Aprendizagem Automática Avançada
2022/2023

Projeto

Classificação de Sinais de Trânsito com Recurso a *Convolutional Neural Networks* e *Transfer Learning*

João Faia, 47051
Tomás Oom, 59447

1. Introdução

A aprendizagem automática surgiu como uma força transformadora em vários domínios, revolucionando a forma como se resolvem problemas complexos e se tomam decisões informadas. No contexto da segurança rodoviária, a classificação exata e eficiente dos sinais de trânsito desempenha um papel fundamental na tecnologia atual e futura dos veículos autónomos e no auxílio ao condutor. Neste contexto, as técnicas de aprendizagem automática, em particular as *Convolutional Neural Networks* (CNN), ganharam uma força significativa devido à sua ótima capacidade de extrair características relevantes dos dados visuais[1, 2]. A eficácia da interpretação dos sinais de trânsito pode estar dependente da qualidade dos modelos produzidos, sendo a sua testagem um passo essencial nesta área. As CNN, como as restantes redes neurais, possuem uma flexibilidade enorme para se adaptar a diferentes tipo de dados e aprender os seus padrões, no entanto, são modelos com uma elevada customização, podendo chegar aos milhões de parâmetros[3, 4]. A arquitetura de uma CNN é o que determina a sua performance em determinada tarefa, no entanto, a escolha da arquitetura é um processo complexo e resumido à experimentação. Dependendo do problema, a criação de uma CNN de raiz pode não ser uma opção, existindo diversos modelos pré-treinados em diferentes *datasets* (de grande densidade de dados) que providenciam uma base bastante sólida[5] para uma generalidade de problemas. Este tipo de modelos são importados e é realizado o *fine-tuning* dos pesos já treinados, especializando o modelo para uma tarefa específica. Em certos casos, quando o *dataset* não possui um tamanho satisfatório ou o poder computacional é reduzido, estes modelos podem ser utilizados sem a realização do *fine-tuning,* congelando os pesos. Deste modo, o presente projeto tem como principal objetivo a avaliação da performance de duas arquiteturas de CNN e dois modelos de *transfer learning* para a classificação de sinais de trânsito.

2. Implementação

O projeto utiliza o *dataset* GTSDB (*German Traffic Sign Dataset Benchmark*)[6] que consiste num conjunto de imagens de sinais de trânsito capturadas na Alemanha. Este *dataset* inclui no total 51839 imagens divididas em 43 classes, apresentando um número robusto de exemplos para treinar uma CNN de origem. O *dataset* proveio da fonte já dividido em conjunto de treino (39209 imagens) e teste (12630 imagens – 25%), sendo este último bastante satisfatório a nível de variabilidade e aleatoriedade, ficando assim como conjunto de validação utilizado para avaliar a performance dos modelos. O conjunto de teste, utilizado como uma avaliação final do melhor modelo, foi construído a partir de x imagens retiradas aleatoriamente da internet das diferentes classes do *dataset*, permitindo ao modelo classificar exemplos com uma origem diferente ao do conjunto de dados e mostrar o seu poder de generalização. As imagens de treino foram incluídas num array e foram primeiramente analisadas ao nível da sua resolução. A figura 1 permitiu ter uma ideia da distribuição das resoluções presentes nas imagens do *dataset*, sendo essencial para os modelos um redimensionamento para que todas fiquem do mesmo tamanho (tamanho determinado na camada de input).

