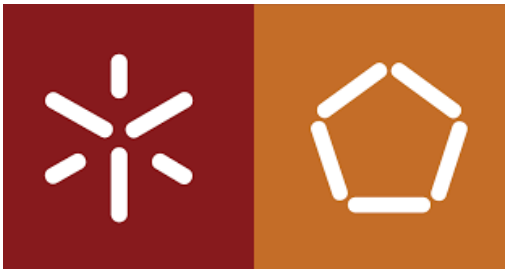
Visão por Computador

1º ano - 2º semestre

Trabalho Prático

Grupo 7

Carlos Machado (PG53721)
Vasco Oliveira (PG54269)



Mestrado em Engenharia Informática
Departamento de Informática
Universidade do Minho
2022/2023

Índice

1 Introdução

Este trabalho tem como objetivo explorar modelos de deep learning aplicados ao dataset GTSRB (um dataset com diversas imagens de sinais de trânsito) para alcançar a melhor precisão possível no conjunto de teste. O trabalho está dividido em duas partes principais: a primeira parte foca-se em técnicas de *data augmentation* para melhorar o desempenho dos modelos, e a segunda na utilização de *ensemble* de redes neuronais para reforçar os resultados obtidos na primeira parte.

Neste relatório iremos apresentar e falar sobre a rede neuronal utilizada, as diversas *data augmentations* que testamos, os melhores modelos que obtivemos e a *ensemble* de redes desenvolvida com o seu melhor resultado.

2 Rede Neuronal

2.1 Modelo

Para este problema de classificação de imagens treinamos *Redes Neuronais Convolucionais* usando *pytorch*, com as camadas definidas a seguir:

```
==========================================================================================
Layer (type:depth-idx)      Input Shape           Output Shape          Param #
==========================================================================================
Conv                        [BATCH, 3, 32, 32]        [BATCH, 43]               --
|-Conv2d: 1-1               [BATCH, 3, 32, 32]        [BATCH, 16, 30, 30]       448
|-BatchNorm2d: 1-2          [BATCH, 16, 30, 30]       [BATCH, 16, 30, 30]       32
|-ReLU: 1-3                 [BATCH, 16, 30, 30]       [BATCH, 16, 30, 30]       --
|-Conv2d: 1-4               [BATCH, 16, 30, 30]       [BATCH, 32, 28, 28]       4,640
|-BatchNorm2d: 1-5          [BATCH, 32, 28, 28]       [BATCH, 32, 28, 28]       64
|-ReLU: 1-6                 [BATCH, 32, 28, 28]       [BATCH, 32, 28, 28]       --
|-MaxPool2d: 1-7            [BATCH, 32, 28, 28]       [BATCH, 32, 14, 14]       --
|-Conv2d: 1-8               [BATCH, 32, 14, 14]       [BATCH, 48, 12, 12]       13,872
|-BatchNorm2d: 1-9          [BATCH, 48, 12, 12]       [BATCH, 48, 12, 12]       96
|-ReLU: 1-10                [BATCH, 48, 12, 12]       [BATCH, 48, 12, 12]       --
|-Conv2d: 1-11              [BATCH, 48, 12, 12]       [BATCH, 48, 10, 10]       20,784
|-BatchNorm2d: 1-12         [BATCH, 48, 10, 10]       [BATCH, 48, 10, 10]       96
|-ReLU: 1-13                [BATCH, 48, 10, 10]       [BATCH, 48, 10, 10]       --
|-MaxPool2d: 1-14           [BATCH, 48, 10, 10]       [BATCH, 48, 5, 5]         --
|-Linear: 1-15              [BATCH, 1200]             [BATCH, 43]               51,643
==========================================================================================
Total params: 91,675
Trainable params: 91,675
Non-trainable params: 0
==========================================================================================
```

2.2 Treino

Para o treino dos modelos usamos:

1. **Optimizer** Adam
2. **Função *loss*** *Cross Entropy Loss*
3. **Learning Rate Scheduler** - Reduce on Plateau
4. **Dataset de Validação**, feito de uma divisão aleatória de 30% do *dataset* de treino
5. **Tolerância** de falta de melhorias de 9 *epochs*

3 Data Augmentations

No pré-processamento dos dados, utilizamos técnicas de *data augmentation*, tanto nos dados de treino como nos dados de validação e teste, de maneira a preparar os dados. A este conjunto de transformações damos o nome de `base_transform`, constituido por estas transformações:

