USP - Universidade de São Paulo

Instituto de Ciências Matemáticas e Computação

Tópicos Avançados em Ciências de Computação I

SCC0910

Projeto - Entrega 2

Alunos - Grupo 8

Nome: Gustavo Vieira Ferreira **N°USP:** 13672690

Nome: Jônatas Alves Lopes **N°USP:** 11796552

Nome: Rafael de Almeida **N°USP:** 11872028

RESUMO

Na primeira parte do projeto utilizou-se o dataset “mnist” para treinar uma rede neural de classificação de dígitos numéricos de 0 a 9 desenhados à mão. E pela simplicidade do banco utilizado, visto que as características dos desenhos eram bem uniformes e as imagens possuíam duas cores apenas, preto e branco, obtivemos um resultado satisfatório de 99% de acurácia ao treinar o modelo com as 60000 imagens do dataset de treinamento. Tendo isso em vista e a fim de continuar a análise proposta e melhorar o baseline apresentado, tem-se como base nessa segunda parte do projeto, a utilização do banco SVHN para classificação de dígitos em imagens do mundo real, especificamente imagens com números de placas em casas advindas do Google Street View. A proposta então será melhorar o modelo de treinamento implementado na primeira parte para maximizar a acurácia de classificação dos dígitos neste outro banco, o SVHN, que apresenta maior diversidade e complexidade nos dados.

INTRODUÇÃO

As redes neurais convolucionais (CNN) são redes neurais especializadas principalmente na análise de imagens digitais. Esse processo é realizado por meio de camadas convolucionais, nas quais operadores de convolução são aplicados de forma sucessiva para extrair características da imagem. Para este trabalho, semelhante à primeira parte, foi desenvolvida uma CNN para a classificação de dígitos de imagens, neste caso em imagens do mundo real. E além das camadas de convolução foram utilizadas na rede as camadas de Batch Normalization, de Droupout e MaxPooling.

Este projeto é relativamente mais complexo em relação ao anterior, que se baseava no banco MNIST, se tratando de um problema mais aplicável no mundo real e ainda aberto para melhoras no meio de pesquisa de redes neurais.

Tendo isso em vista, foi inicialmente feito um levantamento de artigos, publicações e páginas de pessoas que trabalharam previamente com este dataset. Grande parte dos artigos analisados trazem informações de resultados para diferentes modelos de redes aplicadas no SVHN e comparando também com outros datasets como o CIFAR-10, CIFAR-100, Stanford Cars e ImageNet. Em um dos artigos, denominado “AutoAugment: Learning Augmentation Strategies from Data”, o dataset SVHN foi treinado com o modelo Wide-ResNet-28-10 e no modelo Shake-Shake em 160 epochs, conseguindo uma taxa de erro de apenas 1,5% e 1,4% respectivamente.

Dataset	Model	Baseline	Cutout [12]	AutoAugment
CIFAR-10	Wide-ResNet-28-10 [67]	3.9	3.1	2.6±0.1
	Shake-Shake (26 2x32d) [17]	3.6	3.0	2.5±0.1
	Shake-Shake (26 2x96d) [17]	2.9	2.6	2.0±0.1
	Shake-Shake (26 2x112d) [17]	2.8	2.6	1.9±0.1
	AmoebaNet-B (6,128) [48]	3.0	2.1	1.8±0.1
	PyramidNet+ShakeDrop [65]	2.7	2.3	**1.5 ± 0.1**
Reduced CIFAR-10	Wide-ResNet-28-10 [67]	18.8	16.5	14.1±0.3
	Shake-Shake (26 2x96d) [17]	17.1	13.4	**10.0 ± 0.2**
CIFAR-100	Wide-ResNet-28-10 [67]	18.8	18.4	17.1±0.3
	Shake-Shake (26 2x96d) [17]	17.1	16.0	14.3±0.2
	PyramidNet+ShakeDrop [65]	14.0	12.2	**10.7 ± 0.2**
SVHN	Wide-ResNet-28-10 [67]	1.5	1.3	1.1
	Shake-Shake (26 2x96d) [17]	1.4	1.2	**1.0**
Reduced SVHN	Wide-ResNet-28-10 [67]	13.2	32.5	8.2
	Shake-Shake (26 2x96d) [17]	12.3	24.2	**5.9**

Figura 1- Taxa de erro (%) para diferentes modelos em diferentes datasets com ou sem Cutout e AutoAugment. [1]

No artigo “Reading Digits in Natural Images with Unsupervised Feature Learning”, disponível na própria página do dataset SVHN, existe a informação da acurácia do teste no SVHN para diferentes algoritmos de classificação dos dígitos do banco, incluindo também a performance humana que possui acurácia de 98%, ainda superior à dos algoritmos propostos no artigo.