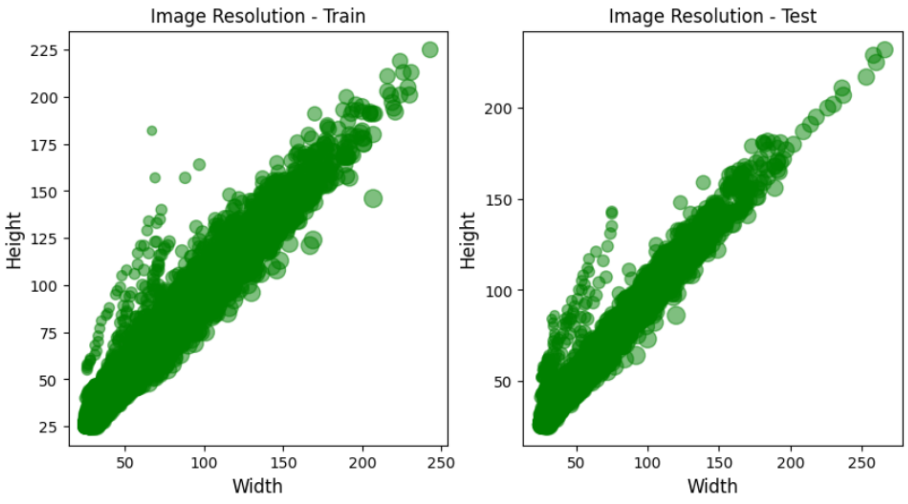

Figura 1 – Representação da distribuição de resoluções do conjunto de treino e de teste. Os tamanho dos círculos representa a proporção das imagens.

A maioria das imagens contém uma proporção (*aspect ratio*) perto de 1, e um tamanho variável de aproximadamente 25x25 até 225x225. Tendo em conta que o poder computacional necessário para treinar os modelos é tanto maior quanto maior o número de pixéis, foi decidido realizar um redimensionamento de todas as imagens para uma resolução 32x32. Adicionalmente, o redimensionamento de imagens para maiores resoluções está associado à interpolação de novos pixéis, não acrescentando qualquer tipo de informação à imagem[7]. Por outro lado, a redução de imagens através da remoção de pixéis, permite a preservação de certas características presentes originalmente, sendo esta abordagem, neste caso, preferível. Posteriormente, foi também analisado o número de imagens pertencentes a cada classe de modo a verificar a existência de um possível desequilíbrio no *dataset* que provocaria um enviesamento do modelo para a classe mais prevalente. A figura 2 permite perceber que o conjunto de dados se encontra bastante desequilibrado, contendo classes com aproximadamente 200 imagens e outras com 2000. A abordagem escolhida para tratar este problema foi a remoção de imagens aleatórias para classes com mais de 1000 imagens e a aplicação de *data augmentation* para classes com menos de 1000 imagens. O processo de *data augmentation* realizado escolheu aleatoriamente imagens para cada classe em falta e aplicou pequenas transformações às imagens, gerando novos exemplares muito parecidos aos originais. Estas alterações incluíram a translação da imagem para a direita, esquerda, cima e/ou baixo e/ou zoom in/out. O método utilizado para preencher pixéis que possam ter ficado em falta (exemplo: zoom out), foi o "*nearest*". A *data augmentation* para além de contribuir para a redução do enviesamento do modelo, origina também um conjunto de dados mais variável que se traduz num modelo menos propenso a *overfitting* e que generaliza melhor.

Após o equilíbrio das classes, o conjunto de treino ficou com 43000 imagens, 1000 para cada categoria. Um último passo de pré-processamento, foi a normalização dos pixéis para que estes se enquadrassem numa escala de 0 a 1. Este processo torna os dados mais consistentes e melhora a convergência dos modelos, considerando que a amplitude de valores é menor, diminuindo o enviesamento. A normalização é realizada simplesmente através da divisão de cada pixel por 255, valor máximo para uma escala de 8-bits.

O pré-processamento necessário para a criação das variáveis com as categorias (y_train e y_test) é apenas o *one-hot encoding* de cada exemplo, agregado num array.

Além das imagens e das classes associadas, o *dataset* inclui a informação adicional da região de interesse (ROI) de cada imagem. O ROI é definida por quatro variáveis que delimitam a caixa que envolve o sinal. Esta informação é útil para delimitar a área relevante da imagem durante o processo de classificação.*

Tendo em conta que as imagens originais sofreram um redimensionamento e novas imagens foram criadas através da *data augmentation*, as variáveis do ROI tiveram também de sofrer o mesmo redimensionamento e a mesma geração de novos ROI com as transformações coincidentes.