1. `v2.ToImage()`
   - **Descrição**: Converte os dados de entrada em imagens.
   - **Objetivo**: Assegura que os dados estejam no formato correto.
2. `v2.Resize((IMG_SIZE, IMG_SIZE))`
   - **Descrição**: Redimensiona as imagens para um tamanho fixo de `IMG_SIZE x IMG_SIZE`.
   - **Objetivo**: Garante que todas as imagens tenham as mesmas dimensões, facilitando o processamento do modelo.
3. `v2.ToDtype(torch.float32, scale=True)`
   - **Descrição**: Converte os dados para o tipo de dado `float32` e escala os valores das imagens.
   - **Objetivo**: Prepara os dados para serem compatíveis com as operações de tipo *float32*, necessárias para o treino do modelo.
4. `v2.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])`
   - **Descrição**: Normaliza as imagens com médias e desvios padrões específicos para cada canal de cor RGB.
   - **Objetivo**: Centraliza os valores dos pixels das imagens escalando-os de acordo com os valores de média e desvio padrão, o que ajuda a estabilizar e acelerar o treino do modelo.

Também usamos técnicas de *data augmentation*, exclusivamente nos dados de treino, para aumentar a diversidade do conjunto de dados e melhorar a robustez do modelo. A estas transformações damos o nome de `transform`, que inclui as seguintes técnicas:

1. `v2.RandomRotation(x, v2.InterpolationMode.BILINEAR)`
   - **Descrição**: Roda as imagens aleatoriamente até x graus usando interpolação bilinear.
   - **Objetivo**: Simula variações leves na orientação das imagens, tornando o modelo mais robusto a pequenas rotações e ajudando-o a generalizar melhor.
2. `v2.RandomAffine(degrees=x, translate=(0.0y,0.0y))`
   - **Descrição**: Aplica transformações afins aleatórias às imagens, com rotações até x graus e translações até y% do tamanho da imagem.
   - **Objetivo**: Introduz variações na posição e orientação das imagens, aumentando a robustez do modelo a mudanças na localização e ângulo dos sinais de trânsito.
3. `v2.RandomPerspective(distortion_scale=x, p=y)`
   - **Descrição**: Aplica uma transformação de perspectiva aleatória às imagens, com uma escala de distorção x e probabilidade y.
   - **Objetivo**: Simula diferentes ângulos de visão, tornando o modelo mais robusto a variações nas diferentes perspectivas dos sinais de trânsito.
4. `v2.ColorJitter(brightness=x, contrast=y, saturation=z)`
   - **Descrição**: Altera aleatoriamente o brilho, contraste e saturação das imagens com os fatores especificados.
   - **Objetivo**: Simula diferentes condições de iluminação e cores, tornando o modelo mais robusto a variações de iluminação e saturação.

5. `v2.RandomErasing(p=1, scale=(0.0x, 0.0y), ratio=(z, w), value='random')`

    - **Descrição**: Apaga aleatoriamente uma região retangular da imagem onde a escala da área a ser apagada varia entre x% e y% da imagem e o rácio varia entre z e w. O valor dos pixeis apagados é aleatório.
    - **Objetivo**: Ajuda a regularizar o modelo, introduzindo oclusões aleatórias nas imagens durante o treino, tornando o modelo menos sujeito a *overfitting* e mais robusto a oclusões.

Para além dessas *data augmentations* mais comuns também experimentamos usar transformações que esperam lotes de amostras como entrada, e não imagens individuais.

1. `CutMix`

    - **Descrição**: Combina duas imagens de treino ao cortar uma região de uma imagem e substituí a mesma por uma região correspondente de outra imagem.
    - **Objetivo**: Fornece uma regularização adicional ao forçar o modelo a aprender a partir de partes combinadas de diferentes imagens, aumentando a robustez e melhorando a generalização.

2. `MixUp`

    - **Descrição**: Cria novas imagens de treino ao combinar pares de imagens e seus respectivos rótulos de forma linear.
    - **Objetivo**: Melhora a generalização ao fornecer exemplos interpolados entre diferentes classes.

4 Modelos

4.1 Testes

Para conseguirmos chegar aos melhores modelos que temos atualmente foi necessário realizar vários testes, não só para descobrir que transformações melhoram os modelos mas também para verificar se a nossa rede neuronal tinha problemas ou não. Como tal foram criados vários modelos, todos estes guardados na pasta `models` e documentados em ficheiros de texto de maneira a podermos analisar posteriormente os seus resultados com maior facilidade.

