MULTI-CLASS CLASSIFICATION OF SUPERCAR LOGOS USING CNN

M09. Introduction to Deep Learning

Grupo 3: Ana Sofia Duarte | José Leitão | José Sousa | Luís Correia

EDIT. 03/02/2025

Introdução

O objetivo deste projeto é desenvolver e treinar uma Convolutional Neural Network (CNN) para a multi-classificação de logotipos de supercarros.
O dataset inicial encontra-se disponível no Kaggle e contém cerca de 30 imagens para cada marca de carros, sendo que estas se encontram distribuídas por 15 marcas diferentes.
Sendo que o dataset inicial apresenta uma quantidade relativamente baixa de dados para treinar estes tipos de modelos, foi realizada uma recolha de imagens destas marcas de carros de forma a aumentar o tamanho do dataset usado para treinar a CNN. O dataset foi então aumentado para cerca de 80 a 100 imagens para cada marca, para treino e cerca de 5 imagens no teste. Para além disso, foi colocada 1 imagem de cada marca numa pasta para realizar single prediction.

1. Modelo Inicial

1.1. Pré-processamento de Dados

Foi realizada data augmentation, um aumento na quantidade de dados através de transformações artificiais nas imagens de forma a ajudar o modelo a generalizar melhor as variações das imagens em questão.

Data Augmentation

- rescale (1./255) -> todos os pixéis apresentam valores entre 0 e 1;
- shear_range (0.2);
- zoom_range (0.2);
- horizontal_flip (True).

Este pré-processamento foi aplicado ao dataset de treino, sendo que no de teste apenas se aplica o rescale. Com o objetivo de garantir que o treino e o teste eram realizados no mesmo formato de imagens foi definida a dimensão (64, 64) e um total de 32 imagens em cada batch. Para além disso, tendo em conta que se trata de um problema de multi-classificação, o tipo de classificação foi definido como categorical.

- target_size (64, 64);
- batch_size (32);
- class_mode (categorical).

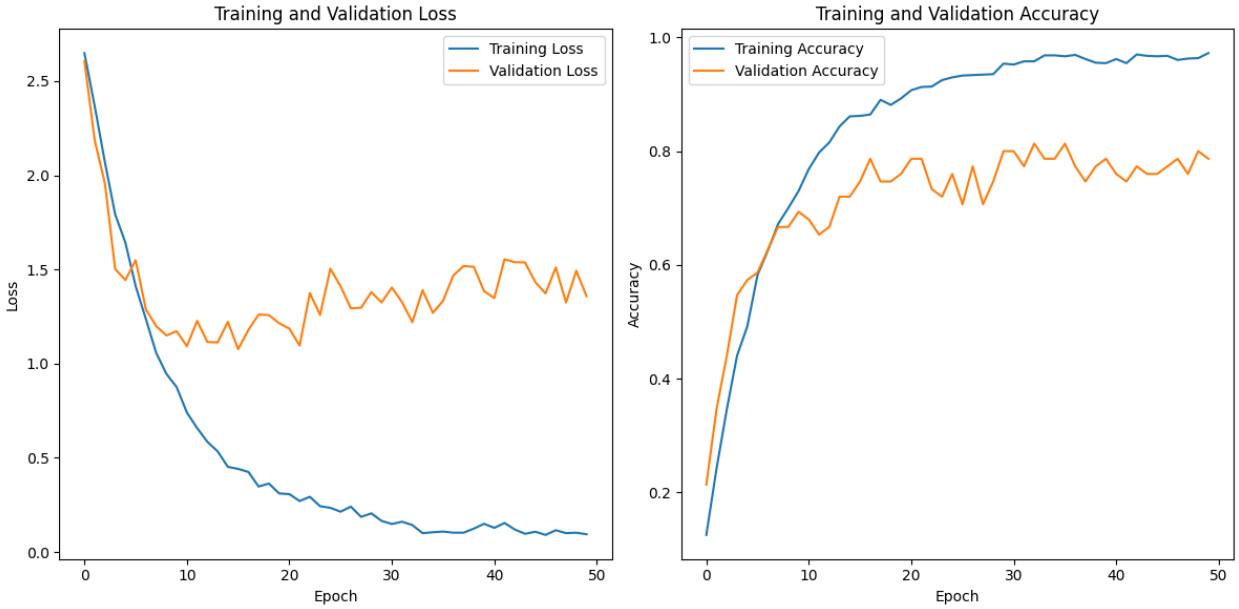
1.2. Arquitetura CNN

- Step 1: 1st convolutional layer:
    - shape (64, 64, 3);
    - filters (32);
    - kernel_size (3);
    - activation function (Relu);
- Step 2: 1st first max pooling:
    - pool_size = 2;
    - strides = 2;
- Step 3: 2nd convolutional layer:
    - filters (64);
    - kernel_size (3);
    - activation function (Relu);
    - max pooling (pool_size =2, stride =2);
    - dropout (0.25).
- Step 4: Flattening
- Step 5: Full connection (2 layers):
    - units (256);
    - activation function (Relu);
    - dropout (0.25);
    - units (256);
    - activation function (Relu);
    - dropout (0.25);
- Step 6: output layer:
    - units (15);
    - activation function (softmax).

1.3. Treino CNN

O treino do primeiro modelo foi realizado através do otimizador Adam, sendo que a loss function é calculada através de categorical crossentropy tendo em conta que o problema é de multi-classificação.

1.4. Resultados



No gráfico da Accuracy podemos ver que existe um gap entre a linha do treino e da validation, ou seja, como a accuracy do treino está muito acima do teste, podemos concluir que o modelo está em overfitting.