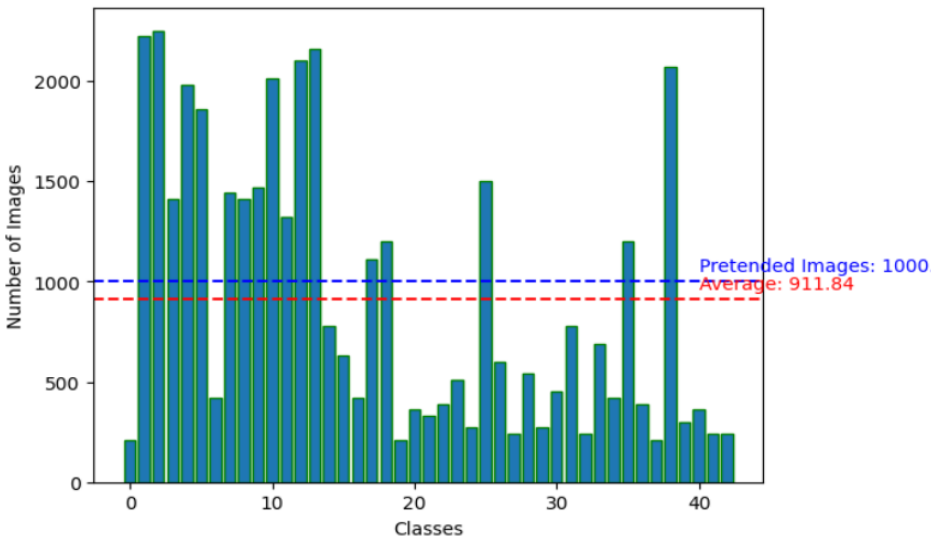

Figura 1 - Número de imagens presentes em cada classe do conjunto de dados.

*Na eventualidade de aplicação de *image segmentation*, a variável ROI seria essencial para identificar múltiplas classes numa só imagem ou frame de vídeo. A classificação de um vídeo era o plano original para o teste final deste projeto (como referido na proposta inicial), no entanto todas as abordagens testadas não foram bem sucedidas. Ainda assim, o ROI foi mantido para verificar se a rede neural conseguiria prever a localização do sinal.

2.1 Modelos

Com o intuito de classificar as imagens do conjunto de dados, 4 modelos distintos foram concebidos de modo a serem testados em termos de performance. Os dois primeiros modelos construídos foram duas CNN com arquiteturas distintas, principalmente devido ao número de filtros e tamanhos diferentes nas hidden layers. Os dois últimos foram baseados em transfer learning dos modelos ResNet50 e DenseNet121, ambos pré-treinados no dataset ImageNet. Devido a limitações computacionais, não foi possível ajustar os pesos das *hidden layers* dos modelos de *transfer learning (fine-tuning*), no entanto, esta impossibilidade levantou a interessante questão de qual a performance basal de um modelo bastante desenvolvido, treinado num *dataset* massivo (ImageNet), e o seu poder de generalização para tarefas mais especificas.

Além da classificação das imagens, tendo em conta que cada imagem tem um ROI associado, outra tarefa pretendida para as redes neurais é a previsão do ROI em cada imagem de input, para posteriormente o modelo estimar a localização do sinal de trânsito na imagem e a que classe pertence. Para se conseguir uma dupla tarefa, foi necessário a criação de duas *layers* de *output*, cada uma com o seu propósito e respetiva *loss function*. A arquitetura da primeira CNN (CNN1) está ilustrada na figura 3, estando a segunda no material suplementar (Figura S1).

