecido como "MNIST-120k"[1]. Este dataset contém um total de 120.000 imagens de dígitos manuscritos. É importante notar que estes dados são uma versão expandida e com rótulos verificados, originados do trabalho de reconstrução do QMNIST por Yadav e Bottou [12].

Para o desenvolvimento deste trabalho, o conjunto completo de 120.000 imagens foi primeiramente embaralhado e, em seguida, dividido em uma proporção de 80% para treinamento (96.000 imagens) e 20% para teste (24.000 imagens). As imagens, com dimensões de 28x28 pixels, foram 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].
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.

Figura 3.1: Amostra de cinco dígitos do dataset utilizado.

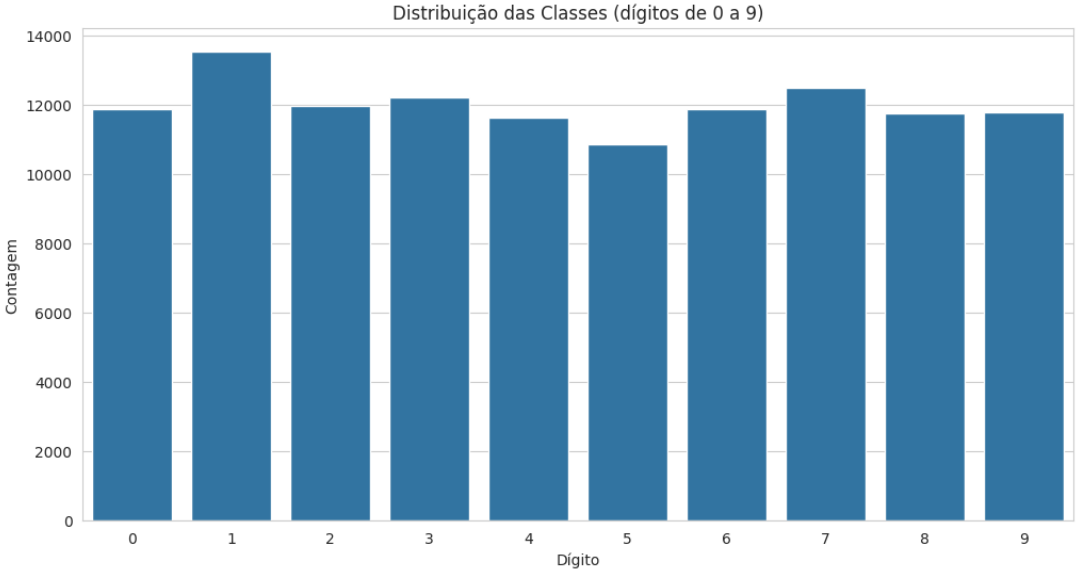


Fonte: Autor (2025).

3.2 Análise Exploratória e Modelo Base

A análise exploratória confirmou o balanceamento do dataset entre as 10 classes (Figura 3.2). Uma arquitetura de CNN base foi proposta e sua robustez foi avaliada através de validação cruzada (‘StratifiedKFold‘). As curvas de aprendizado (Figuras 3.3, 3.4 e 3.5) demonstraram uma convergência estável, atingindo uma acurácia média de **99,19%** nos folds de validação.

Figura 3.2: Distribuição das classes no dataset de 120.000 imagens.

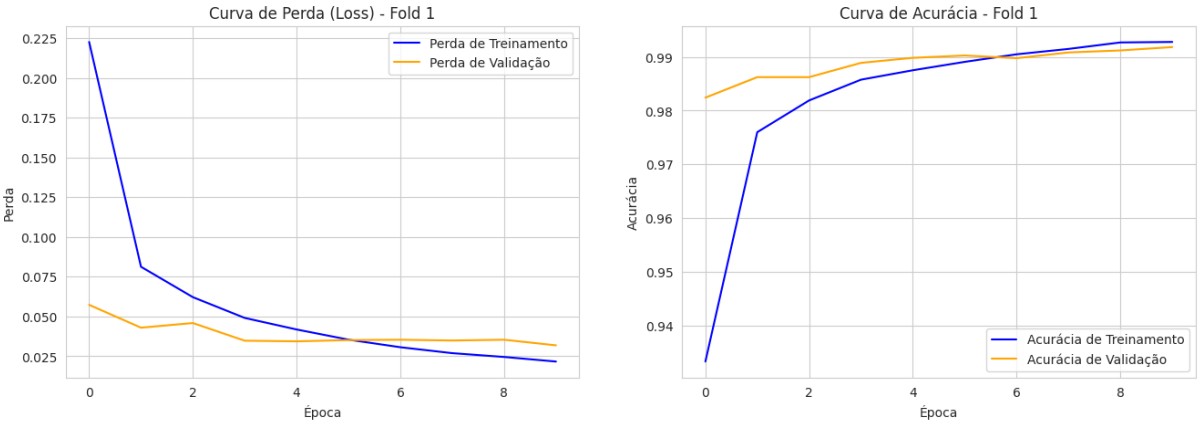


Fonte: Autor (2025).