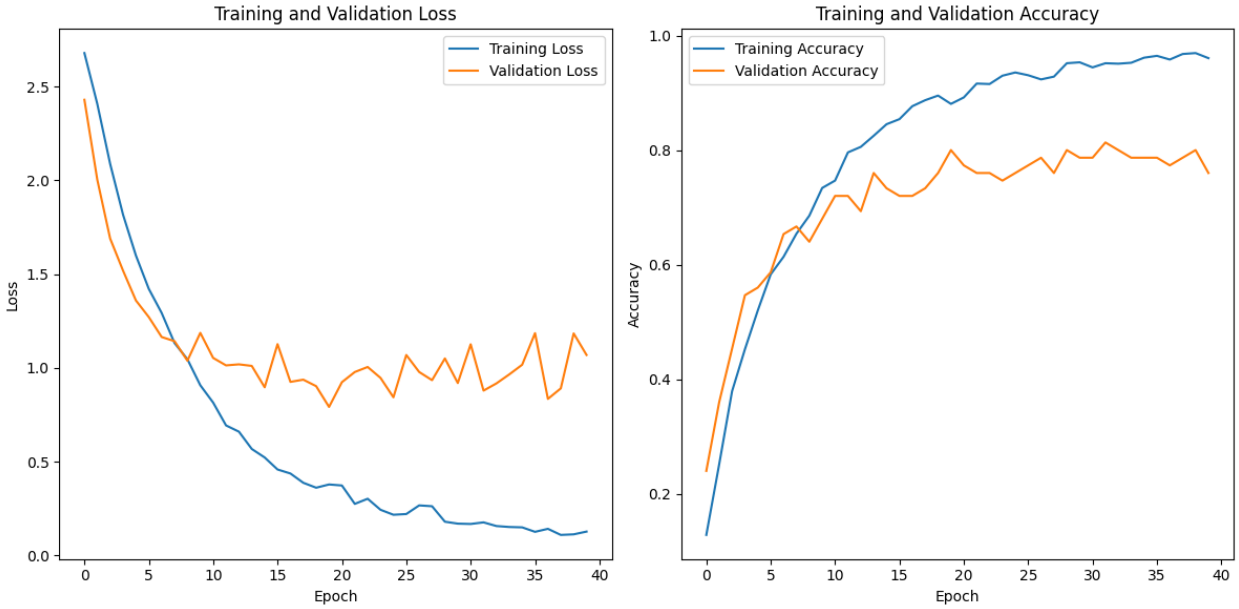
No gráfico da Loss existe também um gap entre as duas linhas, sendo que após algumas Epochs a validação começa a aumentar, ou seja, o modelo está se a ajustar demasiado aos dados de treino perdendo capacidade de generalizar.

2. Modelo 2

2.1. Arquitetura CNN

Neste modelo, foi aumentado o tamanho das imagens, com o objetivo de recolher mais informação das imagens e, tendo em conta que os resultados anteriores apresentam overfitting e uma loss function bastante instável, foi reduzido o Learning rate do otimizador para 0.0005.

2.2. Resultados



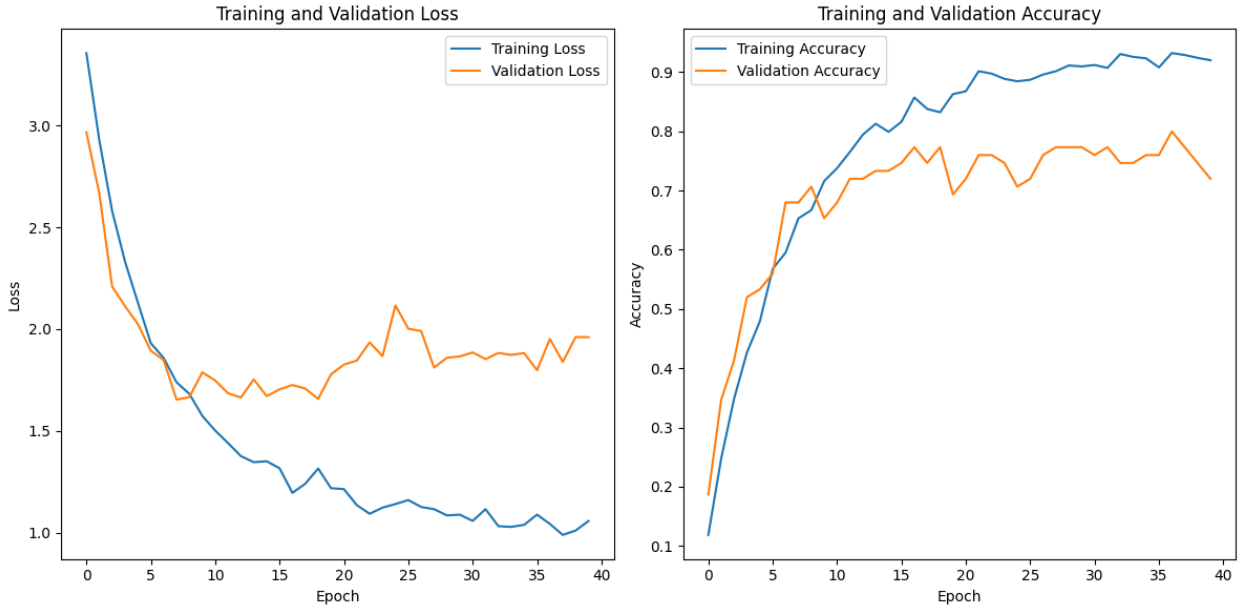
No gráfico da accuracy e no da loss podemos ver que o gap ficou mais pequeno, o que mostra que houve uma diminuição do overfitting, no entanto a sua presença ainda é bastante notória.

3. Modelo 3

3.1. Arquitetura CNN

Neste modelo foram incluídas técnicas de regularização L2 com o valor de 0.001 em ambos os layers de convulsão e full connection, com o objetivo de procurar uma maior estabilidade na loss function e rduzir o overfitting.

