Object Recognition with different Machine Learning models

Lorenzo Labarta Arilla

2017022

Coursework: Object Recognition

CS-345: Big Data and

Machine Learning

22nd December 2022

Contents

List of Figures

List of Tables

1 Introduction

The project's main task was to develop several Machine Learning algorithms to determine which performs better when recognising objects in a Data set of different images with different labels such as aquatic mammals, fish, flowers, and food containers... For humans this action seems easy, however, a Machine Learning model needs to understand the relationship in those images from a non-existing "previous knowledge" to end with results that fit the minimum standard for us to automatize jobs and benefit from that.

Since we want to recognise patterns in those images I thought that Deep Neural Networks would be the best option for the problem, without any doubt and a single search on Google confirmed that[4] in this case Convolutional Neural Networks, CNN, is the algorithm that will adapt best. So I chose to implement a Deep Neural Network and a Convolutional Neural Network as my final choice.[1]

Table 1: Result table based on the accuracy of the model

Data set	Neural Network	Convolutional Neural Network
Label Fine	26.25%	**32.27**%
Label Coarse	37.12%	**45.44**%

2 Methods

I focused on NN and CNN because I knew they were the best-performing algorithms. Choosing CNN as my main method to take into account and the one I am going to explain.

The idea of Neural Networks is to mimic the working of the human brain, on the other side Convolutional Neural Networks work as the part of the brain that uses the visual sensory organs to recognise different objects.

2.1 How features are extracted? and feature processing

Convolutional neural networks use a type of feature processing called convolution. In convolutional layers, the network applies a set of kernels to the input data. Each kernel is a matrix that slides over the input data, in my case a layer of 32 and two of 64 kernels with a 3x3 kernel size, performing an element-wise multiplication and summing the results to produce a single output value. The output of the convolution operation is a set of values called feature map. I applied multiple filters to the input data, so it's able to extract multiple features simultaneously, the combination of these features allows the CNN to learn more complex patterns in the data to detect objects.[2]

I introduced non-linearity to the model so it can learn more complex patterns with non-linear activation functions, such as ReLU (Rectified Linear Unit). The output of the activation function is then passed through a pooling layer, which reduces the spatial dimensionality of the data by applying in my case a max pooling function. That helps to reduce the number of parameters in the model which makes it less prone to overfitting and faster to train, I specified that this max pooling operation has a pooling size of 2x2 and a stride of 2 so it takes the maximum value from a 2x2 window of pixels and places it in a new feature map reducing the size by half repeating this for each window of pixels in the feature map.

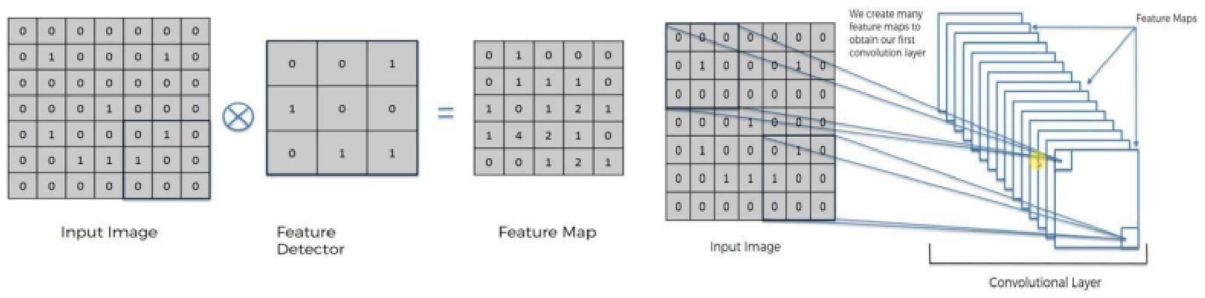


Figure 1: Convolution to produce an Activation Map & Convolution layer [1]

2.2 Classifier

After the convolutional and pooling layers and flattening the output, I included fully connected layers, FC, in the CNN, which perform a linear operation on the input data followed by an activation function. The output of the FC layers is a vector of values representing the probability of each class in a classification task.

