Instituto Tecnológico de Aeronáutica

CT-213

Laboratório 8 - Redes Neurais Convolucionais

Aluno: Pedro Elardenberg Sousa e Souza
Professor: Marcos Ricardo Omena de Albuquerque Maximo

7 de janeiro de 2024

Conteúdo

1 Resumo

Neste Laboratório, foi criada uma rede neural LeNet-5 usando o *dataset* MNIST. O MNIST consiste num conjunto grande de imagens anotadas de dígitos decimais escritos à mão. Portanto, esta atividade reproduz um trabalho clássico da Literatura de Redes Neurais Convolucionais (CNNs), que foi realizado originalmente por Yann LeCun.

Após se implementar a rede neural, foi executado um *script* que realiza o treinamento da rede neural. Com a ferramenta *Tensorboard*, foi possível vizualizar a evolução do treinamento da rede.

Por fim, foi executado um *script* de validação da rede, pelo qual foram obtidas amostras aleatórias de casos em que o algoritmo teve resultados corretos e incorretos.

Os resultados foram então mostrados e discutidos neste relatório.

2 Introdução

Muitas vezes a análise e *debugging* de uma rede neural se torna muito difícil pela quantidade de parâmetros e iterações do algoritmo, então uma forma de fazer isso é por meio de uma interface gráfica. O Tensorboard é uma ferramenta do *framework* Tensorflow que permite visualizar o andamento e desempenho de uma rede neural. Ele permite rastrear métricas de experimentos como perda e precisão, visualizar o gráfico do modelo, projetar *embeddings* para um espaço dimensional inferior e muito mais[2]. A figura 1 mostra o painel de gráficos escalares do Tensorboard para este laboratório.

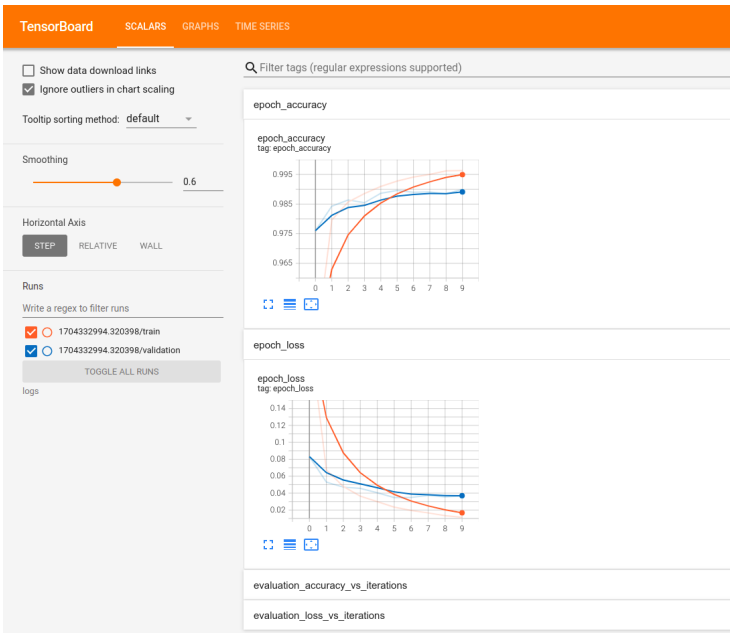

Figura 1: Painel inicial do Tensorboard para a aplicação da rede Lenet-5 deste Laboratório

2.1 Rede Lenet-5

Lenet-5 é uma arquitetura de rede neural utilizada para reconhecimento de caracteres escritos à mão. Ela funciona como uma rede neural convolucional de várias camadas para classificação de imagens[1].

A rede possui 5 camadas com hiperparâmetros, com 3 conjuntos de camadas convolucionais seguidas de uma camada de *average pooling*. Depois dessas camadas, há duas camadas totalmente conectadas. Por fim, um classificador Softmax classifica as imagens às suas respectivas classes.

A entrada do modelo é uma imagem 32×32 em escala de cinza, portanto o número de canais é 1.