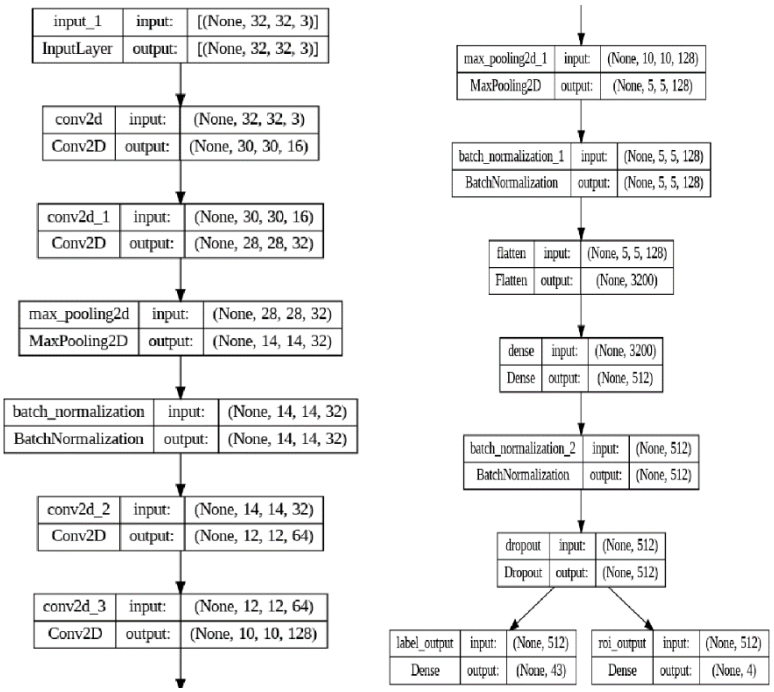

Figura 2 – Arquitetura da CNN1 com a indicação do tipo de *layer* e o respetivo input/output.

A CNN1 possui uma input *layer* que recebe dados com uma dimensão 32x32x3, sendo esta a dimensão das imagens previamente redimensionadas (largura x altura x canais RGB). Todas as convolutional layers desta arquitetura possuem um tamanho 3x3 e uma função de ativação ReLU associada. As *hidden layers* iniciam com uma *convolutional layer* de 16 filtros que alimenta uma segunda *layer* com 32 filtros, onde posteriormente é realizado o max pooling (2 por 2) para extrair as características mais importantes identificadas pelos filtros, seguido da *batch normalization layer* para normalizar o output e servir de regularização (através da adição de ruído). Após esta primeira fase, este padrão repete-se com duas *convolutional layers* de 64 e 128 filtros, sendo posteriormente o seu output achatado numa *array* de uma dimensão que serve de input na *fully connected network* composta por 512 neurónios. A *dropout layer*, antes das *output layers*, permite regularizar o modelo para prevenir o *overfit*, inativando 50% dos neurónios e prevenindo a especialização de certas regiões da rede. As duas *output layers* como referido anteriormente, são direcionadas para duas tarefas diferentes. A primeira, com a função de ativação softmax, origina 43 valores de probabilidades (referentes a cada categoria de sinal de trânsito), sendo a

mais elevada a escolhida para classificar o input. A *loss function* utilizada nesta layer é a *categorical cross-entropy*. A segunda *output layer*, possui uma função de ativação linear e retorna 4 valores de ROI, referentes aos lados do retângulo da região de interesse. A *loss functio*n, por se tratar de uma variável contínua, é a *mean squared error* (MSE). O algoritmo de otimização escolhido foi o *Adaptive Moment Estimation* (adam), por se tratar de um algoritmo eficiente com uma rápida convergência, sendo este aspeto essencial quando o número de imagens do conjunto de treino é bastante elevado.