ALGORITHM	SVHN-TEST (ACCURACY)
HOG	85.0%
BINARY FEATURES (WDCH)	63.3%
K-MEANS	90.6%
STACKED SPARSE AUTO-ENCODERS	89.7%
HUMAN PERFORMANCE	98.0%

Figura 2- Acurácia no SVHN-test. [5]

Código fonte disponível no github: [Github-Projeto]

DADOS

O SVHN é um dataset de imagens do mundo real para o estudo e desenvolvimento de algoritmos de aprendizado de máquina e reconhecimento de objetos, a partir de requisitos mínimos de pré-processamento e formatação dos dados. Este pode ser considerado

bem semelhante ao MNIST, visto que as imagens apresentadas também são de pequenos dígitos recortados, mas incorpora uma ordem de magnitude bem maior, pois tem muito mais dados rotulados (mais de 600.000 imagens de dígitos) e vem de um problema do mundo real significativamente mais difícil e não exatamente resolvido, reconhecer dígitos e números em imagens de cenas naturais. O SVHN é obtido a partir dos números das casas nas imagens do Google Street View. [4]

A classificação é em 10 classes, uma para cada dígito. O dígito ‘1’ tem label 1, o dígito ‘2’ tem label 2 e assim respectivamente, salvo que o dígito ‘0’ possui o label 10.

Ao todo no dataset são 73257 imagens de dígitos para treinamento, 26032 dígitos para o teste, e 531131 adicionais, que são amostras menos difíceis de se classificar, para serem utilizadas como treinamento adicional.

É possível utilizar o dataset em dois formatos diferentes, um que possui as imagens originais com tamanhos variáveis e com parâmetros de posição, altura e largura para caixas que demarcam os dígitos.

Figura 3 - Imagens do dataset SVHN no primeiro formato (as caixas em azul são apenas ilustrativas e não se encontram assim nas imagens do dataset).

E o outro formato é um pouco mais parecido com a forma do dataset MNIST, possuindo imagens de tamanho fixo de 32x32 pixels e centralizadas em um único carácter cada. A classificação da imagem se refere sempre ao carácter no centro, mesmo que existam outros em volta. Esse segundo formato foi o utilizado para o treino da rede neste projeto, visto que se assemelha mais ao que foi feito anteriormente.

Figura 4 - Imagens do dataset SVHN no segundo formato (32x32).

METODOLOGIA

A partir disso, construímos uma das redes encontradas para aprender a classificar digitos com o SVHN. A segunda parte do projeto também foi implementada utilizando notebook e na linguagem Python com auxílio da plataforma do Google Colaboratory.

Utilizou-se as seguintes bibliotecas do python:

- Numpy: Biblioteca especializada em manipulações numéricas e de arrays.
- Matplotlib: O módulo “Pyplot” foi utilizado para construção dos gráficos e visualização das imagens.
- TensorFlow: Biblioteca especializada em aprendizado de máquina, no qual o módulo “Keras” foi utilizado para construção da rede neural.
- SciPy: expande o NumPy ao oferecer funcionalidades adicionais e ferramentas especializadas para computação científica.
- Scikit-learn: Oferece um conjunto rico de ferramentas e algoritmos para várias tarefas de aprendizado de máquina.

O dataset SVHN foi carregado do Google Drive em formato de arquivo do MATLAB usando a função loadmat da biblioteca SciPy e a conexão com o drive do Colab. Os arquivos carregados foram “train_32x32.mat”, que possui as 73257 imagens de tamanho 32x32 pixels e os labels para treinamento, e “test_32x32.mat” que possui 26032 imagens e labels para os testes.

Com o pré-processamento, as imagens são normalizadas e os labels são binarizados.

A estrutura do modelo consiste em uma camada de convolução de 32 filtros com tamanho de kernel 3 por 3, mantendo o mesmo tamanho da entrada. Essa camada usa a função de ativação ReLU para introduzir não-linearidade e é aplicada a uma entrada com formato de (32, 32, 3), representando uma imagem colorida de 32 por 32 pixels.

Em seguida, é adicionada uma camada de normalização em lote (Batch Normalization) que normaliza os valores de saída da camada anterior, tornando-os mais robustos e acelerando o treinamento da rede.