3.2. Resultados



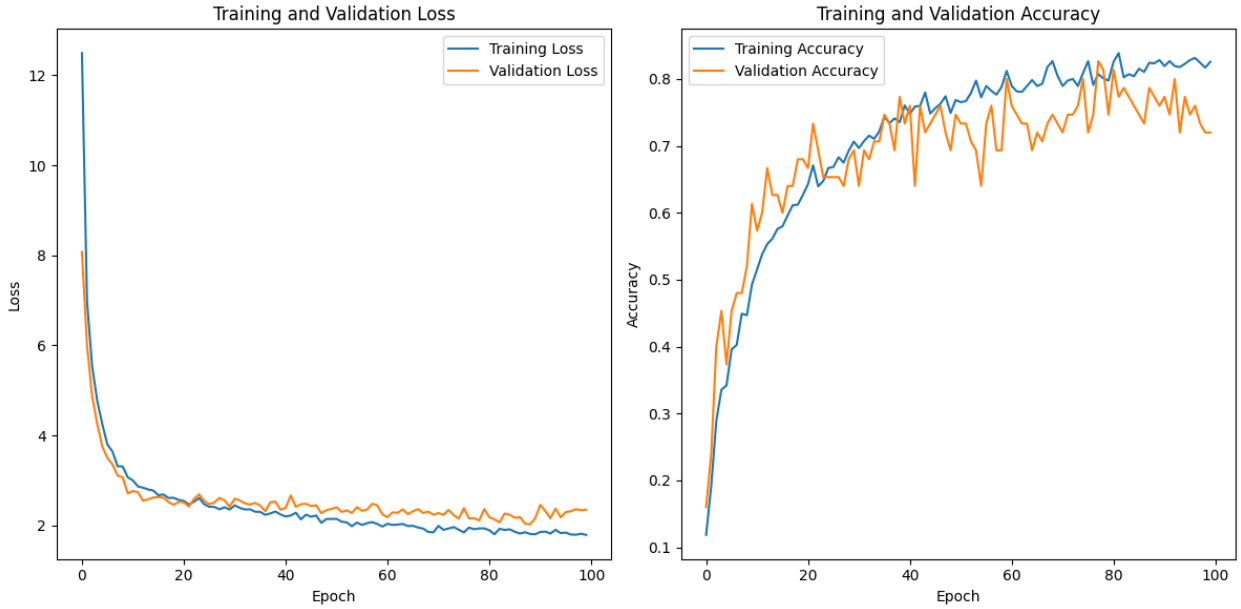
No gráfico da Accuracy e no da Loss podemos ver que o gap ficou novamente mais pequeno, o que mostra que houve uma diminuição do overfitting, no entanto a sua presença ainda é bastante notória. Para além disso, a loss function parece ter perdido alguma capacidade, não baixando de 1.0.

4. Modelo 4

4.1. Arquitetura CNN

Os filtros do primeiro e segundo layer de convulsão foram aumentados para 64 e 128, respetivamente. Para além disso, a regularização L2 aplicada no modelo anterior, foi retirada destes layers e apenas deixada na fully connection com os valores de 0.03. Foi também aplicado weight decay de 0.03 no otimizador. No final foram usadas 100 epochs.

4.2. Resultados



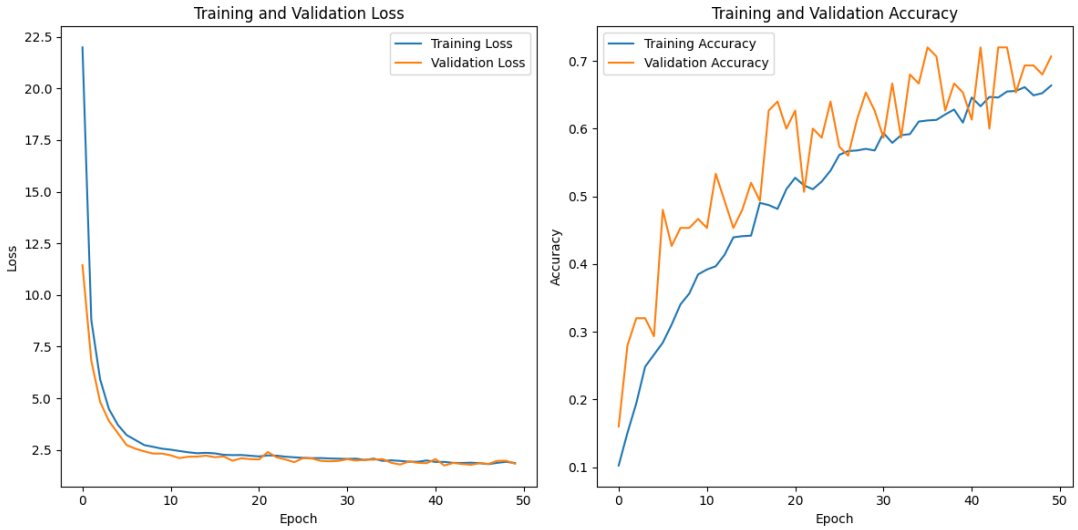
De um modo geral, podemos ver que houve grandes alterações em ambos os gráficos. No gráfico da Accuracy o gap entre as duas linhas diminuiu drasticamente, no entanto ocorreu exploding gradient. Já no gráfico da Loss podemos ver que existe uma atenuação de ambas as linhas, ainda que estas se mantenham um pouco instáveis e com valores mais altos do que os pretendidos.

5. Modelo 5

5.1. Arquitetura CNN

Neste caso, foi realizada uma redução no tamanho das imagens, novamente, para (64,64) pixéis. Aplicada uma regularização nos fully connected layers de 0.05, assim como no weight decay. Tendo em conta a redução do tamanho de imagens, foi decidido reduzir o número de filtros nas Convolutional layers para os valores originais e aumentar o tamanho do kernel no primeiro.

5.2. Resultados



Estes resultados mostram um aumento na estabilidade do comportamento da loss function, sublinhando que o exploding gradient se manteve.

6. Modelo 6

6.1. Arquitetura CNN

Foi realizada uma redução na regularização L2 e weight decay para 0.03.

6.2. Resultados

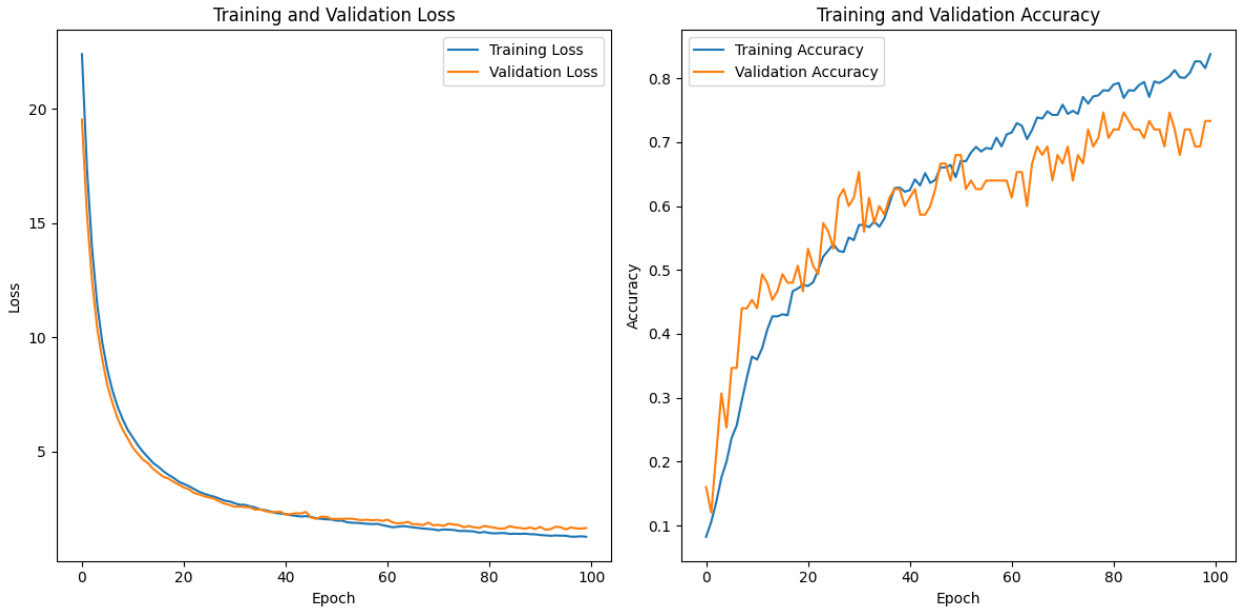
É notável a redução do exploding gradient e a loss function atingiu valores mais reduzidos.

7. Modelo 7

7.1. Arquitetura CNN

A Learning rate foi reduzida de 0.005 para 0.0001, com o objetivo de reduzir o exploding gradient e tentar reduzir o overfitting.

7.2. Resultados



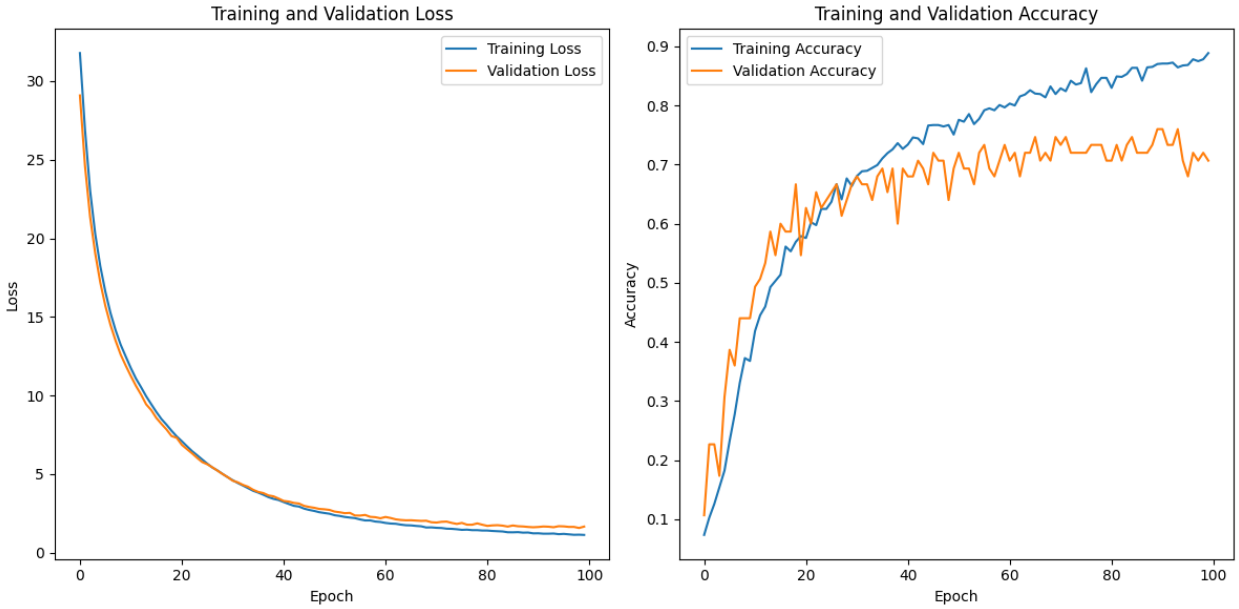
Apesar desta alteração o gradiente parece ainda muito instável e o overfitting mantém-se. Desta vez é possível notar que o valor inicial da loss function é bastante mais alto do que o anterior, ainda que tenha atingido um valor final semelhante.

8. Modelo 8

8.1. Arquitetura CNN

Com o objetivo de garantir uma inicialização mais controlada dos weights, foram utilizadas técnicas de inicialização, neste caso He Initialization nas fully connected layers. Para além disso, foi adicionado clipnorm ao otimizador com o objetivo de reduzir o exploding gradient.