A segunda arquitetura CNN2 (figura S1) foi construída de forma semelhante à primeira, mas com uma abordagem diferente ao nível dos filtros. Primeiramente, o primeiro filtro foi mais abrangente com um tamanho 5x5 em vez de 3x3 e com uma menor quantidade de filtros (8). Esta abordagem teria como propósito reduzir a complexidade das primeiras características captadas pelos filtros tendo em conta que os pixéis iniciais contêm bastante ruído. Em segundo lugar, e em oposição à CNN1, as *convolutional layers* não duplicam o número de filtros a cada nova *layer*. A primeira passa de 8 para 16, mas a terceira mantém 16 filtros, sendo apenas na quarta que é aumentada para 32, na qual mantém novamente na quinta *layer* e última *convolutional layer*. Este processo tem também o propósito de manter o mesmo nível de complexidade de *features* extraídas duas *layers* seguidas para que possa existir mais uma oportunidade para filtrar algum ruído que possa estar presente. A terceira e última característica diferente da CNN1 são os 128 neurónios utilizados na parte final da rede em vez dos 512, produzindo uma rede com menor número de parâmetros e, por conseguinte, menor probabilidade de *overfit*.

Os dois modelos de transfer learning ResNet50 e DenseNet121 possuem uma arquitetura extremamente complexa, com 50 e 121 *hidden layers*, não podendo ser apresentadas em figuras como as anteriores. Ainda assim, a arquitetura do modelo necessita de ser completada para ficar adaptada à dimensão dos *inputs* e *outputs* deste problema. Para este fim, foi criada uma *input layer* e duas *output layers* iguais aos restantes modelos, além disso, na *layer* anterior aos outputs, foi também criada uma layer de 512 neurónios com ReLU e um *dropout rate* de 50%. É necessário apontar que todas as *layers* anteriores foram congeladas, sendo apenas atualizados os pesos das recém criadas.

Todos estes modelos utilizaram o conjunto de validação enquanto estavam a ser treinados, tendo sido calculadas as métricas de performance a cada *epoch*, avaliando a evolução do modelo em ambos os conjuntos de dados.

Os modelos foram treinados por 30 *epochs* e com um *batch size* de 32, sendo avaliados pela *accuracy* para a tarefa de classificação e pelo MSE para a previsão dos ROI. É importante ressalvar que a *accuracy* é uma métrica bastante influenciável pelo desequilíbrio das classes, no entanto esse problema já foi resolvido na fase de *data augmentation*

3. Resultados

Após o treino dos modelos, foi possível analisar a evolução da sua performance ao longo das 30 *epochs* (figura 4). Numa primeira análise relativa às duas CNN com os parâmetros totalmente treinados, é possível verificar que ambas começam com uma *accuracy* relativamente baixa (*burn-in*), mas têm uma subida repentina durante as primeiras 5 *epochs* para CNN1 e 10 *epochs* para CNN2, estabilizando naquela região. É possível verificar que a CNN2 consegue melhores resultados no conjunto de validação do que no de treino, o que sugere que a regularização aplicada afetou ligeiramente a performance no teste em detrimento de uma melhor classificação global. No MSE para a previsão do ROI, o *burn-in* é quase nulo em ambos os casos, pois na primeira *epoch* o valor de MSE desce e estabiliza instantaneamente. Entre as duas abordagens, a CNN1 mostrou resultados superiores à CNN2 (Tabela 1), sendo o valor de *accuracy* = 0.97 e MSE = 1.19, valores bastante satisfatórios para o problema. Uma possível explicação para o melhor desempenho da CNN1 é o facto de possuir *convolution layers* com filtros até 128 contra os 32 da CNN2. Esta diferença é suficiente para a CNN1 conseguir captar *features* mais complexas e ajustar-se melhor aos dados do que a CNN2, não tendo sido a hipótese apresentada acima relativa ao ruído das imagens tão relevante para o problema.

No que toca aos dois modelos de *transfer learning*, é possível verificar um ligeiro *burn-in*, no entanto, ambos estão *underfit* aos dados, apresentando valores de *accuracy* muito baixos no conjunto de validação. Ainda assim, o modelo DenseNet121 conseguiu atingir valores de *accuracy* de 0.55, duas vezes melhor que o ResNet50, com um valor de 0.24. Uma possível explicação para esta diferença pode ser o número de *layers* de cada modelo, tendo em conta que DenseNet121 tem duas vezes mais *layers* do que o ResNet50, tenderá possivelmente a aprender mais padrões das imagens do ImageNet, generalizando melhor para os sinais de trânsito do presente *dataset*. De qualquer forma, ambos modelos têm melhor performance do que adivinhar aleatoriamente a classe. Relativamente aos valores de MSE,

surpreendentemente, são valores próximos dos obtidos na CNN1 e CNN2, o que, assumindo que os pesos obtidos pelo pré-treino do modelo na ImageNet não contribuem para a previsão dos ROI, parece indicar que a camada de 512 neurónios adicionada antes do output é suficiente para realizar uma boa previsão das regiões de interesse. Conclui-se, no entanto, que o congelamento dos pesos pré-treinados, apesar de reduzir astronomicamente a exigência computacional do treino do modelo, não produz bons resultados para um classificador, sugerindo que os modelos de *transfer learning* só conseguem generalizar até certo ponto, ainda que 0.55 seja um valor bastante aceitável para um modelo que teve mais de 95% limitado no treino.

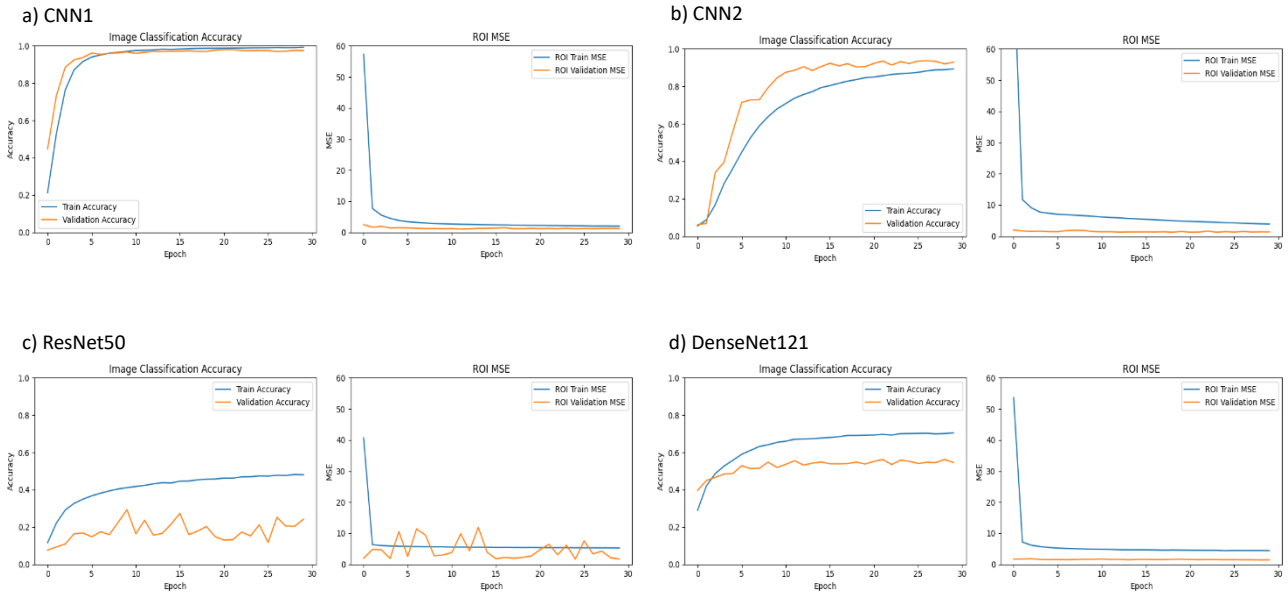

Figura 3 - Comparação da *accuracy* (classificação de imagens) e *mean squared error* (ROI) no conjunto de treino e de validação ao longo do número de *epochs* para o modelo: a) CNN1; b) CNN2; c) ResNet50; d) DenseNet121.