Existem várias implementações dessa rede. A utilizada neste Laboratório é a seguinte: Após a primeira camada, é aplicada a convolução com o filtro tamanho 5×5 para 6 filtros. Após o primeiro *pooling*, a operação reduz o tamanho do mapa para $14 \times 14 \times 6$. Em seguida, os passos anteriores de convolução e *pooling* são repetidos e o tamanho do mapa é reduzido para $5 \times 5 \times 16$.

Com uma convolução posterior, o tamanho do modelo passa para $1 \times 1 \times 120$. Com uma camada do tipo totalmente conectada e por fim uma camada de saída com 10 classes, correspondentes aos 10 algarismos que se deseja classificar. A figura 2 mostra essa arquitetura.

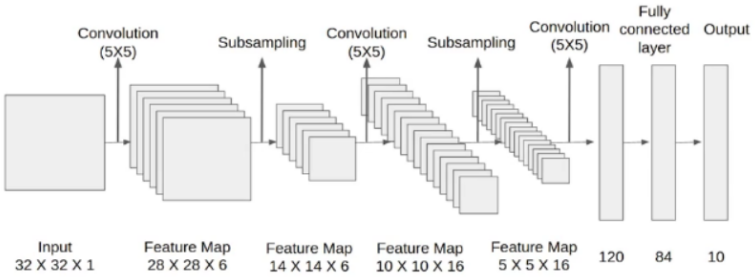

Figura 2: Arquitetura final do modelo Lenet-5[1]

3 Análise dos Resultados

Neste Laboratório, foi implementada no código **lenet5.py** a rede neural Lenet-5, com a seguinte arquiterura:

Tabela 1: Arquitetura da Lenet-5 para este Laboratório

# Camada	**Tipo**	**# Filtros**	**Saída**	**Kernel**	**Stride**	**Ativação**
Entrada	Imagem	1	32×32	-	-	-
1	Conv2D	6	28×28	5×5	1	tanh
2	AveragePooling2D	6	14×14	2×2	2	-
3	Conv2D	16	10×10	5×5	1	tanh
4	AveragePooling2D	16	5×5	2×2	2	-
5	Conv2D	120	1×1	5×5	1	tanh
6	Dense (FC)	-	84	-	-	tanh
7	Dense (FC)	-	10	-	-	softmax

Depois de ter baixado as imagens do MNIST e implementado **lenet5.py**, foi realizado o treinamento da rede através do *script* **train_lenet5.py**. Esse *script* separa o *training set* original, deixando algumas imagens para o *validation set*. Após executar e compilar a rede Lenet-5, as imagens do conjunto de treinamento e de validação são mapeadas e passadas para o Tensorboard, que mostra, entre outras informações, os gráficos que demonstram o aprendizado da rede. A figura 3 mostra a precisão, isto é, qual a parcela das imagens de treinamento e de validação que correspondem à predição do modelo. Semelhantemente, a figura 4 mostra a perda das imagens desses conjuntos.

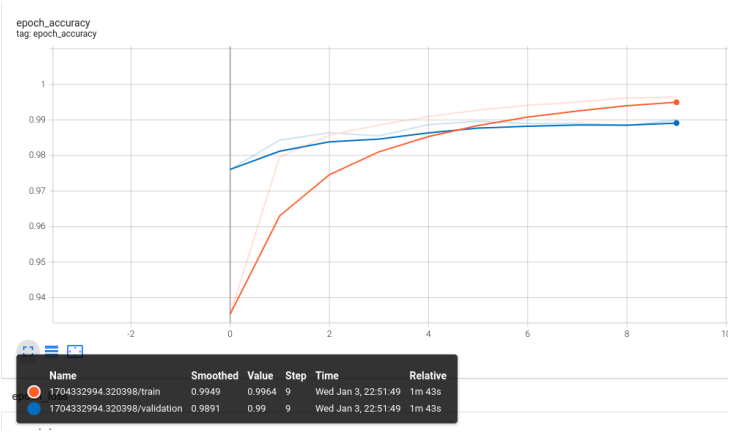

Figura 3: Precisão dos conjuntos de treinamento e de validação para cada *epoch* do treinamento da rede

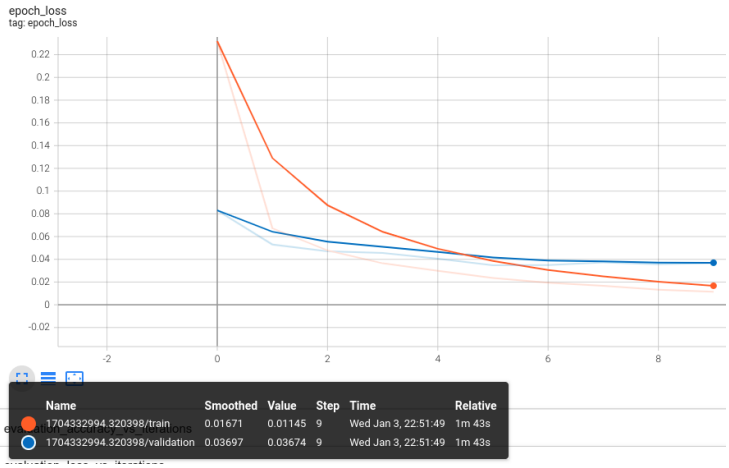

Figura 4: Perda dos conjuntos de treinamento e de validação para cada *epoch* do treinamento da rede

Da figura 3, vê-se que a rede predisse corretamente 99.64% dos exemplos de treinamento e 99% dos exemplos de validação, de um total de 48000 imagens de treinamento.

Em seguida, foi executado o *script evaluate_lenet5.py*. Esse *script* toma os dados calculados pelo *script* anterior e mostra exemplos de imagens que foram corretamente classificadas pela rede e imagens que não o foram. As figuras 6 e 5 são dois desses exemplos.

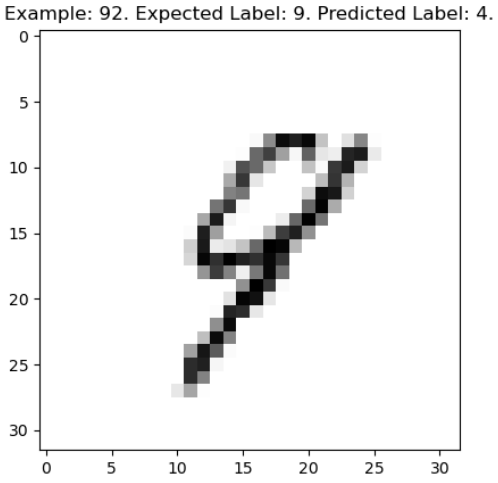

Figura 5: Imagem classificada incorretamente pela rede

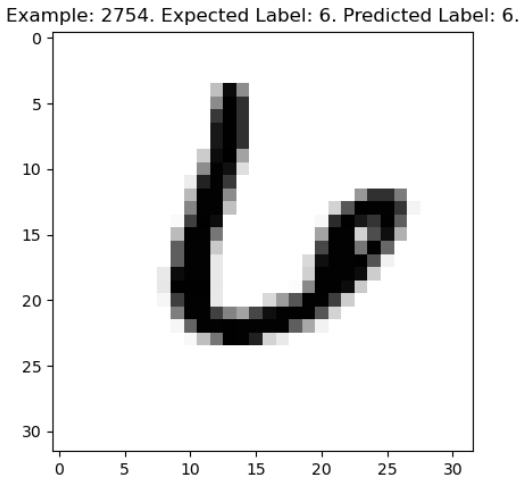

Figura 6: Imagem classificada corretamente pela rede

4 Conclusão

Dos exemplos mostrados, bem como os outros que o código retornou, é possível perceber que o algoritmo Lenet-5 é capaz de detectar caracteres de uma forma bem ampla e com alta precisão, com diferentes caligrafias e para exemplos não tão claros, como o caso do "6"da figura 6. As imagens incorretamente classificadas assemelham-se ao predito pela rede. O "9"da figura 5 possui semelhança com o "4"incorretamente classificado. O mesmo se observa para outros casos em que a rede não prediz corretamente a imagem.

Referências

[1] The architecture of lenet-5. Acessado em 07 de Janeiro de 2024. URL: `https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/`.

[2] Tensorboard overview. Acessado em 06 de Janeiro de 2024. URL: `https://www.tensorflow.org/tensorboard/get_started?hl=pt-br`.