2.3 Training and Testing

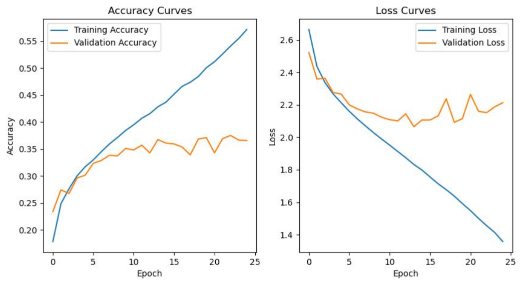
Since the dataset was already big enough and it didn't need any preprocessing like normalizing pixel values I directly chose a network architecture, I chose an architecture that is similar to the ones we used in the labs, and after that, I defined the loss function and the optimization algorithm to measure how well the CNN is performing and the optimization algorithm to adjust the weights to minimize the loss function, Sparse Categorical Crossentropy and Stochastic gradient descent respectively.

I implemented for more effective work callback in the model so it saves the model at its best result for the metrics I gave it to monitor, in my case the "*val sparse categorical accuracy*" so I do not have to fit it every time I modify my code.

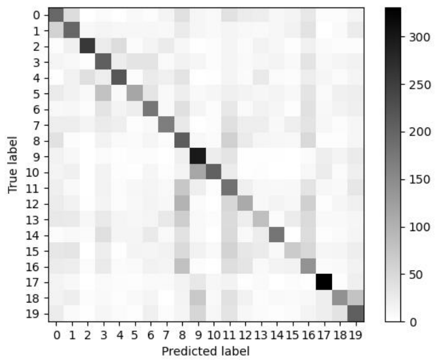
Now I can finally train it by fitting the training dataset,*trnImage* and *trnLabel fine/coarse*. To test it you need to evaluate it on the test dataset, *tstImage* and *tstLabel fine/coarse*, in order to determine how well it generalizes to new examples, for this I used the evaluate method.

3 Results

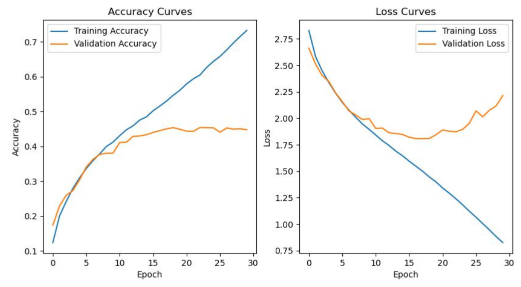
To evaluate which model performed the best I used the same metric from the labs, the accuracy because it measures the percentage of correct predictions by the CNN. In addition to that, we were asked to include the confusion matrix, CM, a table that shows the number of true positive, true negative, false positive, and false negative predictions made by the CNN. It can be useful for understanding the types of errors that the CNN is making and for identifying patterns in the data.[4]



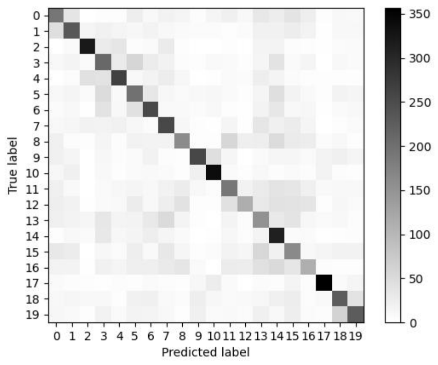
(a)



(b)

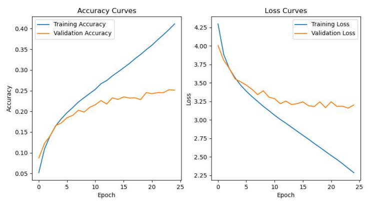


(c)