Outra camada de convolução com 32 filtros e tamanho de kernel 3 por 3 é adicionada, novamente com ativação ReLU. Essa camada é seguida por uma camada de MaxPooling de tamanho (2, 2), que reduz a dimensionalidade da saída pela metade, preservando as características mais importantes.

Uma camada de Dropout com taxa de 0.3 é inserida após o MaxPooling. O Dropout desativa aleatoriamente uma porcentagem dos neurônios durante o treinamento, o que ajuda a evitar o overfitting, fazendo com que a rede se generalize melhor para novos dados.

Esse padrão de camadas de convolução, normalização em lote, convolução, MaxPooling e Dropout se repete mais duas vezes, com a segunda camada de convolução tendo 64 filtros e a terceira camada de convolução tendo 128 filtros. Cada repetição busca extrair características mais complexas das imagens.

Após a última camada de Dropout, a saída é achatada (Flatten) em um vetor unidimensional para ser processada por camadas totalmente conectadas. A primeira camada densa possui 128 neurônios com ativação ReLU, seguida por uma camada de Dropout com taxa de 0.4.

Finalmente, a última camada densa possui 10 neurônios com ativação softmax, que atribui probabilidades a cada uma das 10 classes possíveis de classificação. A classe com a maior probabilidade é escolhida como a predição final do modelo.

RESULTADOS

Após um treino de 10 *epochs* do modelo com 73257 imagens de treino e 26032 de teste, foi obtido um resultado (no melhor caso) de:

- Perdas no teste 26.26%
- Acurácia no teste 93.92%

e no final de:

- Perdas no teste 38.23%
- Acurácia no teste 88.52%

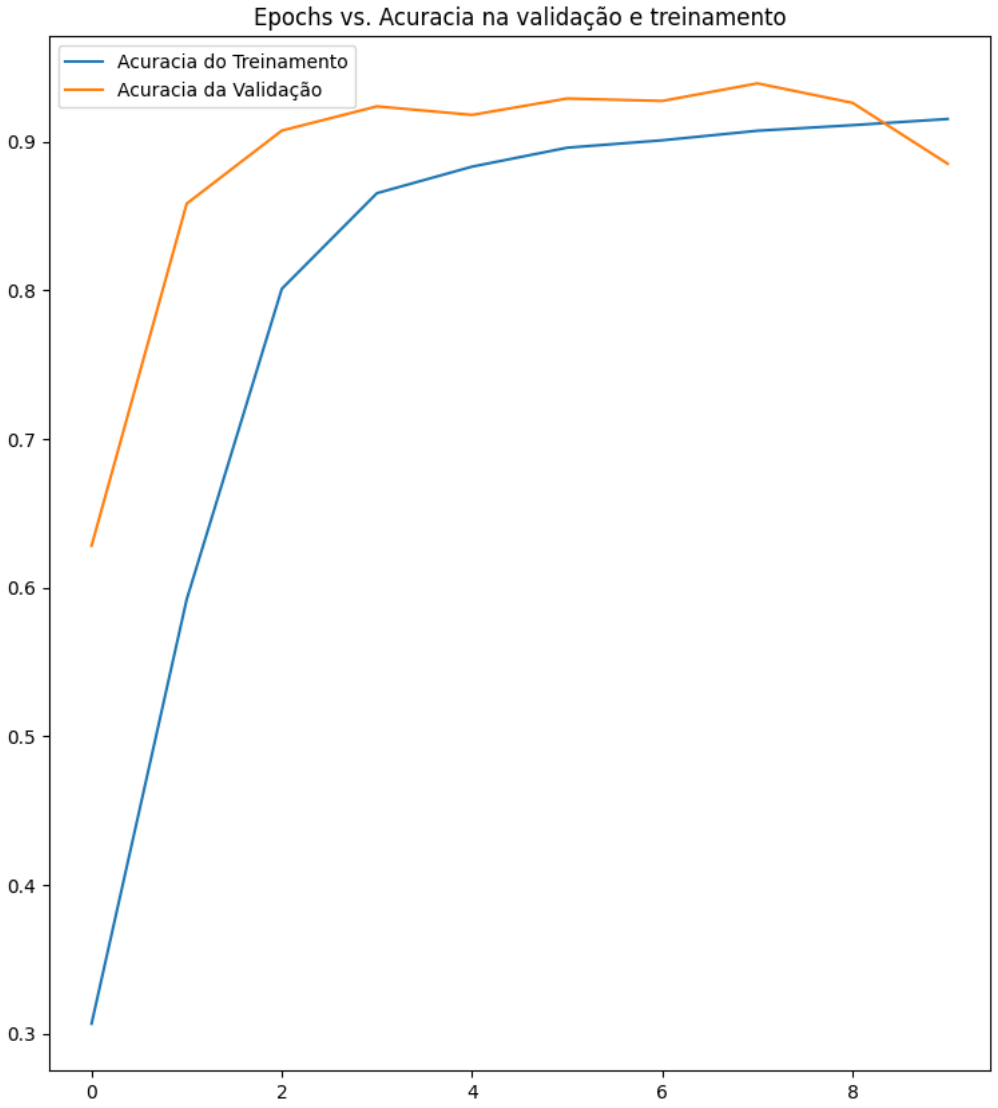

Figura 5 - Acurácia em cada epoch do treinamento com 10 epochs.

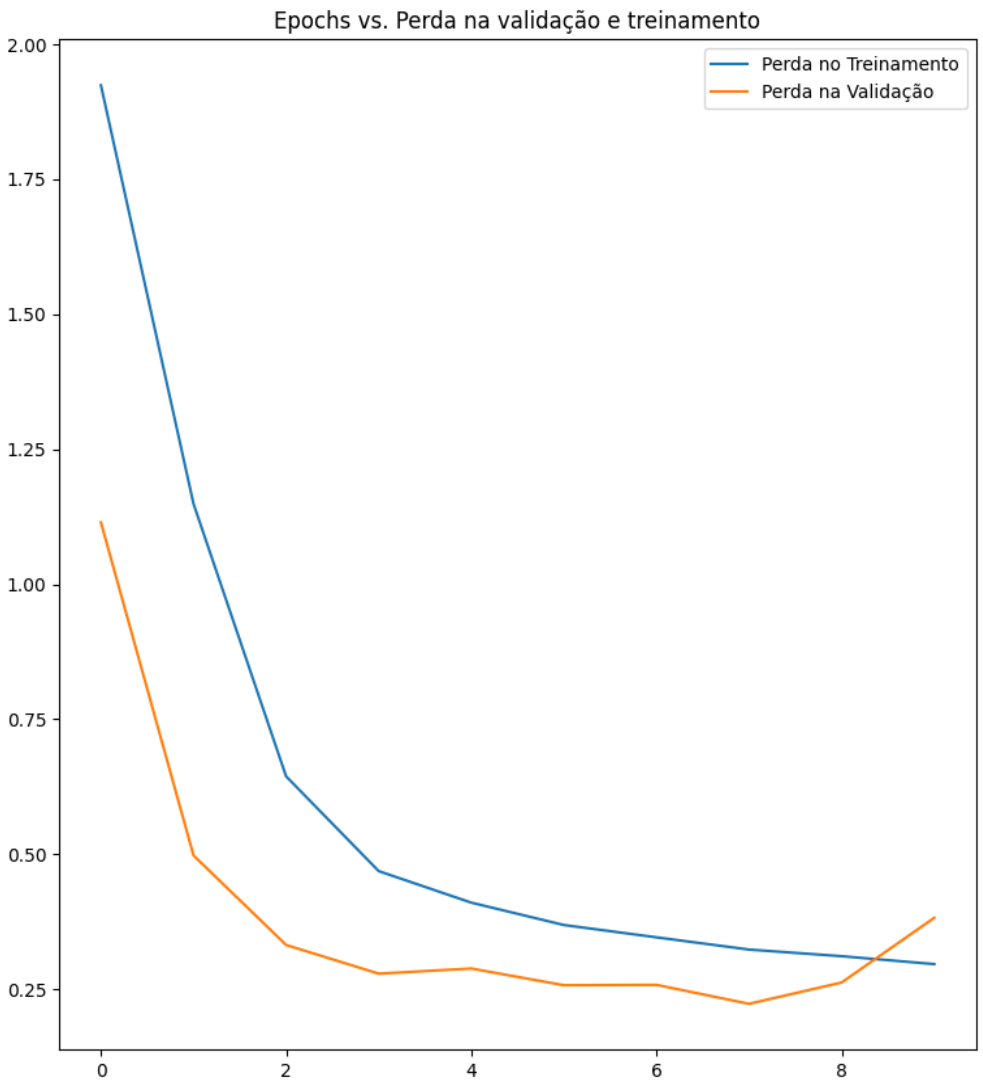

Figura 6 - Taxa de erro em cada epoch do treinamento com 10 epochs.