3.3 Otimização de Hiperparâmetros

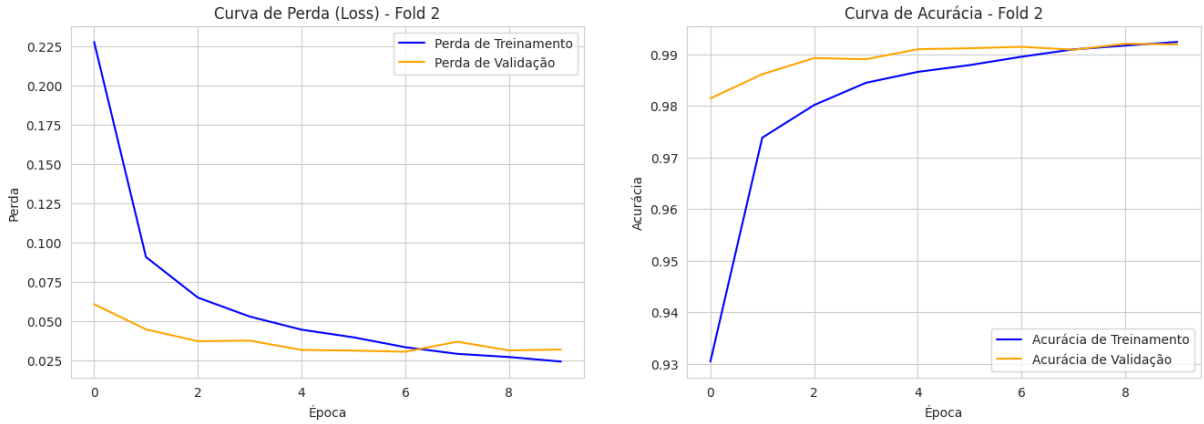
Para investigar a possibilidade de melhorias, foi conduzido um processo de otimização de hiperparâmetros com o KerasTuner [8], utilizando a técnica 'RandomSearch'.

Figura 3.3: Curvas de aprendizado (perda e acurácia) para o Fold 1 da validação cruzada.



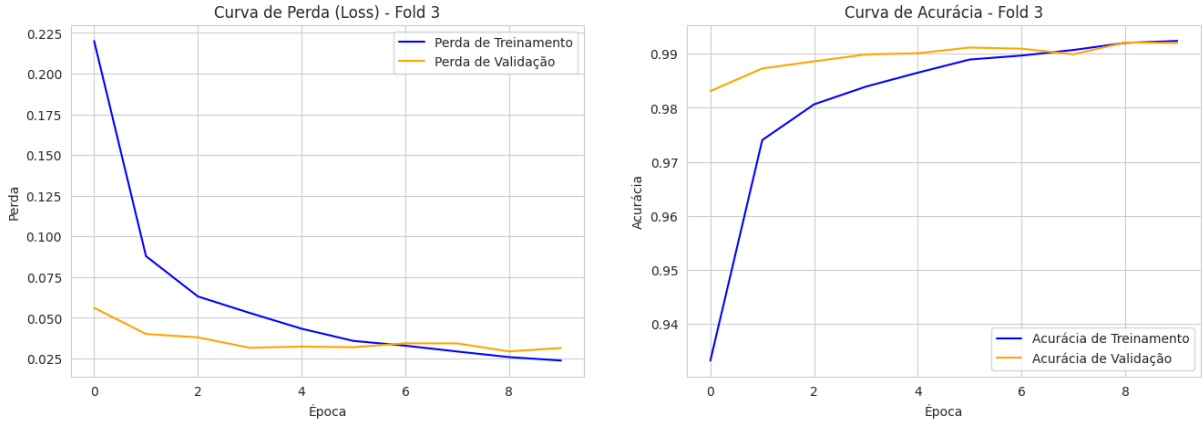
Fonte: Autor (2025).

Figura 3.4: Curvas de aprendizado (perda e acurácia) para o Fold 2 da validação cruzada.



Fonte: Autor (2025).

Figura 3.5: Curvas de aprendizado (perda e acurácia) para o Fold 3 da validação cruzada.



Fonte: Autor (2025).

