Aprendizado de Máquina
18 de novembro de 2025

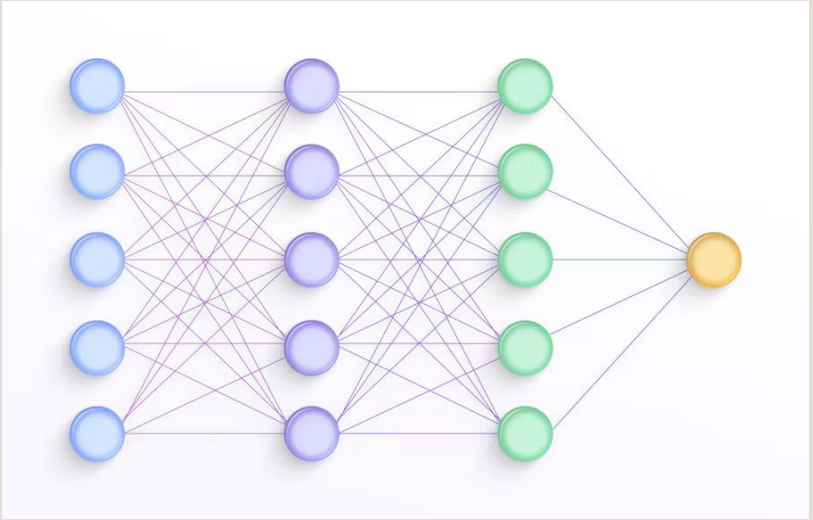
Trabalho 2:
Redes Neurais Profundas

Guilherme Malgarizi Vásquez, Henrique Zapella Rocha, Pedro Mezacasa Muller

Contents

Referencial Teórico

O objetivo do projeto é classificar sinais mão em três classes: “down”, “ok” e “up”. Utilizamos uma arquitetura de rede neural convolucional. As camadas convolucionais conseguem extrair características das imagens pelos kernels que deslizam sobre a imagem original. Utilizamos a função de ativação ReLU para quebrar a linearidade. Essa função zera valores negativos e mantém os positivos iguais. As operações são seguidas de operações de *pooling* para realizar a diminuição da dimensão. Ao final, usamos camadas totalmente conectadas para fazer a classificação.



Tecnologias

PyTorch

Utilizamos o PyTorch para criar nossa rede neural e montar o dataset com os dados em formato de tensors para usar com o modelo.

Scikit-learn

Usamos o scikit para ter acesso ao *train_test_split* para separar o dataset em partes para treinamento do modelo e teste do modelo.

Pandas

Usamos o pandas para fazer a leitura dos arquivos csv que guardam as informações das classes de cada imagem do dataset.

Matplotlib & Seaborn

Utilizamos essas duas bibliotecas para montar os gráficos apresentados no notebook e mostrados na apresentação.

Desenvolvimento

```python
class ConvNeuralNet(nn.Module):
    def __init__(self):
        super().__init__()

        self.conv1 = nn.Conv2d(3, 24, kernel_size=5) # (24, 60, 60)
        self.pool  = nn.MaxPool2d(2, 2) # (24, 30, 30)
        self.conv2 = nn.Conv2d(24, 48, kernel_size=5) # (48, 26, 26) → (48, 13, 13) → Flatten (48 * 13 * 13)
        self.fc1   = nn.Linear(48 * 13 * 13, 120)
        self.fc2   = nn.Linear(120, 84)
        self.fc3   = nn.Linear(84, 3)

    def forward(self, x):
        x = self.pool(nn.functional.relu(self.conv1(x)))
        x = self.pool(nn.functional.relu(self.conv2(x)))
        x = torch.flatten(x, 1)
        x = nn.functional.relu(self.fc1(x))
        x = nn.functional.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

A arquitetura da rede foi criada com apoio de vídeos. Aproveitamos a organização das camadas de convolução e totalmente conectadas. Nosso modelo se diferencia nos parâmetros que passamos para criar cada camada já que isso depende diretamente dos nossos dados. Começamos com 3 entradas, 1 para representar cada canal de cor da imagem. Os valores de *features* foram escolhidos na base de teste juntamente com o *kernel*. Os comentários no código colocam o passo a passo para criarmos as camadas seguintes. Para a camada conectada, nos baseamos na *LeNet* que utiliza 120, 84 e 10 saídas. Nesse caso, alteramos a saída de 10 para 3 já que temos apenas 3 classes *target*.

Coleta de Dados

- **300 imagens**
  - 100 por classe
    - “Up”
    - “Down”
    - “Ok”
- **Fotos tiradas em proporção 1:1**
- **Diferentes**
  - Níveis de luz
  - Sujeitos
- **Arquivo csv**
  - Nome do arquivo
  - Classe

“Up”

“Down”

“Ok”

Pré-processamento

- **Resize**
  - 3024x3024 -> 64x64
- **Grayscale**
  - Resultado ficou pior

“Up”

“Down”

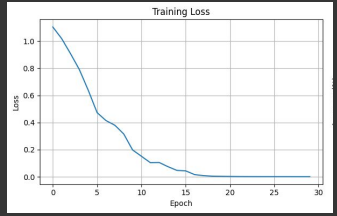
“Ok”

Treinamento

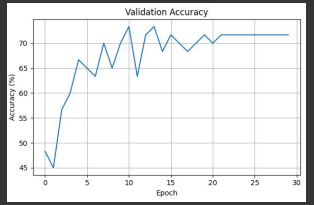
```
Epoch 5/30 | Loss: 0.6377 | Val Acc: 66.67%
Epoch 10/30 | Loss: 0.1979 | Val Acc: 70.00%
Epoch 15/30 | Loss: 0.0469 | Val Acc: 68.33%
Epoch 20/30 | Loss: 0.0031 | Val Acc: 71.67%
Epoch 25/30 | Loss: 0.0008 | Val Acc: 71.67%
Epoch 30/30 | Loss: 0.0005 | Val Acc: 71.67%

Final Training Loss: 0.0005
Final Validation Accuracy: 71.67%
```

Treinamos o modelo usando 30 epochs. Usamos batch_size de 32. Usamos máquina local para treinar e testar (Apple M2).

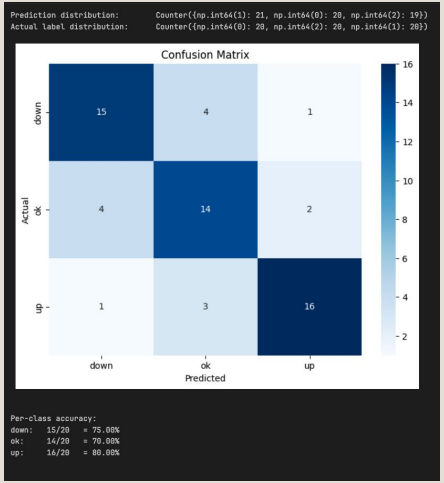


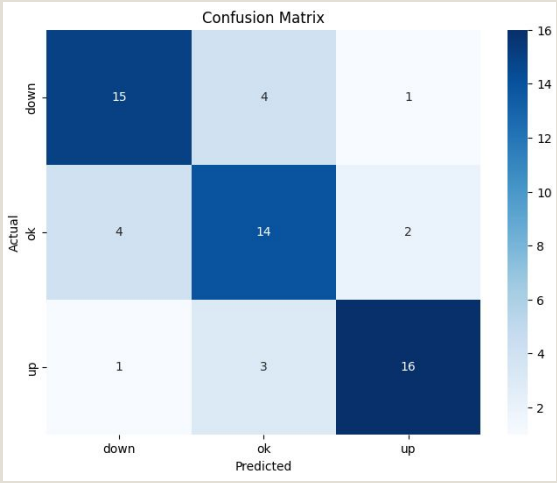
No final do treinamento, geramos um gráfico para ver a progressão do *loss* da rede. Conseguimos ver que a rede chega próxima de uma assíntota e converge.



Também construímos um gráfico da acurácia do modelo com os dados de teste durante o treinamento. Podemos acompanhar e ver que a acurácia se estabiliza junto com o *loss*.

Avaliação





Avaliamos a performance do modelo nos dados de teste e percebemos uma melhor performance em acertar o up, com 80%. Mas os valores absolutos ainda são próximos para ter alguma significância.

Resultados e Conclusões

Modelo final ficou com acurácia de 73.33% nos dados de validação que a rede não conhecia ainda. Ela acertou 11 dos 15 dados.

Um ponto importante sobre nosso trabalho foi o tamanho do dataset e a variedade de imagens. Mesmo fazendo algumas mudanças nos parâmetros da rede ainda foi difícil conseguir extrair mais performance do modelo sendo treinado do zero.

Se usarmos um modelo pré treinado e apenas fizermos ajustes nele para nosso dataset é possível que possamos conseguir uma performance melhor sem resultar em *overfitting*.

```
Image: IMG_2937.jpeg    Label: up       Prediction: up
Image: IMG_2938.jpeg    Label: down     Prediction: down
Image: IMG_2939.jpeg    Label: ok       Prediction: ok
Image: IMG_2940.jpeg    Label: down     Prediction: ok
Image: IMG_2941.jpeg    Label: down     Prediction: down
Image: IMG_2942.jpeg    Label: down     Prediction: down
Image: IMG_2943.jpeg    Label: down     Prediction: down
Image: IMG_2944.jpeg    Label: ok       Prediction: ok
Image: IMG_2945.jpeg    Label: ok       Prediction: ok
Image: IMG_2946.jpeg    Label: ok       Prediction: ok
Image: IMG_2947.jpeg    Label: ok       Prediction: ok
Image: IMG_2948.jpeg    Label: up       Prediction: ok
Image: IMG_2949.jpeg    Label: up       Prediction: ok
Image: IMG_2950.jpeg    Label: up       Prediction: down
Image: IMG_2951.jpeg    Label: up       Prediction: up

Evaluation Accuracy: 73.33% (11/15)
```

Apoio

Image Classification CNN in PyTorch. YouTube, NeuralNine, 6 Aug 2024, youtube.com/watch?v=CtzfbUwrYGI.

Image Classifier in PyTorch. YouTube, Kie Codes, 19 Mar 2024, youtube.com/watch?v=igQeI29FIQM.

PyTorch. *Build the Neural Network*. 24 Jan 2025, docs.pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html#further-reading.

PyTorch. *Datasets & DataLoaders*. 24 Sep 2025, docs.pytorch.org/tutorials/beginner/basics/data_tutorial.html.

PyTorch Project: Handwritten Digit Recognition. YouTube, NeuralNine, 22 Aug 2023, youtube.com/watch?v=vBlO87ZAiiw.

Zhang, A. et al. *Dive into Deep Learning*. Cambridge University Press. 7 Dec 2023. https://d2l.ai/chapter_convolutional-neural-networks/lenet.html

Obrigado