Após um treino de 46 *epochs* do modelo com apenas 6000 imagens de treino e 1000 de teste, foi obtido um resultado de:

- Perdas no teste 36.14%
- Acurácia no teste 91%

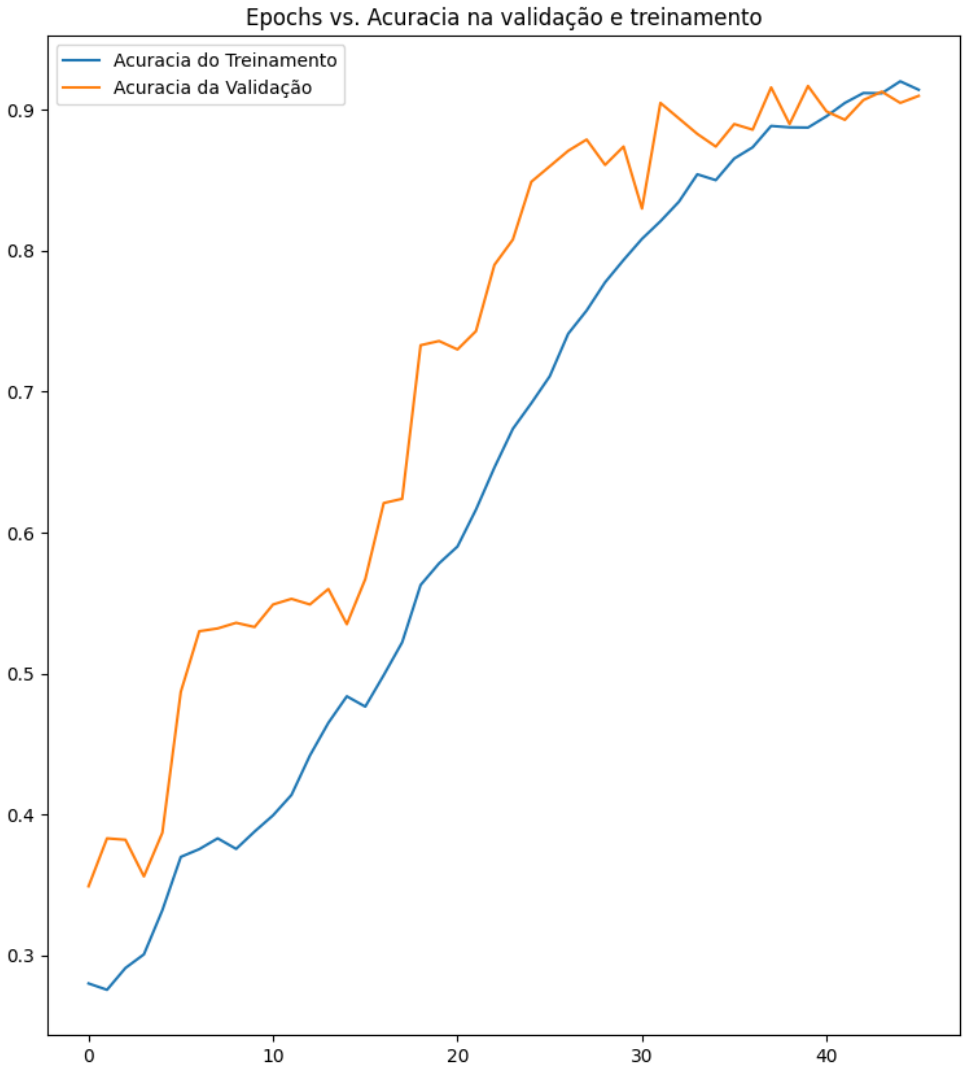

Figura 7 - Acurácia em cada epoch do treinamento com 46 epochs.