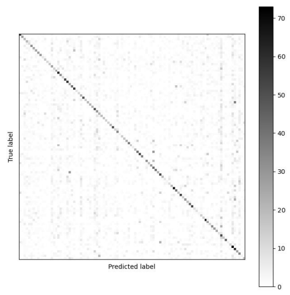


(d)

Figure 2: (a) accuracy and & loss curves of NN (b) Confusion matrix of NN (c) accuracy and & loss curves of CNN (d) Confusion matrix of CNN **coarse labels**

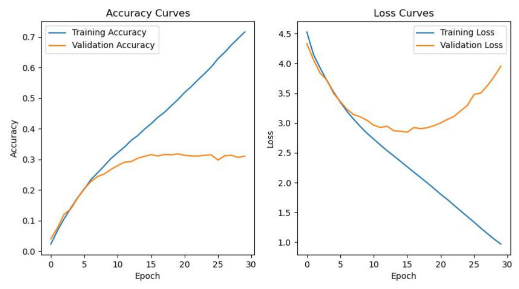


(a)

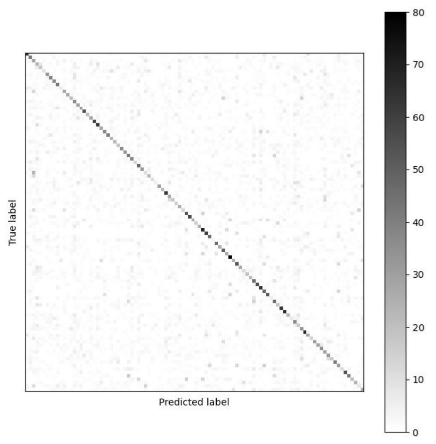


(b)

Figure 3: (a) accuracy and & loss curves of NN (b) Confusion matrix of NN **fine labels**



(a)



(b)

Figure 4: (a) accuracy and & loss curves of CNN (b) Confusion matrix of CNN **fine labels**

3.1 Table of results/quantitative results

The actual results for the test dataset were:

Table 2: Result table based on the accuracy of the model

Data set	Neural Network	Convolutional Neural Network
Label Fine	26.25%	**32.27%**
Label Coarse	37.12%	**45.44%**

4 Conclusion

NN does not have the same structural properties as CNN and is not as well-suited for processing data with a grid-like topology, such as an image. NN does not have the ability to learn local spatially-invariant features, which makes them less robust to small translations and deformations in the input image. NN does not have the ability to learn hierarchical representations of the input data, which limits its ability to learn complex patterns in the data.NN typically require a much larger number of parameters and more data to achieve good performance on image recognition tasks, making them less efficient and more prone to overfitting. NN may be more difficult to design and implement compared to CNN, especially for large and complex datasets. Overall, CNN is generally considered to be a better choice than NN for image recognition tasks because of their ability to learn local and hierarchical features effectively. However, CNN may require more design and implementation effort and more computational resources to train, especially for large and complex datasets.[3][5]

References

[1] International Conference on Emerging Trends in Information Technology and Engineering. *A Review of Convolutional Neural Networks*. (2020. URL: https://www.mriquestions.com/uploads/3/4/5/7/34572113/icetite049pid6395729.pdf.

[2] Keras. *Conv2 API*. (2022. URL: https://keras.io/api/layers/convolution_layers/convolution2d/.

[3] Niklas Lang. *Using Convolutional Neural Network for Image Classification*. (2021. URL: https://towardsdatascience.com/using-convolutional-neural-network-for-image-classification-5997bfd0ede4.

[4] Oxbowerce. *val sparse categorical accuracy*. (2021. URL: https://datascience.stackexchange.com/questions/103665/val-sparse-categorical-accuracy.

[5] Saining Xie. *Exploring Randomly Wired Neural Networks for Image Recognition*. (2021. URL: https://openaccess.thecvf.com/content_ICCV_2019/papers/Xie_Exploring_Randomly_Wired_Neural_Networks_for_Image_Recognition_ICCV_2019_paper.pdf.