8.2. Resultados



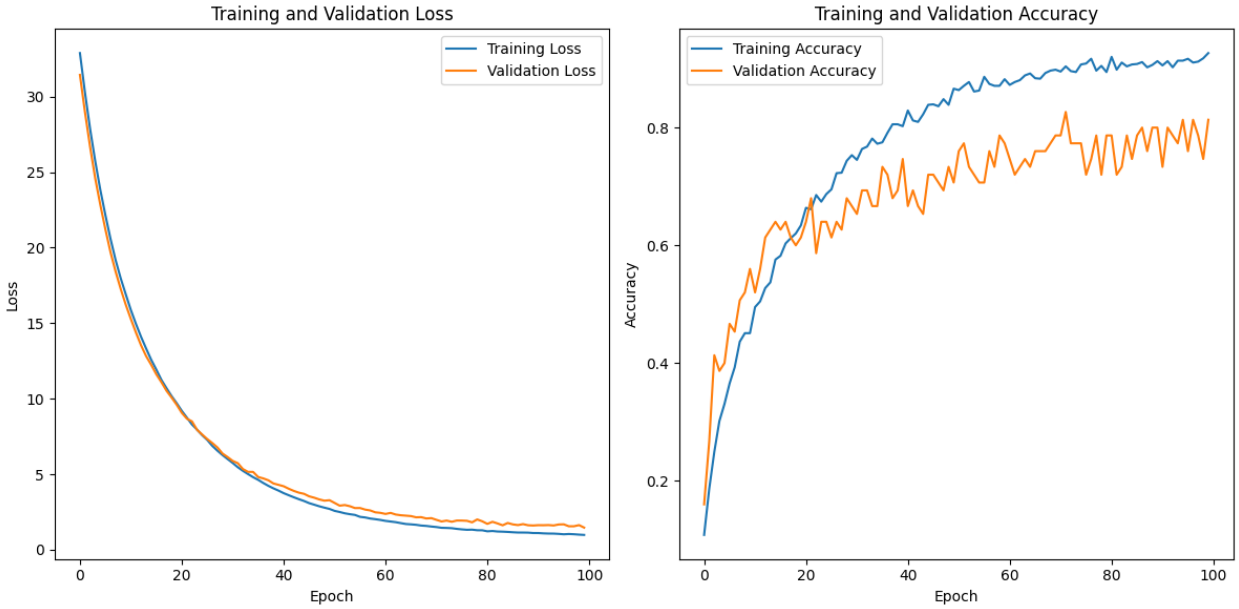
Este modelo apresenta uma redução significativa na loss function, de forma estável ainda que o overfitting se mantenha. Apesar disso, a accuracy parece ter um acompanhamento de treino e validação razoavelmente estável até aos 65%.

9. Modelo 9

9.1. Arquitetura CNN

Desta vez foi aplicada normalização dos batches através do Batch Normalization com o intuito de estabilizar as ativações da rede e acelerar o processo de treino. Para além disso, esta técnica foi utilizada com o intuito de testar se a regularização, semelhante ao dropout é capaz de reduzir overfitting.

9.2. Resultados



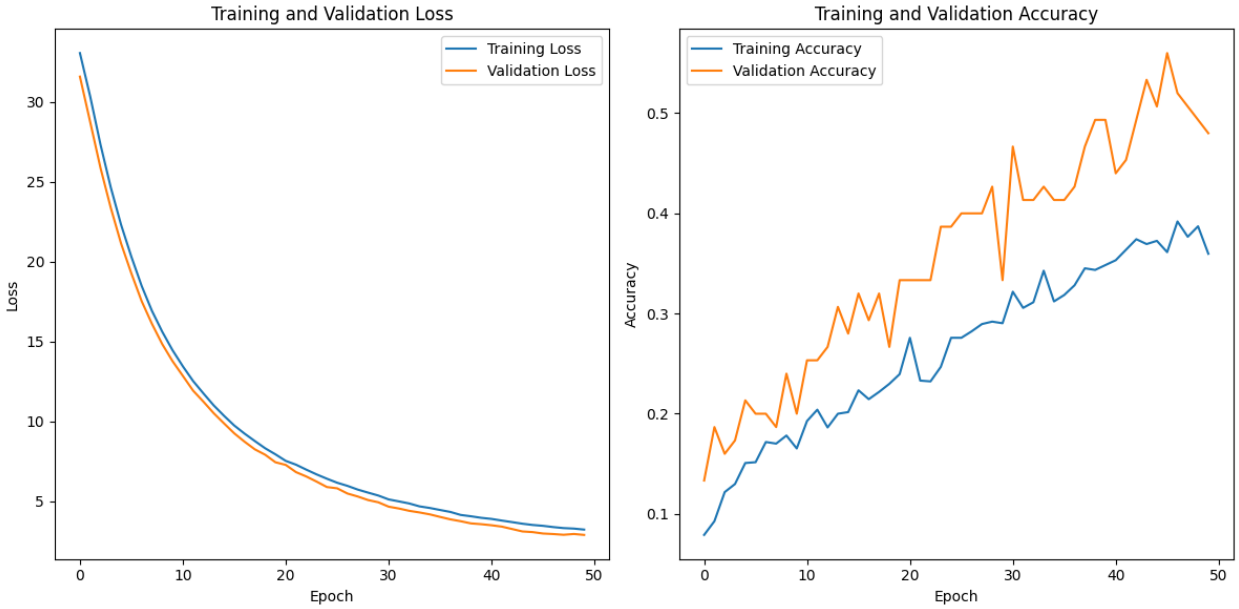
As alterações não resultaram em melhorias significativas. Pelo contrário, a loss function apresenta um gap maior a partir de 5.0.

10. Modelo 10

10.1. Arquitetura CNN

Desta vez foi realizado um aumento na data augmentation, adicionando técnicas de rotação e variações de altura e largura. Para além disso, os valores dos dropouts foram aumentos de 0.25 para 0.35 e 0.30 para 0.40.

10.2. Resultados



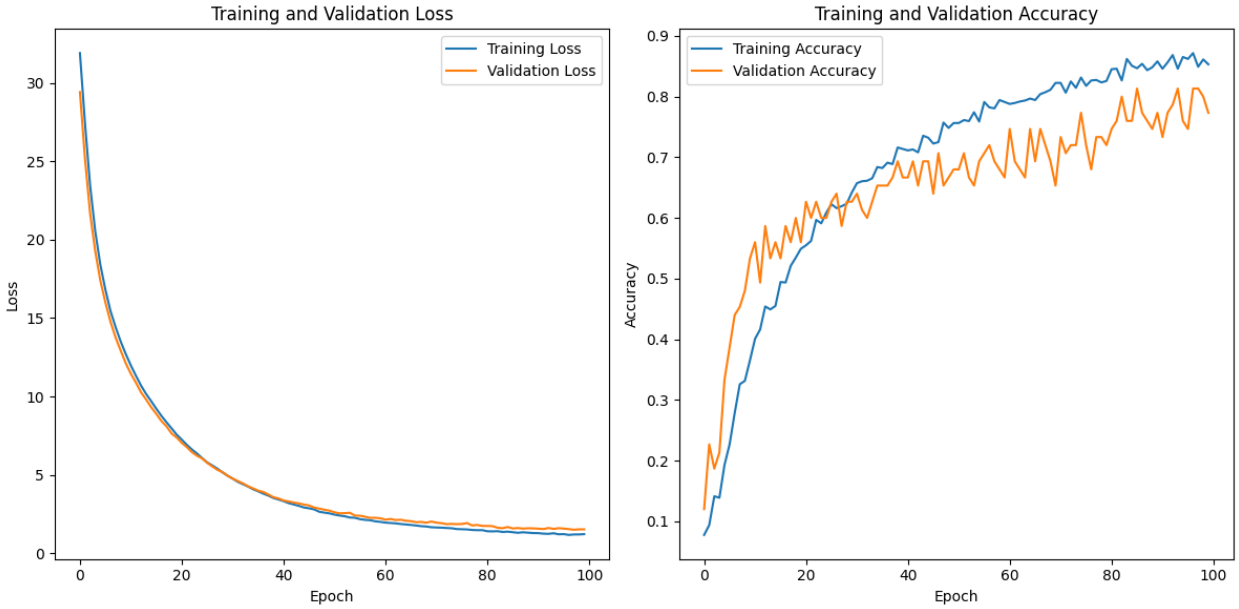
Pelos gráficos é possível notar uma redução na aprendizagem do modelo, visto que a curva e accuracy apresenta uma subida muito mais lenta do que os modelos anteriores. Seria de esperar uma subida mais rápida.

11. Modelo 11

11.1. Arquitetura CNN

Tendo em conta os maus resultados da tentativa anterior, foram eliminadas as alterações realizadas relativamente à data augmentation e reduziu-se o dropout aplicado ao primeiro layer da rede fully connected.

11.2. Resultados



Desta vez o modelo voltou a apresentar comportamentos mais positivos, sendo que ainda é possível reconhecer overfitting. Este modelo continua sem ultrapassar a accuracy de 70% na validação.

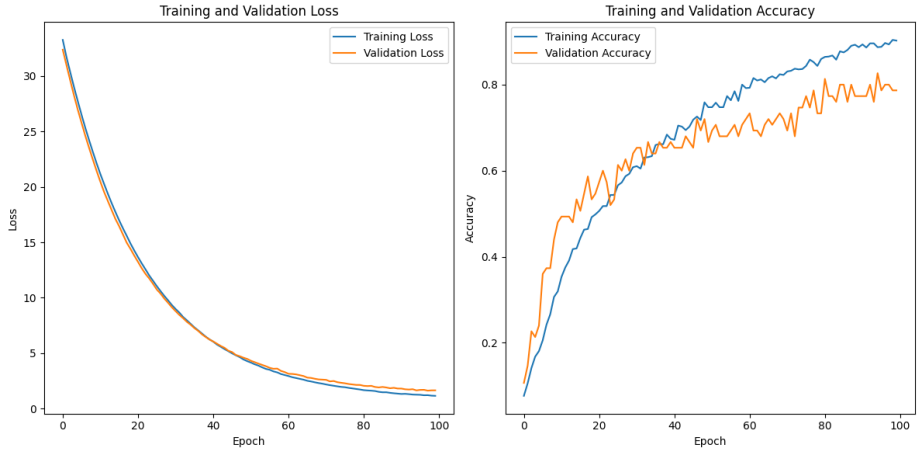
12. Modelo 12

Depois do modelo 11 tomou-se a decisão de testar optimizadores diferentes e tentar perceber o comportamento de cada um.

12.1. Arquitetura CNN

Neste caso o otimizador Adam é substituído por SGD (Stochastic Gradient Descent) com Learning rate 0.001 e momentum 0.9. O weight decay foi eliminado.

12.2. Resultados



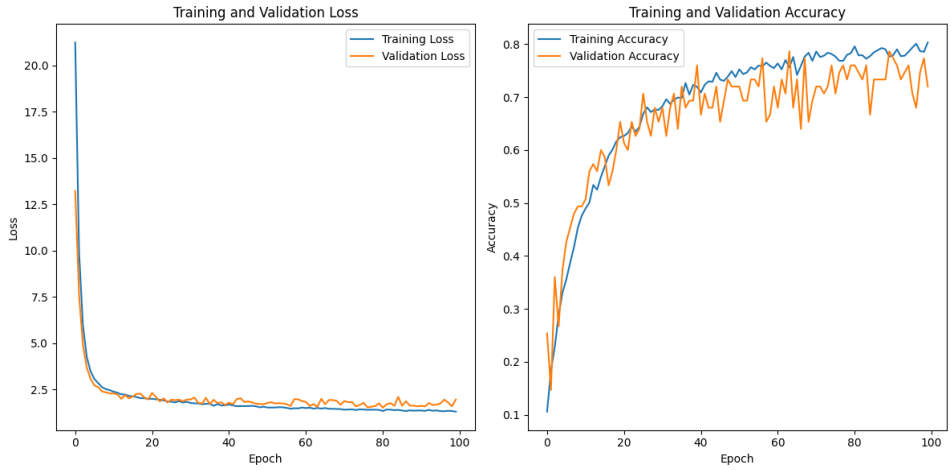
Com este otimizador o modelo não apresentou alterações significativamente distintas dos anteriores.

13. Modelo 13

13.1. Arquitetura CNN

Desta vez, foi utilizado o otimizador RMSprop em troca do SGD. A Learning rate foi mantida em 0.001, foi aplicado um rho de 0.9 e eliminado o momentum.