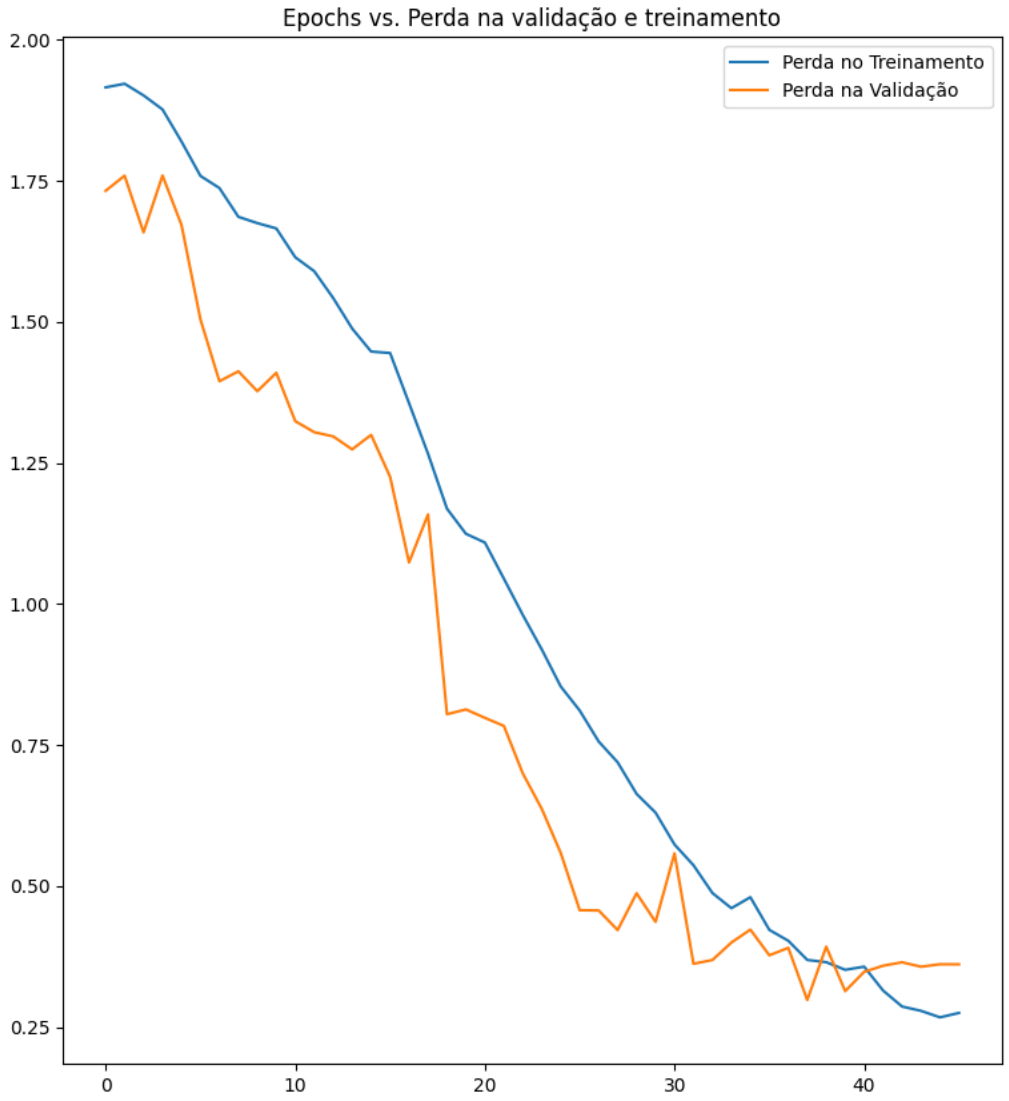

Figura 8 - Taxa de erro em cada epoch do treinamento com 46 epochs.

CONCLUSÃO

Por fim, o modelo alcançou uma acurácia de teste de 93.92%, e portanto não demonstrando melhorias em relação ao modelo treinado no MNIST. Isso indica que a abordagem adotada foi eficaz para lidar com a diversidade e complexidade dos dados do SVHN, embora não com a mesma eficácia do problema mais simples de outrora.

REFERÊNCIAS

[1] Ekin D. Cubuk, Barret Zoph, Dandelion Man´e, Vijay Vasudevan, Quoc V. Le. **AutoAugment: Learning Augmentation Strategies from Data.**
Google Brain. Disponível em: <https://arxiv.org/pdf/1805.09501.pdf>

[2] Dimitrios Roussis. **SVHN Classification with CNN (Keras - ~96% Acc).** Disponível em: <https://www.kaggle.com/code/dimitriosroussis/svhn-classification-with-cnn-keras-96-acc>

[3] LIAO, Zhibin; CARNEIRO, Gustavo. **Competitive Multi-scale Convolution.** Disponível em: <https://arxiv.org/abs/1511.05635>

[4] **The Street View House Numbers (SVHN) Dataset.** Disponível em: <http://ufldl.stanford.edu/housenumbers/>

[5] NETZEL, Yuval; WANG,Tao; COATES, Adam; BISSACCO, Alessandro; WU, Bo; NG, Andrew Y. **Reading Digits in Natural Images with Unsupervised Feature Learning.** Disponível em: <http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf>