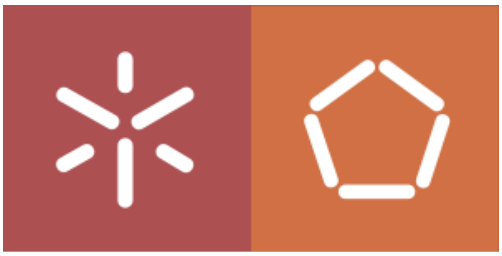

Universidade do Minho
Escola de Engenharia

Visão por Computador e Processamento de Imagem

GTSRB Traffic Sign Recognition

Grupo 13

Nelson Almeida (pg52697) Gustavo Pereira (pg53723)

29 de maio de 2024

Mestrado em Engenharia Informática - Computação Gráfica

Índice

1 Introdução

O presente documento relata o desenvolvimento do trabalho prático, realizado no âmbito da unidade curricular de Visão por Computador e Processamento de Imagem do 1º ano do Mestrado de Engenharia Informática da Universidade do Minho.

O projeto em questão consiste na construção de modelos *Deep Learning* para o reconhecimento de sinais de trânsito recorrendo a técnicas de processamento de imagem, *Data Augmentation* e *Ensemble Learning*, tendo como base para o treino dos mesmo o conjunto de dados GTSRB (*German Traffic Sign Recognition Benchmark*). Estes modelos possuem um papel preponderante nos sistemas de condução autónoma auxiliados pela visão por computador, ao que se procura alcançar uma precisão superior, tornando-os mais seguros no desempenhar do seu papel.

2 Modelos

2.1 Modelo 1

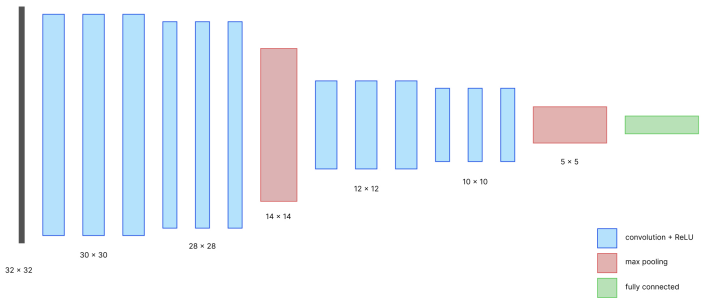

Figura 1: Arquitetura da nossa solução

```python
import torch
import torch.nn as nn

class Conv(nn.Module):
    def __init__(self, num_classes):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 16, 3)
        self.bn1 = nn.BatchNorm2d(16)
        self.relu1 = nn.ReLU()

        self.conv2 = nn.Conv2d(16, 32, 3)
        self.bn2 = nn.BatchNorm2d(32)
        self.relu2 = nn.ReLU()

        self.maxpool1 = nn.MaxPool2d(2)

        self.conv3 = nn.Conv2d(32, 48, 3)
        self.bn3 = nn.BatchNorm2d(48)
        self.relu3 = nn.ReLU()

        self.conv4 = nn.Conv2d(48, 48, 3)
        self.bn4 = nn.BatchNorm2d(48)
        self.relu4 = nn.ReLU()

        self.maxpool2 = nn.MaxPool2d(2)

        self.fc1 = nn.Linear(1200, num_classes)

    def forward(self, x):
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu1(x)
```

```
        x = self.conv2(x)
        x = self.bn2(x)
        x = self.relu2(x)
        x = self.maxpool1(x)
        x = self.conv3(x)
        x = self.bn3(x)
        x = self.relu3(x)
        x = self.conv4(x)
        x = self.bn4(x)
        x = self.relu4(x)
        x = self.maxpool2(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        return x
```

3 Pré-processamento e Data Augmentation

Antes de passarmos ao pré-processamento, devemos analisar os dados existentes.

Os dados contidos no *dataset* de treino têm as seguintes dimensões:

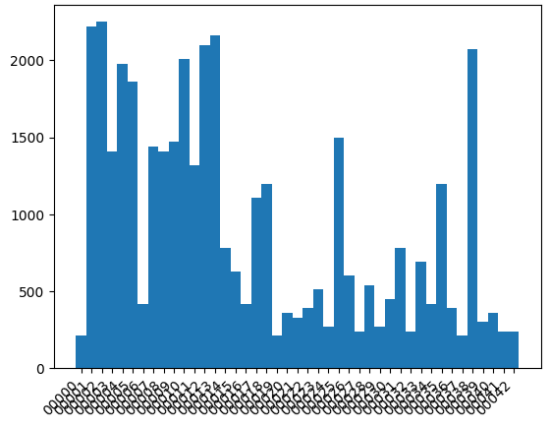

Figura 2: Dados de treino originais

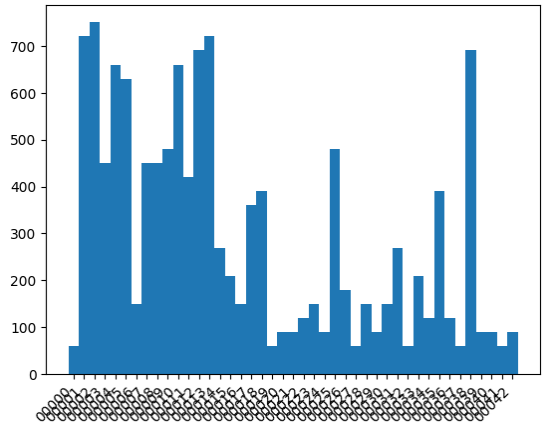

Figura 3: Dados de teste originais

É possível detetar um desbalanceamento na quantidade de imagens disponíveis para cada sinal tanto no *dataset* de treino quanto no de teste.

Para mitigar esta discrepância optamos por duas abordagens, completar os dados em falta com dados gerados de forma sintética através de uma combinação aleatória das seguintes transformações:

```
transformations = {
            "skew": self.skewImage,
            "rotate": self.rotateImage,
            "brightness": self.changeBrightness,
            "perlin": self.addPerlinNoise,
            "gaussian": self.addGaussianNoise,
            "blur": self.addGaussianBlur,
            "colorspace": self.colorspaceTransform
        }
```

As imagens geradas são do seguinte tipo:

Figura 4: Imagens geradas sintéticamente

Com isto obtemos um dataset de treino com as seguintes características:

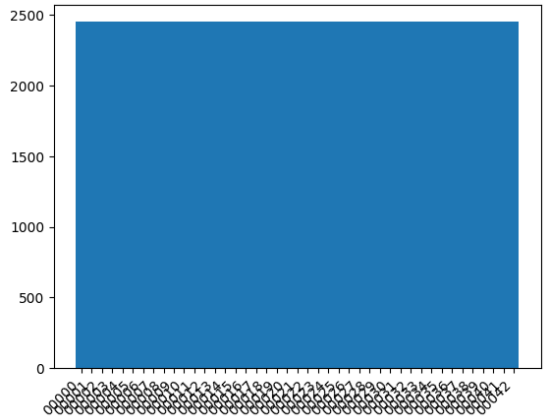

Figura 5: Dados de treino mais dados gerados sintéticamente

Utilizamos ainda uma abordagem que passa por completar o dataset pegando em imagens de cada sinal e aplicando transformações aleatórias até atingir o máximo de imagens por dataset original.

O dataset obtido é do seguinte género:

Figura 6: Imagens balanceadas

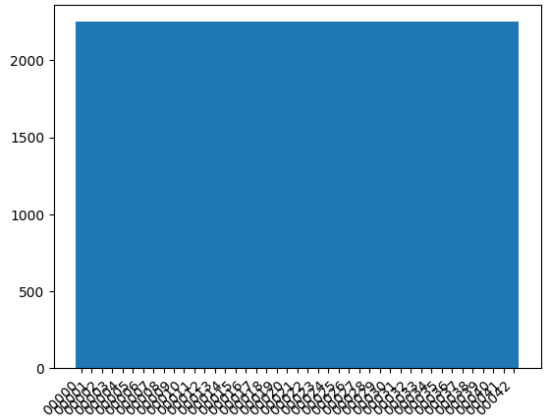

Figura 7: Dados de treino aumentado

De forma a tentarmos melhorar a performance do nosso modelo, vamos partir o/os datasets de treino num rácio 70/30 para validarmos o treino do nosso modelo com um dataset específico para isso(dataset de validação).

4 Testes e Análise de Resultados

Para testar e avaliar a performance do nosso modelo, dividimos os testes nos seguintes moldes.

Começamos por elaborar um benchmark onde treinamos o nosso modelo no

dataset de treino original sem aplicar quaisquer transformações e os melhores resultados de precisão obtidos para o teste e para a validação foram os seguintes

Métrica	Valor
Melhor precisão de treino	99.75260925292969
Melhor precisão de validação	99.77894592285156

Tabela 1: Resultados de precisão de treinamento e validação

Aquando da avaliação do modelo face os dados de teste obtivemos a seguinte precisão: 0.967695951461792

De seguida treinamos um modelo também com os dados de treino originais, aplicando apenas as seguintes transformações:

```
transform = v2.Compose([
        v2.RandomPerspective(distortion_scale=0.2,p=1.0),
        v2.RandomAffine(degrees=3, translate=(0.03,0.03)),
        v2.RandomRotation(3, v2.InterpolationMode.BILINEAR),
        v2.ColorJitter(brightness=0.1, contrast=0.2, saturation
            =0.2),
        v2.RandomEqualize(p=0.5)
])
```

Nesta abordagem já obtivemos resultados de precisão mais satisfatórios, obtendo os seguintes resultados durante o treino:

Métrica	Valor
Melhor precisão de treino	99.61016082763672
Melhor precisão de validação	99.9234848022461

Tabela 2: Resultados de precisão de treinamento e validação

Aquando da avaliação do modelo face os dados de teste obtivemos a seguinte precisão: 0.9888361096382141

Para terminar os treinos com os dados originais aplicando apenas transformações adicionamos ao rol anterior a transformação RandomErasing, que omite secções aleatórias dos dados de treino de visando mitigar o "overfitting"do modelo

```
transform = v2.Compose([
        v2.RandomPerspective(distortion_scale=0.2,p=1.0),
        v2.RandomAffine(degrees=3, translate=(0.03,0.03)),
        v2.RandomRotation(3, v2.InterpolationMode.BILINEAR),
        v2.ColorJitter(brightness=0.1, contrast=0.2, saturation
            =0.2),
        v2.RandomEqualize(p=0.5)
        v2.RandomErasing(p=1, scale=(0.02, 0.1), ratio=(0.3, 3.3),
            value='random')
])
```

Com esta transformação adicional obtivemos os melhores resultados até então com as seguintes precisões:

Métrica	Valor
Melhor precisão de treino	99.78504180908203
Melhor precisão de validação	99.85546875

Tabela 3: Resultados de precisão de treinamento e validação

Aquando da avaliação do modelo face os dados de teste obtivemos a seguinte precisão: 0.992794930934906

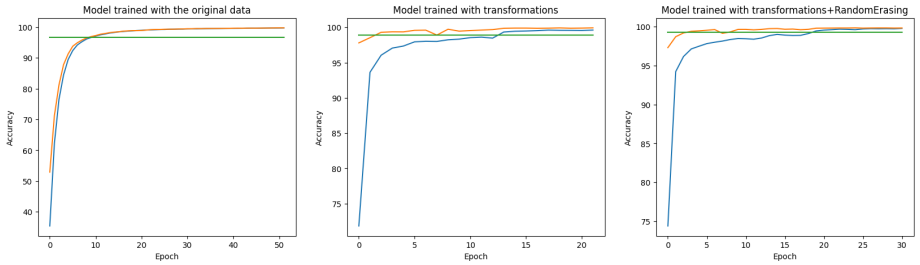

Figura 8: Resultados dos treinos aplicando apenas transformações

Tendo agora obtido resultados de avaliação acima dos 99% passamos à abordagem de treino com os dados aumentados.

Com o dataset completado com os dados sintéticos e as seguintes transformações:

```
transform = v2.Compose([
        v2.RandomPerspective(distortion_scale=0.2,p=1.0),
        v2.RandomAffine(degrees=3, translate=(0.03,0.03)),
        v2.RandomRotation(3, v2.InterpolationMode.BILINEAR),
        v2.ColorJitter(brightness=0.1, contrast=0.2, saturation
            =0.2),
        v2.RandomEqualize(p=0.5)
        v2.RandomErasing(p=1, scale=(0.02, 0.1), ratio=(0.3, 3.3),
            value='random')
])
```

Com estas transformações e os dados aumentados sintéticamente obtivemos os seguintes resultados:

Métrica	Valor
Melhor precisão de treino	97.98402404785156
Melhor precisão de validação	99.95879364013672

Tabela 4: Resultados de precisão de treinamento e validação

Aquando da avaliação do modelo face os dados de teste obtivemos a seguinte precisão: 0.9566112160682678

Estes resultados ficam significativamente à quem do expectável.

Por fim utilizando o dataset completado com os dados aumentados através de transformações dos dados de treino e as seguintes transformações:

```
transform = v2.Compose([
        v2.RandomPerspective(distortion_scale=0.2,p=1.0),
        v2.RandomAffine(degrees=3, translate=(0.03,0.03)),
        v2.RandomRotation(3, v2.InterpolationMode.BILINEAR),
        v2.ColorJitter(brightness=0.1, contrast=0.2, saturation
            =0.2),
        v2.RandomEqualize(p=0.5)
        v2.RandomErasing(p=1, scale=(0.02, 0.1), ratio=(0.3, 3.3),
            value='random')
])
```

Com estas transformações e os dados balanceados obtivemos os seguintes resultados:

Métrica	**Valor**
Melhor precisão de treino	98.44444274902344
Melhor precisão de validação	99.97449493408203

Tabela 5: Resultados de precisão de treinamento e validação

Aquando da avaliação do modelo face os dados de teste obtivemos a seguinte precisão: 0.9950910806655884

Em suma temos então:

Modelo	**Precisão de Teste**
Modelo original sem transformações	0.967695951461792
Modelo com transformações básicas	0.9888361096382141
Modelo com transformações básicas e RandomErasing	0.992794930934906
Modelo com dados sintéticos	0.9566112160682678
Modelo com dados balanceados	0.9950910806655884

Tabela 6: Resultados de precisão de teste para cada modelo

Para colmatar estes resultados, aplicamos a técnica de Ensemble Learning com as abordagens de Soft Voting e Hard Voting com os modelos enumerados acima e os melhores modelos com finetuning de parâmetros como o otimizador de Adam para SGD e os resultados obtidos foram os seguintes:

Métrica	**Soft Voting**	**Hard Voting**
Total de amostras	12630	12630
Todos corretos	11614	11614
Todos incorretos	5	5
Maioria correta	970	970
Voto empatado	12	12
Maioria incorreta	29	29
Percentagem correta	0.9963578780680918	0.9973079968329375

Tabela 7: Resultados de Ensemble Learning: Soft Voting e Hard Voting

5 Conclusão

Face ao trabalho desenvolvido, consideramos ter alcançado um resultado satisfatório, atingindo os objetivos propostos, aplicando as técnicas lecionadas tais como processamento de imagem, *Data Augmentation* e *Ensemble Learning*, com exceção da técnica de *Transfer Learning*. O valor da *accuracy* alcançado condisse com as nossas expectativas, porém, acreditamos haver margem para melhoria através da exploração de outras técnicas de pré-processamento dos dados assim como a aplicação de *Transfer Learning*, para o qual o poder computacional que dispomos dificultou o avanço na nossa pesquisa.