**Instituto Superior Técnico**

MEEC

Machine Learning

# Lab 1

## Linear Regression

**Group 9**

Manuel Diniz, 84125
Alexandre Rodrigues, 90002

**Turno:** 4ªf 11h00

# Contents

# Chapter 1

# Pre-processamento dos dados

De modo a melhor se enquadrarem ao tipo de redes neuronais a usar, os dados são alterados de forma a se obter valores para cada pixel de 0 a 1 em *floating point*, ao invés dos 0 a 255 em *uint8*. Valores normalizados adequam-se melhor a redes neuronais, pelo que se divide por 255. Para além disto converte-se a *label* de cada imagem para representação *one-hot*, um formato mais uma vez mais adequado para os modelos a usar.

# Chapter 2

# *Multilayer perceptron*

# Chapter 3

# *Convolutional neural network*

É agora criado o modelo de uma *CNN*, com a arquitetura especificada. Este modelo é treinado por um máximo de 200 *epochs*, e programado para parar mais cedo se não existirem melhorias na aprendizagem.

O *calback* de *early stopping* tem como objetivo evitar que o modelo fique *overfit*, pelo que é muito importante que este se baseie na métrica correta para decidir quando parar a aprendizagem e restaurar os melhores pesos. A métrica a escolher é claramente *val_loss*, ou *loss* de validação, isto porque é a métrica que dá uma avaliação da *performance* do modelo com dados com qual este não treinou. Se fosse usado, por exemplo, a métrica *loss*, que diz respeito aos dados de treino, o modelo iria tornar-se significativamente *overfit*.

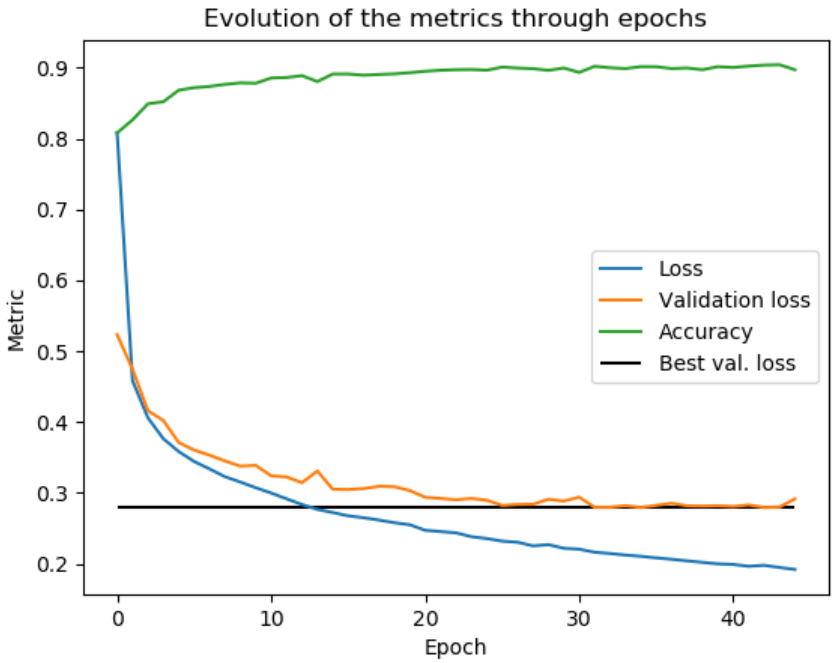


Figure 3.1: Evolução das métricas ao longo dos *epochs*

Como se pode observar, a *loss* continua a diminuir muito depois da *validation loss* estabilizar.

Observa-se ainda uma pequena subida da *validation loss* junto aos últimos *epochs*, antes do *early stopping* ter parado a aprendizagem. Nesta altura o modelo estava a tornar-se *overfit*, melhorando a *performance* nos dados de treino ao custo da nos dados de validação.

A validação final com os dados de teste produz uma *accuracy* de 0.9016, e a *confusion matrix* seguinte:

| Label | T-shirt | Trouser | Pullover | Dress | Coat | Sandal | Shirt | Sneaker | Bag | Ankle boot |
|---|---|---|---|---|---|---|---|---|---|---|
| T-shirt | 870 | 0 | 24 | 25 | 2 | 1 | 71 | 0 | 7 | 0 |
| Trouser | 1 | 971 | 1 | 21 | 2 | 0 | 2 | 0 | 2 | 0 |
| Pullover | 18 | 0 | 868 | 7 | 37 | 0 | 67 | 0 | 3 | 0 |
| Dress | 16 | 2 | 16 | 906 | 28 | 0 | 27 | 0 | 5 | 0 |
| Coat | 1 | 1 | 53 | 19 | 845 | 0 | 77 | 0 | 4 | 0 |
| Sandal | 0 | 0 | 0 | 0 | 0 | 971 | 0 | 21 | 0 | 8 |
| Shirt | 153 | 0 | 68 | 21 | 56 | 0 | 685 | 0 | 17 | 0 |
| Sneaker | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 974 | 0 | 17 |
| Bag | 3 | 1 | 8 | 4 | 4 | 2 | 7 | 5 | 965 | 1 |
| Ankle boot | 1 | 0 | 0 | 0 | 0 | 4 | 0 | 34 | 0 | 961 |

Como se pode observar, o modelo é robusto na identificação dos objetos. O maior volume de enganos ocorre na identificação de peças de roupa semelhantes, como *T-shirts* e *shirts*, o que é razoável, tendo em conta que os seus formatos são parecidos.

Observando agora as ativações das camadas de convolução para uma imagem exemplo, obtém-se:

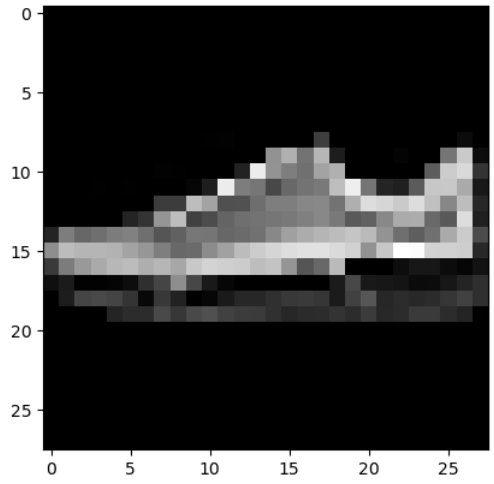


Figure 3.2: Imagem de teste

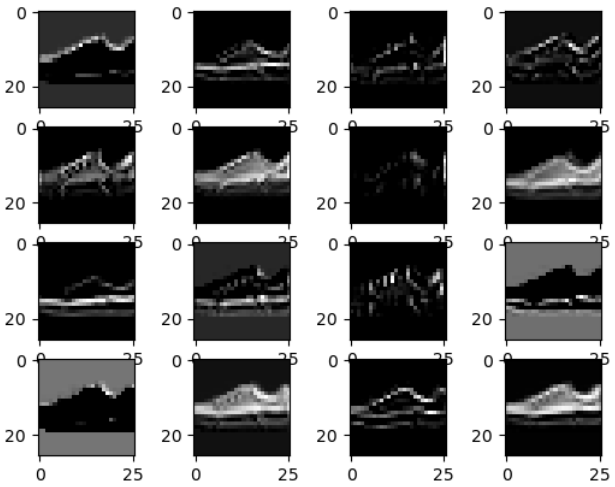

Figure 3.3: Ativação da primeira camada convolucional

Há uma certa dificuldade em tentar entender as *features* que cada canal capta, pois uma rede neuronal nem sempre opera do modo que imaginamos, e há uma certa tendência de impor a nossa lógica ou forma de pensar sobre o modelo, que pode não ser correto. No entanto, parece ser possível extrapolar que o canal 12 (começando a contar do 0) e talvez o 11 extraem, por exemplo, o formato geral do sapato.

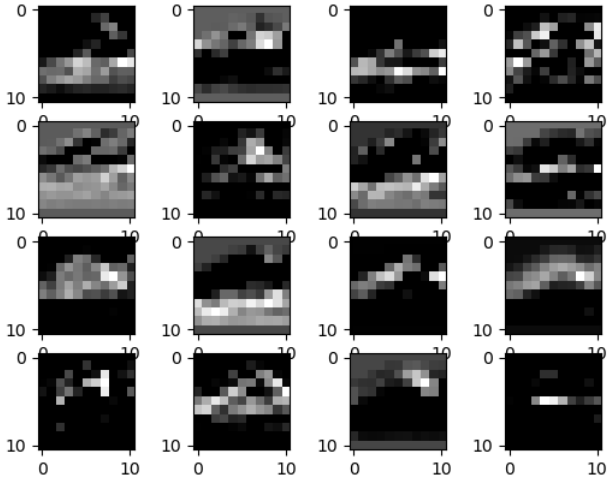

Figure 3.4: Ativação da segunda camada convolucional

A segunda convolução já não permite, através de um olho humano, entender minimamente o processo que o modelo usa, ou que *features* está a identificar. Algumas ativações parecem realçar as bordas do objeto, mas é difícil dizer ao certo.