SCC0910 - Tópicos Avançados em Ciências de Computação I

Relatório 1

Leonardo Vinicius de Almeida
Luccas Stroppa Saliba Barizon
Matheus Vellosa de Andrade

Acesso ao código do projeto:
https://colab.research.google.com/drive/1SvHBZZS9rBJMJ1uY1P-qvxcAgu0duHg7?authuser=1#scrollTo=q9mf52gLEi9V

1. Problemática abordada e motivação

A problemática abordada pelo grupo no projeto foi a identificação e diferenciação de placas de trânsito, utilizando, como base, dados de placas de trânsito americanas, como será explicado no tópico seguinte. O tema foi selecionado por conta do potencial de impacto prático, como a utilização para carros autônomos e para auxílio de pedestres que possuam algum tipo de deficiência visual.

2. Referências bibliográficas

1. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9718991/
2. https://repositorio.ufscar.br/bitstream/handle/ufscar/15692/Monografia_Final_Gustavo_Bulka_B_Bademian_744593.pdf?sequence=1&isAllowed=y
3. https://repositorio.ufsc.br/bitstream/handle/123456789/192159/DanielaPreto_TCCVersaoFINAL.pdf?sequence=1&isAllowed=y
4. https://www.kaggle.com/datasets/andrewmvd/road-sign-detection

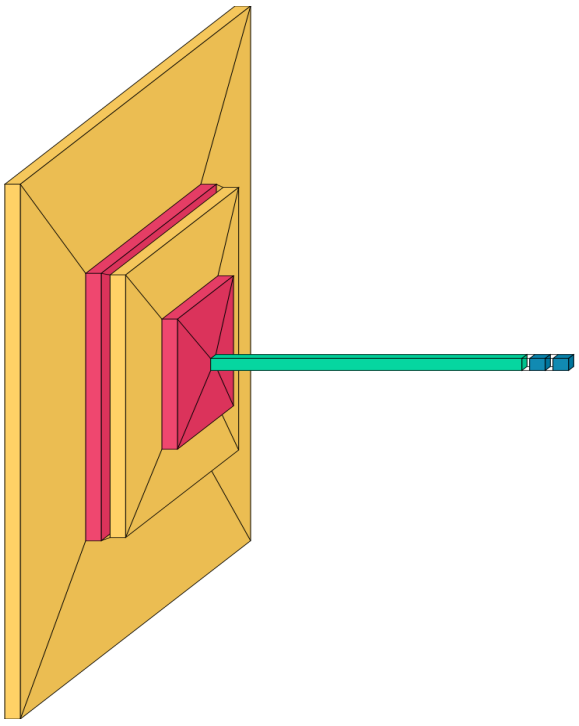
- Observações:
    - Devido à falta de material brasileiro no tema, o grupo escolheu treinar a rede com base nos dados de placas de trânsito americanas[4].
    - O dataset utilizado se encontra na referência de número 4 para realização de testes, caso necessário.

3. Análise dos dados

- 877 imagens, classificadas;
- Classes: ”Trafic Light”, “Stop”, “Speedlimit” e “Crosswalk”;
- Observações:
    - Poucas classes, e presença de classes bem distintas (por exemplo placas e semáforos), por isso um alto desempenho do modelo é esperado;
    - Número relativamente baixo de imagens.

4. Implementação

- O código foi desenvolvido em Python, utilizando principalmente as bibliotecas: Keras e TensorFlow;
- Dados para aprendizado e teste separados em imagem e xml classificando as imagens;
- Arquitetura da CNN;
    - Estrutura do modelo baseada em heurística do grupo (observando outros modelos);
        - Por se tratar do modelo inicial, a arquitetura foi tratada como ponto a ser melhorado futuramente.
    - Estrutura:
        1. Convolucional com 32 filtros de 3x3
        2. Camada de pooling (tamanho 2x2)
        3. Segunda camada convolucional com 64 filtros de 3x3
        4. Segunda camada de pooling
        5. Camada flatten
        6. Camada densa (128 neurônios)
        7. Camada de saída com 4 classes e ativação softmax

