# SCC0910 - Tópicos Avançados em Ciências de Computação I
# Relatório 2

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

## 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.

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 Levantamento bibliográfico

O tema é bem atual e muito explorado comercialmente, por exemplo por empresas como a Tesla, que utilizam da identificação de placas de trânsito no sistema de navegação e tomada de decisão de seus carros autônomos. Além disso, a bibliografia na área é vasta e vai desde exemplos simples a pesquisas mais complexas. Alguns dos trabalhos encontrados que têm mais relação com o presente projeto serão listados nessa seção.

[ Youssouf N.]:
O autor desenvolveu um sistema de reconhecimento de sinais de trânsito robusto utilizando redes neurais convolucionais (CNN). Ele treinou a CNN para classificar 43 diferentes sinais de trânsito usando um conjunto de dados de referência alemão.

O dataset utilizado continha 39,186 imagens para treino e 12,630 para teste e a CNN desenvolvida alcançou uma precisão de mais de 99% com aproximadamente 800 mil parâmetros.

Além disso, também foram avaliadas as redes **Faster R–CNN** e **YOLOv4** na qual o autor obteve um resultado melhor pela YOLOv4, tanto em precisão quanto em latência no processamento dos quadros, e por isso ele indicou que ela seria a melhor escolha para aplicações real time.

[ Bademian, G.B.B.]:
Este autor utilizou o mesmo dataset de placas alemãs que Youssouf[1], e criou uma CNN que obteve 99,2% de acurácia na identificação e classificação dos dados.

Um ponto interessante desse estudo foi que o autor encontrou no desenvolvimento o problema de overfitting, que é muito comum de acontecer quando se está desenvolvendo uma CNN. Para solucionar este problema o autor utilizou do processo de dropout, e assim obteve um resultado melhor do que antes.

O autor também indica que apesar de ter utilizado 10 épocas no treinamento, a partir da 5ª época o modelo ja tinha estabilizado em uma acurácia próxima do resultado final. Isso indica que o número de épocas poderia ser reduzido para diminuir o tempo de treinamento da rede.

[ Preto, D.D.O.]:
A autora teve como objetivo criar uma base de dados de placas brasileiras e treinar uma CNN desenvolvida em MATLAB para o processamento das imagens e testar diferentes redes neurais com as placas de trânsito para analisar os resultados obtidos.

Ela prontamente indica o problema para encontrar um banco de imagens contendo placas brasileiras. Como o objetivo dela incluía a criação deste banco, ela utilizou o google street view para capturar as imagens das placas, porém como o google proíbe a distribuição de tais imagens a autora não pôde disponibilizar o dataset criado.

Por fim, foram capturadas 4078 imagens no total para este trabalho e foram criadas 12 categorias. Após o treinamento a autora conseguiu uma acurácia de 98.7%, tendo maiores problemas nas placas de direção por se tratarem de placas onde o sinal é o mesmo, apenas invertido.

**Observações:**
Como foi observada essa falta de conjunto de dados brasileiros, o grupo optou em treinar a rede desenvolvida com base em um conjunto de placas de trânsito americanas[4]. Esse conjunto de dados será melhor analisado na próxima seção.

# 3 Análise dos dados

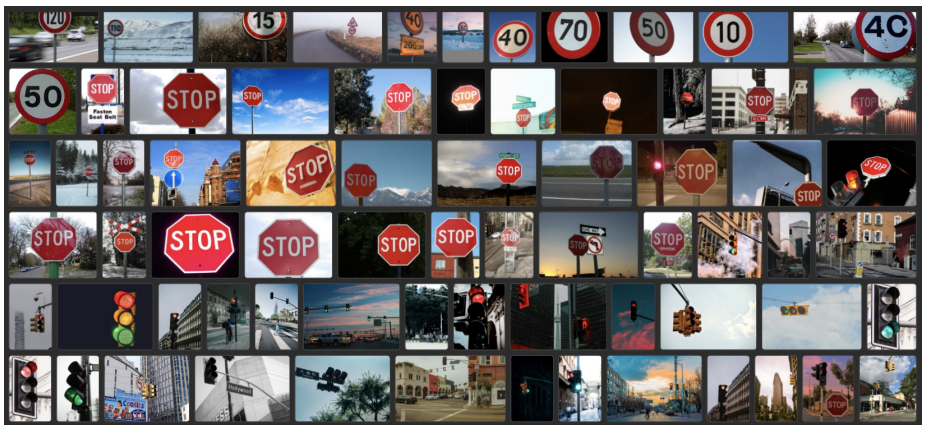

Figure 1: Dataset[4]

O conjunto de dados[4] é composto por 877 imagens previamente classificadas, distribuídas em quatro classes distintas: "Trafic Light" (semáforo), "Stop" (pare), "Speedlimit" (limite de velocidade) e "Crosswalk" (travessia de pedestres).

Uma das observações relevantes é a presença de um número limitado de classes no conjunto de dados. Essas classes são bem diferenciadas entre si, pois abrangem elementos como placas de trânsito e semáforos, que possuem características distintas. Essa distinção clara entre as classes sugere que o modelo deverá ter um desempenho mais elevado na tarefa de classificação.