4. Resultados e Discussão

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

4.1 Desempenho do Modelo Base

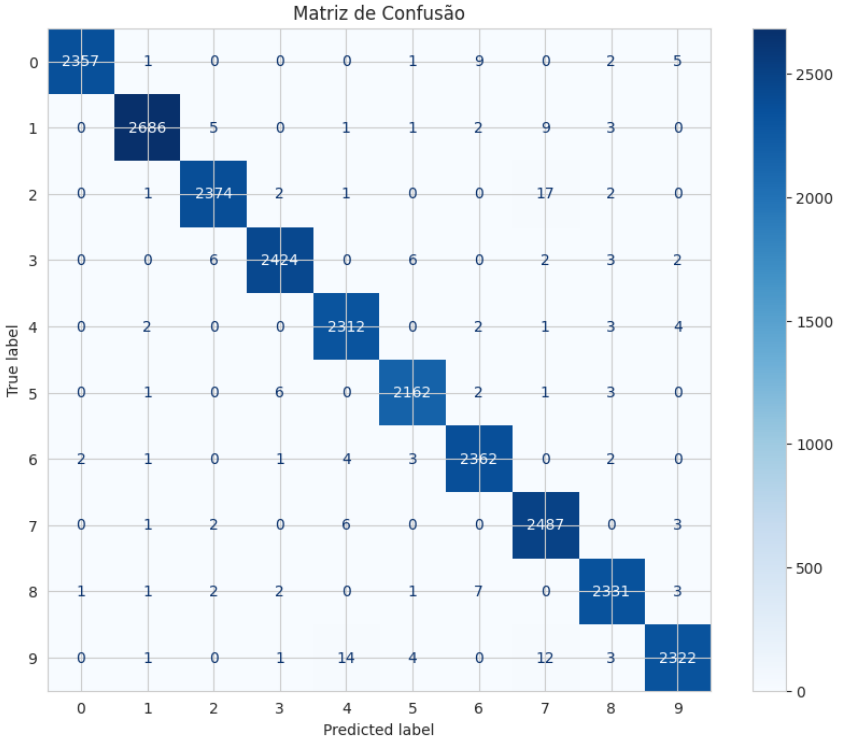
O modelo base, treinado por 15 épocas no conjunto de 96.000 imagens de treino, foi avaliado no conjunto de teste de 24.000 imagens, alcançando uma acurácia final de **99,24%**. O relatório de classificação (Tabela 4.1) detalha as métricas para cada classe, e sua coluna 'support' reflete o número de amostras de teste, totalizando 24.000. A matriz de confusão (Figura 4.1) ilustra a alta performance do modelo.

Tabela 4.1: Relatório de classificação do modelo base no conjunto de teste (24.000 amostras).

	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).

Figura 4.1: Matriz de confusão para o modelo base no conjunto de teste.



Fonte: Autor (2025).

4.2 Análise Comparativa e Contextualização

O processo de otimização com KerasTuner encontrou uma arquitetura que resultou em uma acurácia de **97,55%**, inferior ao modelo base (Tabela 4.2). Este resultado sugere que a busca por hiperparâmetros não foi extensa o suficiente para superar uma arquitetura inicial já bem-sucedida.

Em comparação com o estado da arte (SOTA) para os dados do QMNIST (a fonte original dos dados), que reporta acurácias de até 99,67%, o resultado de 99,24% é altamente competitivo para um único modelo de CNN sem *augmentation* ou *ensembles*.

Tabela 4.2: 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.

Para trabalhos futuros, diversas avenidas podem ser exploradas. A primeira é a aplicação de técnicas de *data augmentation*, como distorções elásticas. Uma segunda etapa seria conduzir uma busca de hiperparâmetros mais ampla. Finalmente, o modelo treinado poderia ser implantado como uma API REST utilizando um microframework como o Flask.

Além disso, a exploração de arquiteturas mais recentes que transcendem as CNNs clássicas, como as *Capsule Networks* [9] ou os *Vision Transformers* [2], 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 traçar um caminho evidente para futuras melhorias e aplicações juntamente com outros projetos mais bem desenvolvidos que foram consultados posteriormente.

Referências Bibliográficas

[1] Kaggle Community. Mnist-120k handwritten digits dataset, 2024. Disponível em plataformas de datasets como o Kaggle. Acesso em: 14 jun. 2025.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.

[7] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] Tom O'Malley et al. Kerastuner. `https://github.com/keras-team/keras-tuner`, 2019.

[9] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems 30*, pages 3856–3866, 2017.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[12] Chhavi Yadav and Léon Bottou. Cold case: The lost mnist digits. In *Advances in Neural Information Processing Systems*, volume 32, 2019.