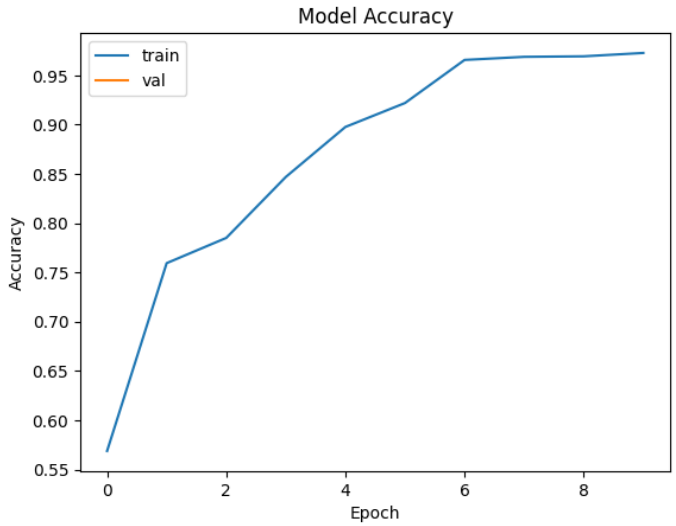


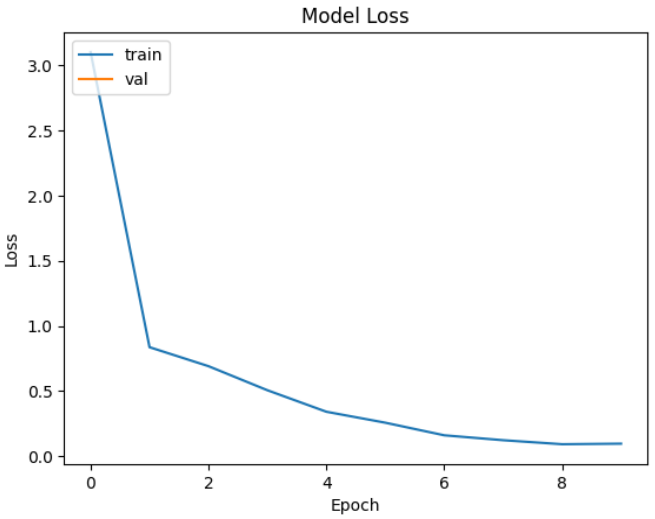
5. Treinamento

- 10 epochs de 10 steps cada;
- Batch size: 46;
- 70% dos dados para aprendizado em 30% para teste;
- Aproximadamente 30 segundos por epoch.

6. Resultados

- Após realizado o treinamento do modelo, a acurácia passou de 0.56 na primeira epoch para 0.97 na última, além disso, o erro foi de 3.1019 para 0.0969;





- Ao realizar o teste do modelo, a acurácia obtida foi de 0.7765 e o erro foi de 0.95.

7. Análise dos Resultados

- O resultado obtido é menor do que o esperado, e definitivamente pode ser melhorado.
- Suspeita dos motivos dos resultados inferiores:
    - Escolha dos dados:
        - Várias das fotos são tiradas de longe e algumas vezes apresentam elementos de mais de uma classe (por exemplo com uma faixa de pedestre e uma placa de pare);
    - Overfitting:
        - Acurácia do modelo caiu muito no teste em relação ao treino;
- Como superar o modelo atual:
    - Mudanças na arquitetura:
        - Como observado por Daniela[2], neste caso, redes mais simples podem alcançar maior acurácia em menos tempo de treino.
            - Além disso, simplificar a arquitetura pode melhorar o problema de overfitting.

        - Utilizar uma arquitetura já testada ao invés de construir o modelo do zero.

    - Transfer Learning:
        - Treinando modelos com os dados de placas alemãs[1] e depois retreinando nas placas americanas, podemos obter um melhor resultado.
    - Maior número de épocas:
        - Loss estava em processo de diminuir e na última época aumentou, portanto mais épocas poderiam reduzir mais o erro.