Além disso, também é valido apontar que o conjunto de dados apresenta um número relativamente baixo de imagens disponíveis. Esse fato pode influenciar na capacidade do modelo de generalizar efetivamente para novos exemplos e pode ser um desafio a ser considerado durante a fase de teste do modelo desenvolvido.

# 4 Implementação

A implementação do trabalho foi realizada em Python, utilizando principalmente as bibliotecas Keras e TensorFlow. Os dados foram organizados em duas partes: um conjunto de imagens e arquivos XML que classificam as imagens.

A arquitetura da rede neural convolucional (CNN) foi projetada levando em consideração a análise de outros modelos existentes, bem como heurísticas do grupo de pesquisa. A estrutura inicial, como é possível ver no tópico seguinte, é composta por 7 camadas, que sofrerão aprimoramento nas próximas etapas.

# 5 Estrutura inicial da CNN

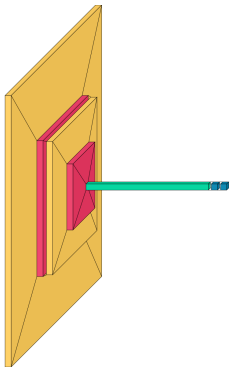

Figure 2: Representação da CNN(visualkeras)

Camadas:

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

# 6 Treinamento

Os dados foram divididos em 70% para aprendizado e 30% para teste e o treinamento consistiu em 10 epochs, cada uma composta por 10 steps. Foi adotado um batch size de 46 durante o processo de treinamento.

Cada epoch teve uma duração média de aproximadamente 30 segundos.

# 7 Resultados Iniciais

Após o treinamento do modelo, observou-se uma melhoria significativa em relação à acurácia. Inicialmente, na primeira epoch, a acurácia era de 0.56, enquanto na última epoch alcançou 0.97. Além disso, o erro diminuiu consideravelmente de 3.1019 para 0.0969.

Ao realizar o teste do modelo treinado, foi obtida uma acurácia de 0.7765 e um erro de 0.95.

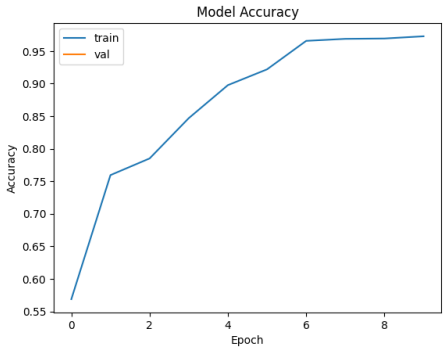


Figure 3: Acurácia durante treino

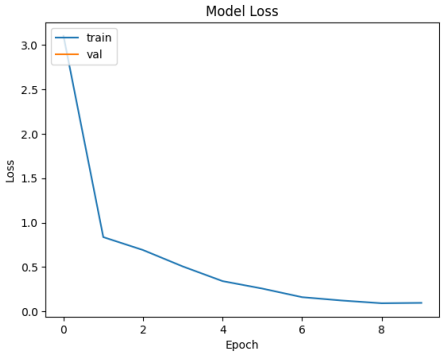


Figure 4: Erro durante treino

# 8 Análise dos Resultados Iniciais

A análise dos resultados iniciais revela um desempenho inferior ao esperado, indicando margem para melhorias significativas.

Uma possível causa para o desempenho é a escolha dos dados utilizados no treinamento. Muitas das fotografias apresentam uma perspectiva distante e, por vezes, elementos de mais de uma classe estão presentes na mesma imagem. Essa complexidade dificulta a correta classificação e impacta negativamente o desempenho do modelo.

Outra possível causa é a ocorrência de overfitting. Observou-se uma queda considerável na acurácia do modelo durante o teste em relação ao treinamento. Esse fenômeno sugere que o modelo pode estar superajustado aos dados de treinamento, comprometendo sua capacidade de generalização para novos exemplos.

Para superar as limitações do modelo atual, serão observadas as seguintes abordagens na segunda etapa deste projeto.

- Modificações na arquitetura: Conforme mencionado por Daniela[3], a utilização de redes mais simples pode resultar em uma acurácia maior, com menor tempo de treinamento. Além disso, simplificar a arquitetura pode ajudar a mitigar o problema de overfitting.

- Transfer Learning: Uma estratégia promissora é treinar modelos utilizando dados de placas alemãs[1] e, posteriormente, reajustá-los utilizando as placas americanas. Essa abordagem permite aproveitar o conhecimento prévio adquirido pelos modelos treinados com os dados alemães, e assim obter resultados aprimorados para as placas americanas.

- Aumentar o número de epochs: Observou-se que a perda estava em processo de diminuição, mas na última epoch ocorreu um aumento. Portanto, realizar mais epochs pode contribuir para uma redução adicional do erro

e melhorar o desempenho do modelo. Porém, existe o risco de aumentar ainda mais o problema de overfitting.

# 9 Segunda abordagem: métodos, justificativas e resultados

Para a segunda abordagem do projeto, foram utilizados 7 métodos para otimização, observando e implementando as anotações realizadas nas análises da etapa anterior, que estão descritos abaixo assim como suas justificativas.

Método 1: Aumentar a complexidade de construção do modelo

Justificativa: Aumentar a complexidade do modelo se mostrou uma estratégia útil para capturar as complexidades e estruturas características das imagens de sinais de trânsito. Os sinais de trânsito, como bem os conhecemos, possuem formas, cores e marcações propositalmente distintas, e um modelo mais complexo pode ajudar a identificar melhor esses atributos ao capturar suas geometrias específicas.

Com a aplicação deste método a acurácia foi 0.7803 e o erro 0.7120.

Método 2: Transfer Learning 1 (VGG16)

Justificativa: A VGG16 é uma arquitetura profunda de uso frequente que, em nosso modelo, demonstrou excelente desempenho nas tarefas de classificação de imagens. Aproveitando a capacidade do modelo pré-treinado VGG16 de reconhecer características comuns de imagens, essa estratégia pode se beneficiar do conhecimento de características ópticas de baixo e alto nível, como bordas, texturas e partes de objetos, para capturar as características específicas que diferenciam umas placas das outras.

Com a aplicação deste método a acurácia foi 0.8523 e o erro 0.4704.

Método 3: Transfer Learning 2 (MobileNetV2)

Justificativa: A estrutura do MobileNetV2 funciona bem em computadores e mantém uma boa precisão ao aplicada. Levando em consideração as necessidades em tempo real dos sistemas de detecção de sinais de trânsito, usar o MobileNetV2 como um filtro pode fornecer um bom equilíbrio entre produtividade e desempenho. A classificação leve possibilita um cálculo mais rápido e ainda captura características adequadas para a classificação de sinais de trânsito, servindo com propriedade às possíveis aplicações propostas para o projeto.

Com a aplicação deste método a acurácia foi 0.8409 e o erro 0.4691.

Método 4: Transfer Learning 3 (MobileNetV2, aumento do número de classes)

Justificativa: Aumentar o número de classes pode ajudar o modelo a convergir mais rapidamente e encontrar melhores soluções. No entanto, os efeitos de taxas de aprendizado mais altas podem ser mais pronunciados no processo de ajuste fino, onde é necessário equilibrar o uso de recursos previamente treinados e a melhoria de tarefas específicas. Nesse caso, o objetivo é aumentar a taxa de

aprendizado e aproveitar as características pré-treinadas do MobileNetV2 para uma convergência mais rápida.

Com a aplicação deste método a acurácia foi 0.8826 e o erro 0.4597.

Método 5: Transfer Learning 4 (MobileNetV2, ajuste fino)

Justificativa: Descongelar as camadas base através do ajuste fino do modelo permite que o modelo modifique as características pré-treinadas do MobileNetV2 específicas para o conjunto de dados de sinais de trânsito. O ajuste fino pode ajudar a refinar a representação aprendida e torná-la mais adequada para a tarefa em questão. No entanto, a otimização do conjunto de dados completo neste caso não resultou em bons resultados, sugerindo que algum congelamento seja útil para preservar características pré-treinadas importantes.

Com a aplicação deste método a acurácia foi 0.8864 e o erro 0.3635.

Método 6: Pequenas alterações (modelo padrão)

Justificativa: Congelar camadas adicionais e ajustar de forma mais fina apenas as camadas do modelo base pode ser uma maneira razoável de detectar sinais de trânsito, por conta de sua constituição que, por vezes, pode ser semelhante. As camadas inferiores do modelo base provavelmente capturam objetos comuns de classificação de imagens em diferentes tarefas.

Com a aplicação deste método a acurácia foi 0.8409 e o erro 3.0855.

# 10 Observações finais

Para o projeto o melhor resultado nos testes foi atingido após aplicado o método 5 (Transfer Learning 4 : MobileNetV2, ajuste fino).

Além disso, assim como é possível perceber nos resultados obtidos nos outputs de cada método, os números expostos podem ser e foram diferentes do esperado em alguns casos, como exposto acima. Isso demonstra que os modelos, quando aplicados sem adaptação e intencionalidade, produzem resultados divergentes, necessitando de estudo e teste para que tenham resultado. No caso de placas de trânsito, essa variável é intensificada, pois, apesar de terem diferenças bem específicas e definidas, possuem também características comuns que podem alterar o desempenho da otimização.

# 11 Referências bibliográficas


1. Youssouf N. Traffic sign classification using CNN and detection using faster-RCNN and YOLOV4. Heliyon. 2022 Nov 26;8(12):e11792. doi: 10.1016/j.heliyon.2022.e11792. PMID: 36471847; PMCID: PMC9718991.

2. Bademian, Gustavo Bulka Bonafé. "Classificação de placas de trânsito com redes neurais para automação de veículos." (2022)

3. Preto, Daniela de Oliveira. "Reconhecimento de placas de trânsito por meio de deep learning." (2018)

4. Kaggle: Road Sign Detection ; (2020).