13.2. Resultados



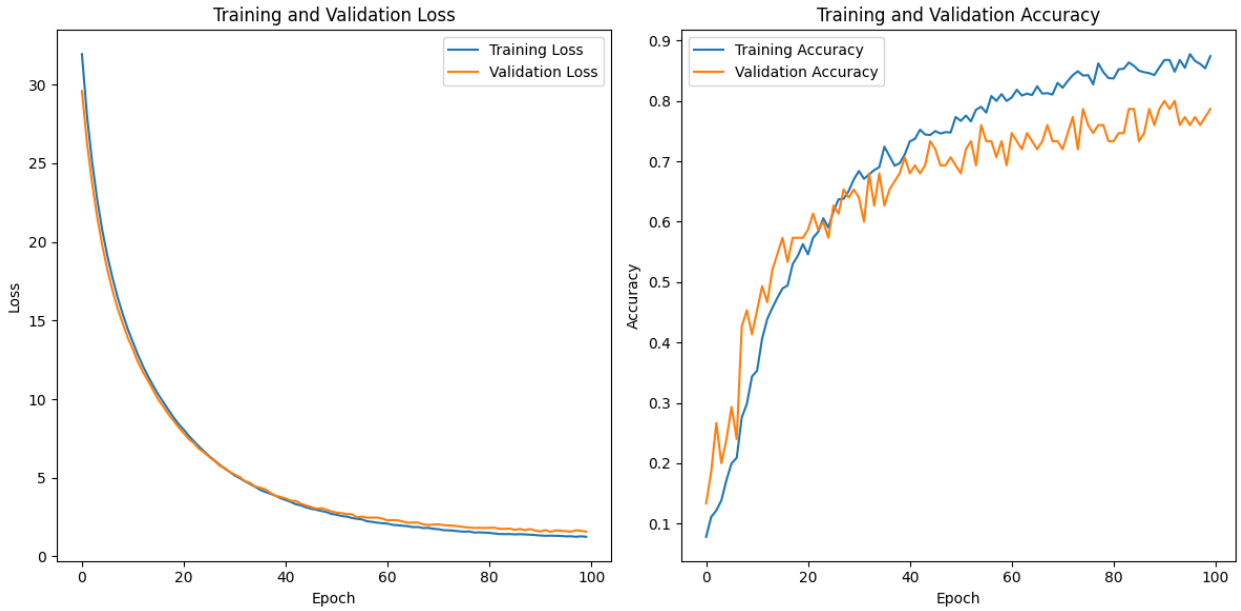
Este otimizador apresenta uma loss curve mais instável e com valores mais elevados. Apesar disso, a accuracy foi capaz de atingir os valores medidos nos modelos anteriores, ainda que com exploding gradient mais acentuado.

14. Modelo 14

14.1. Arquitetura CNN

Substituição do otimizador RMSprop por AdamW. Learning rate reduzida para 0.0001, voltou-se a adicionar weight decay de 0.03 e eliminou-se rho e momentum.

14.2. Resultados



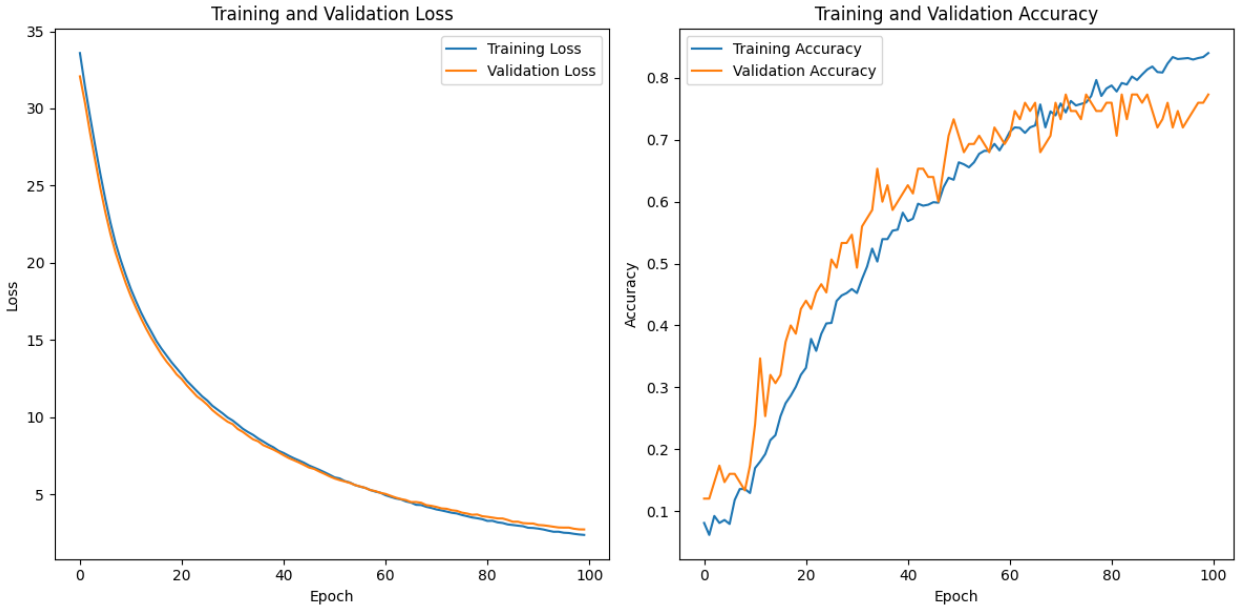
Este otimizador voltou a apresentar um comportamento mais satisfatório, ainda que a forma como este modelo generaliza os dados continua a não apresentar melhorias.

15. Modelo 15

15.1. Arquitetura CNN

Tendo em conta a dificuldade em generalizar e o facto de nenhum dos optimizadores não apresentar melhorias comparativamente ao Adam, neste modelo voltou-se a testar o mesmo otimizador mas, desta vez, aumentando a profundidade da Convolutional network para 3 layers. O terceiro layer é constituído por 64 filtros, um kernel de tamanho 3, normalização dos batches e a função de ativação relu. Para além disso, o max pooling foi construído com um pool size 2 e 2 strides. Complementou-se com um dropout de 0.3. O tamanho das imagens em análise foi aumentado para 128 pixeis e o Learning rate definido com 0.00005.