Em relação à rede neuronal deparamo-nos com alguns problemas, tais como usar os dados de teste no dataset de validação, não dar shuffle aos dados de treino antes da sua utilização, utilizar transformações exclusivas dos dados de treino nos dados de validação, descobrir qual optimização seria melhor (SDG ou Adam) ou que transformações de preparação de dados devíamos usar (`base_transform`).

Em relação às transformações de preparação dos dados (`base_transform`) as transformações apresentadas na secção anterior são todas utilizadas nos resultados dos melhores modelos menos o `v2.Normalize`, sendo que o uso desta transformação levava sempre a uma pior *accuracy* nos testes. As transformações `CutMix` e `MixUp`, apresentadas na secção anterior, também acabaram por não levar o modelo para os melhores resultados.

Também foram testadas outras transformações que não chegaram a ser documentadas na secção anterior, pois não tiveram grandes resultados, como por exemplo `v2.AutoAugmentation` ou `v2.CenterCrop`.

4.2 Melhores Modelos

Aqui vamos apresentar os nossos melhores modelos, ou seja, os que tiveram uma melhor *accuracy* perante os dados de teste. É de apontar que todos os modelos que foram treinados em conjunto com problemas que poderiam afetar a sua legitimidade não serão contabilizados como os melhores modelos, mesmo que tenham obtido resultados melhores.

Todos os melhores modelos tiveram as seguintes transformações de preparação de dados (`base_transform`):

```
base_transform = v2.Compose([
    v2.ToImage(),
    v2.Resize((IMG_SIZE, IMG_SIZE)),
    v2.ToDtype(torch.float32, scale=True),
])
```

Aqui estão os 5 melhores modelos, em ordem decrescente, com as suas respetivas transformações usadas exclusivamente nos dados de treino (`transform`) e o a sua *accuracy* perante os dados de teste (`evaluation`):

1. **model12fr.pt**:

```
transform = v2.Compose([
    v2.RandomPerspective(distortion_scale=0.3,p=1.0),
    v2.RandomAffine(degrees=3, translate=(0.03,0.03)),
    v2.RandomRotation(2, v2.InterpolationMode.BILINEAR),
    v2.ColorJitter(brightness=0.2, contrast=0.4, saturation=0.3),
])

evaluation: 99.17656183242798%
```

2. **model17fr.pt**:

```
transform = v2.Compose([
        v2.RandomPerspective(distortion_scale=0.3,p=1.0),
        v2.RandomAffine(degrees=4, translate=(0.03,0.03)),
        v2.RandomRotation(3, v2.InterpolationMode.BILINEAR),
        v2.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.3),
])

evaluation: 98.9311158657074%
```

3. **model13fr.pt**:

```
transform = v2.Compose([
        v2.RandomPerspective(distortion_scale=0.3,p=1.0),
        v2.RandomAffine(degrees=3, translate=(0.03,0.03)),
        v2.RandomRotation(2, v2.InterpolationMode.BILINEAR),
        v2.ColorJitter(brightness=0.2, contrast=0.4, saturation=0.3),
        v2.RandomErasing(p=1, scale=(0.03, 0.05), ratio=(0.3, 3.3), value='random'),
])

evaluation: 98.8440215587616%
```

4. **model11fr.pt**:

```
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),
])

evaluation: 98.78067970275879%
```

5. **model9fr.pt**:

```
transform = v2.Compose([
    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),
])

evaluation: 98.74901175498962%
```

5 Ensembles de Redes

Usando os 5 melhores modelos, construimos dois *ensembles*, usando *Hard Voting* e *Soft Voting*.

5.1 Resultados:

Encontram-se aqui as previsões dos *ensembles* quanto ao *dataset* de teste.

Hard Voting

```
Total: 12630
All correct: 12246
All incorrect: 6
Majority correct: 306
Tie Vote: 25
Majority Wrong: 47
Percentage right: 0.9938242280285036
```

Soft Voting

```
Total: 12630
Correct: 12577
Incorrect: 53
Percentage right: 0.995803642121932
```

6 Conclusão

Através do desenvolvimento deste trabalho prático conseguimos aprofundar ainda mais o nosso conhecimento sobre modelos de *deep learning*, explorando diversas *data augmentations* e o potencial de utilizar *ensembles* de redes.

Apesar de terem ocorrido diversos problemas ao longo do desenvolvimento das Redes Neuronais e o facto de treinar um modelo demorar muito tempo, achamos que obtivemos bons resultados finais, tanto na criação dos modelos como no *ensemble* de redes, e conseguimos testar uma grande variedade de *data augmentations*.