Numa comparação de todos os modelos criados, é possível verificar (tabela 1) que as CNNs obtiveram melhores resultados na classificação de imagens do que os modelos de *transfer learning*, sendo a CNN1 a que obteve melhor resultado no total. Possivelmente, se se tivesse realizado o *fine-tuning*, os modelos de *transfer learning* seriam tão ou mais eficazes devido à inicialização dos pesos. Existem diversas referências a sugerir que o *transfer learning* é uma ferramenta poderosa e muito eficiente em todo o tipo de tarefas, resolvendo o problema principal da falta de dados.[8, 9, 10, 11, 12]

Tabela 1 – Resumo das métricas finais de *accuracy* e mse de treino e validação para cada modelo.

Model	train_accuracy	val_accuracy	train_mse	val_mse
CNN1	0.992116	0.974584	1.979864	1.191323
CNN2	0.893279	0.928345	3.893484	1.383312
ResNet50	0.480372	0.241409	5.278212	1.710061
DenseNet121	0.704884	0.546556	4.403074	1.457823

Por fim, com o intuito de testar o melhor modelo, foram retiradas imagens de sinais de trânsito da internet para serem classificadas, de modo a estarem completamente independentes do *dataset* original e possibilitarem a simulação da vida real, por exemplo uma segmentação de *frame* de um vídeo gravado por uma câmara de um carro. Foram passadas 25 imagens pelo CNN1 originando o resultado da figura 5. Entre as 25, o modelo errou a classificação de 2, resultando numa *accuracy* de 92%, um resultado bastante satisfatório para uma arquitetura tão simples como esta.

Figura 4 – Resultado final da classificação com CNN1 com o conjunto de 25 imagens de teste independentes ao *dataset* original. A classe prevista pelo modelo está no topo da imagem. A classificação das imagens 10 e 23 estão incorretas. Os retângulos verdes são previsões do modelo de onde estaria a região de interesse da imagem.

4. Comentários finais

Este projeto permitiu realizar uma comparação entre vários modelos para a classificação de sinais de trânsito, demonstrando a variabilidade de abordagens para resolver esta tarefa. Possibilitou também ter uma perceção da generalização dos modelos de *transfer learning* sem qualquer tipo de *fine-tuning*. Um ponto bastante positivo que este projeto revelou foi a eficiência com que se conseguiu implementar um modelo com uma excelente performance, utilizando uma arquitetura bastante simples de CNN. Por outro lado, uma limitação bastante evidente foi a exigência computacional de alguns métodos que levou à alteração do plano proposto inicialmente para o *transfer learning*. Além do referido, outra limitação é o facto das CNNs não conseguirem classificar múltiplas classes em simultâneo, impedindo uma vez mais a realização da classificação de um vídeo de condução como teste final, ideia também proposta inicialmente no plano do projeto. Uma possível melhoria ao projeto teria sido a testagem de mais formas de *transfer learning*, realizando o congelamento parcial das *layers* e não total como foi efetuado. Assim, congelando apenas as *layers* com um número de filtros menor (as iniciais), permitiria a extração de características de alto nível do *dataset* utilizado para o fine-tuning dos pesos. Outra possível melhoria, era a recolha de um número de imagens bastante superior para a realização do teste final, permitindo ter maior confiança na métrica calculada.

Referências

1 - Gadri, S., & Adouane, N. E. (2022). Efficient traffic signs recognition based on cnn model for self-driving cars. In Intelligent Computing & Optimization: Proceedings of the 4th International Conference on Intelligent Computing and Optimization 2021 (ICO2021) 3 (pp. 45-54). Springer International Publishing.

2 - Chishti, S. O., Riaz, S., BilalZaib, M., & Nauman, M. (2018, November). Self-driving cars using CNN and Q-learning. In 2018 IEEE 21st International Multi-Topic Conference (INMIC) (pp. 1-7). IEEE.