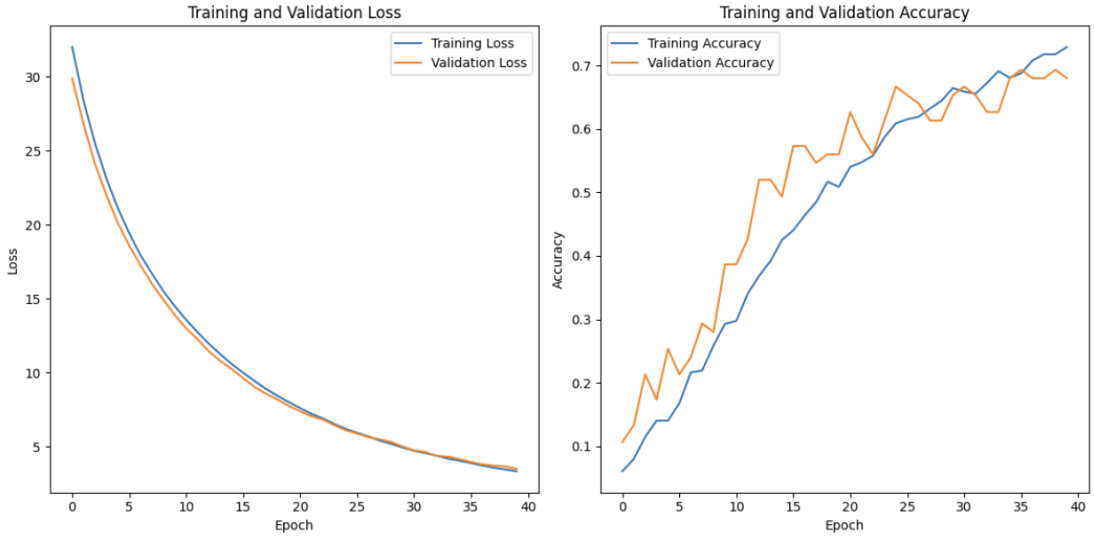
15.2. Resultados



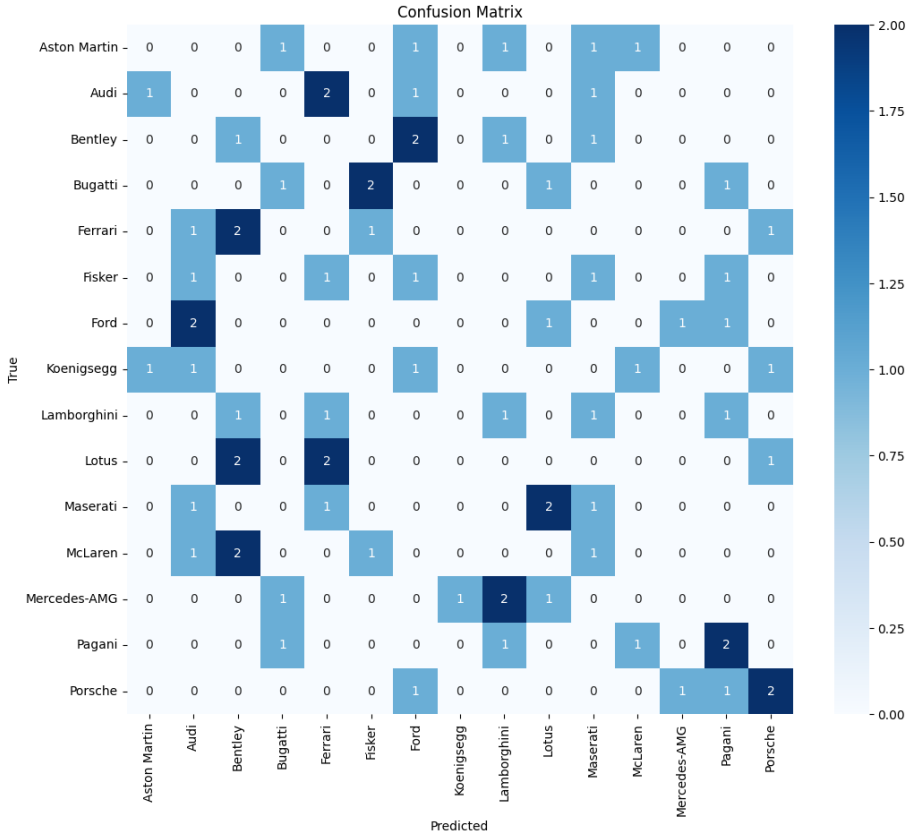
Nesta attempt podemos ver que o modelo teve um pior desempenho. No gráfico da Accuracy nota-se um crescimento lento e instável de ambas as curvas. Para além disso, os valores da loss function são mais elevados do que o pretendido.

Conclusão

Concluímos que o modelo com melhor desempenho foi o executado na attempt 14, devido à curva da loss function e à curva da accuracy. Desta forma, de seguida, o código da attempt 14 é executado novamente mas apenas para 40 epochs, de forma a diminuir o overfitting.



De seguida está representada a matriz de confusão referente a essa attempt.



A análise da matriz de confusão permite tirar vários insights do modelo CNN da attempt 14. Podemos verificar que há uma confusão significativa entre várias marcas, isto porque pode haver uma dificuldade por parte do modelo em diferenciar marcas com logos semelhantes. No entanto, também podemos reparar que para as marcas Bentley, Bugatti, Lamborghini, Maserati, Pagani e Porsche, o modelo conseguiu prever corretamente 50% das vezes ou mais (Pagani e Porsche).

Concluímos que, para além do modelo apresentar uma accuracy de 74% no treino e 68% no teste, ainda era possível melhorá-lo para obter valores de previsão mais precisos. Para isso poderíamos tentar as seguintes abordagens:

- Aumentar o dataset;
- Adicionar mais transformações às imagens do dataset;
- Testar novos hiperparâmetros.