3 - Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90.

4 - Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards AI. Large-scale kernel machines, 34(5), 1-41.

5 - Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Carbin, M., & Wang, Z. (2021). The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 16306-16316).

6 - Kaggle. (n.d.). GTSRB - German Traffic Sign Recognition Benchmark (Version 1.0.0) [Data set]. Retrieved from https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign?select=Train.csv

7 - Dumitrescu, D., & Boiangiu, C. A. (2019). A study of image upsampling and downsampling filters. Computers, 8(2), 30.

8 - Salehi, A. W., Khan, S., Gupta, G., Alabduallah, B. I., Almjally, A., Alsolai, H., ... & Mellit, A. (2023). A Study of CNN and Transfer Learning in Medical Imaging: Advantages, Challenges, Future Scope. Sustainability, 15(7), 5930.

9 - Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

10 - He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

11 - Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

12 - Sarraf, S., & Tofighi, G. (2016). Classification of alzheimer's disease using fmri data and deep learning convolutional neural networks. arXiv preprint arXiv:1603.08631.

Material Suplementar

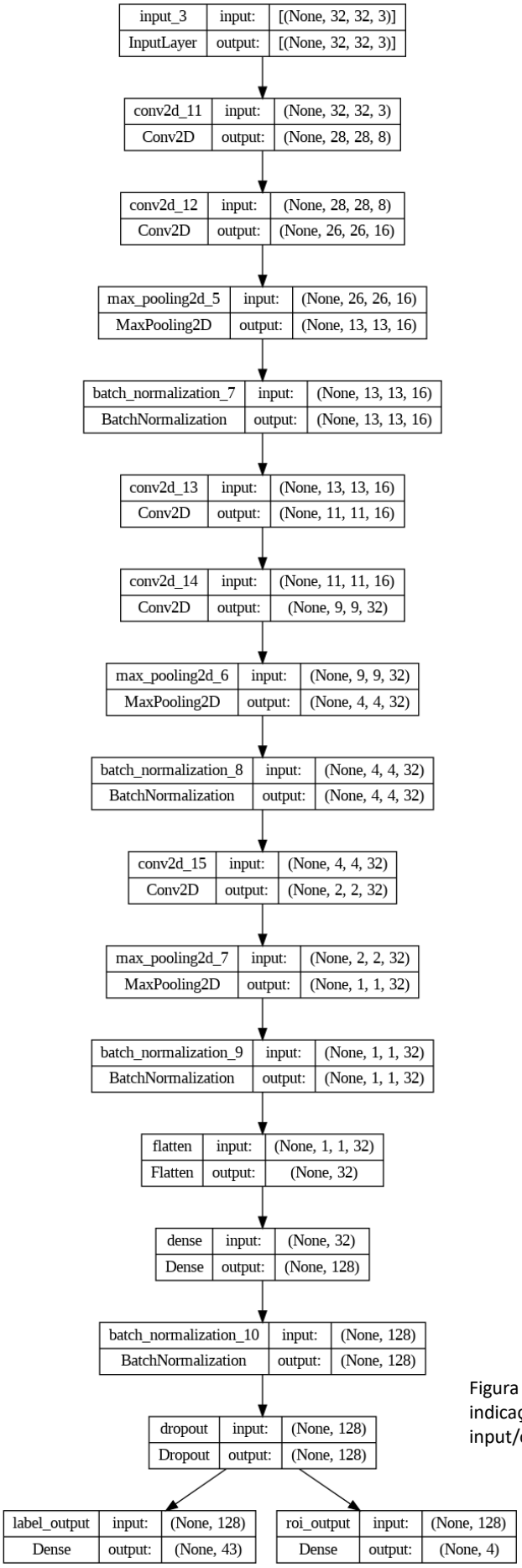

Figura S1 - Arquitetura da CNN2 com a indicação do tipo de layer e o respetivo input/output.