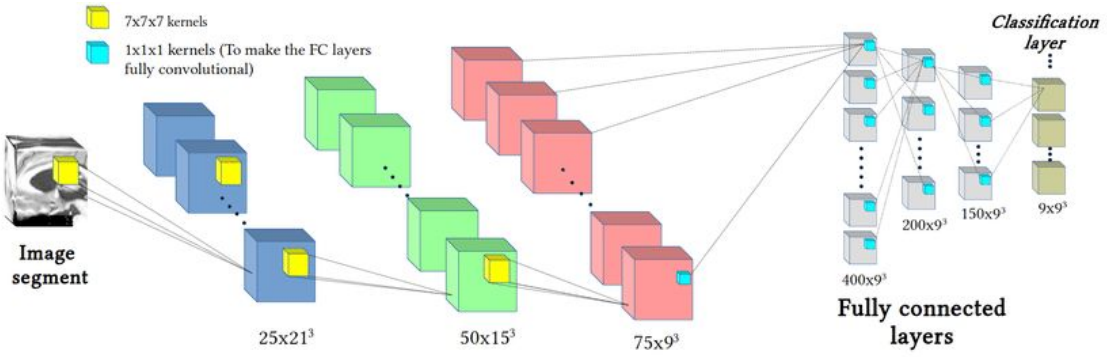


[1]

Venice Boats Classification using Fully Connected Convolutional Neural Networks

HOMEWORK 2

Sayo Michael Makinwa [Matricola: 1858908] | Machine Learning | 29.12.2018

Sapienza University of Rome

1.0 Introduction

Image recognition and classification is one part of Machine Learning that has seen a lot of momentum over the last few decades. This momentum is largely due to the recent success of Neural Networks during the same time frame.

This is a report about applying fully connected convolutional neural networks to image classification, specifically images of boats from the MarDCT dataset.

To allow for fast processing and to be able to conduct a number of experiments, three sub-categories of the datasets [Mototopo, Patanella, and VaporettoACTV] are used. The experiments conducted were majorly to vary the number of convolutional layers and the number of epochs. Overall, it was…

2.0 Methodology

2.1 Data Preprocessing

2.1.1 The Dataset

The dataset consists of training and testing data. The training data are sub grouped into 24 classes, while the testing data are not grouped as such, instead, a ground truth file containing the filename and the class each image belongs to is supplied. In total, there are 4,774 images in the training data and 1,969 images in the testing data.

It was noticed that while some boat classes in the training data have large enough numbers, some have really small numbers. For this reason, and to allow for fast processing and to be able to conduct a number of experiments, three sub-categories of the datasets [Mototopo, Patanella, and VaporettoACTV] are used. These particular classes were selected because they have a relatively good enough number of images for the training model to learn, while also not been so many as to keep the training process go on for long hours since a local machine with no GPU was used for training the model.

Furthermore, since the number of images for the network has been reduced in the way described, it makes sense to use the test data also for validation as opposed to dividing the training set for validation. Hence, it was necessary (for the three

classes used) to have the test data also grouped into class folders, so the first thing that was done after downloading and uncompressing the dataset was to write code that reads the content of the ground truth file and fetch each image in the test set to the class folder it belongs to.

2.1.2 The Input Data

While for some classes, colour might be a really important feature for classification (as in the case of Ambulanza and Polizia classes), for most of the classes, shape is sufficient for classifying the boats (this is true for the three classes selected), hence, the model was kept simple by converting all the images used to grayscale, thereby making them to be of only one channel of 2 dimensional matrices as opposed to three channels of 2 dimensional matrices with full colour RGB images.

In addition, the size of each image was reduced to 150*50 from the original 800*240, relatively maintaining aspect ratio. These two steps greatly reduced the input size of the network and made it them trainable on a local computer within reasonable time, while not losing too many important features of the images.

NOTE: At the beginning of this work, I tried training all the images from all the classes, with the full sizes and full colours, using just two convolution layers - it ran for more than 24 hours before I had to terminate the process!

2.2 The Network

Generally, the network was built to have the input layer, the convolutional layer, and the fully connected layer. The convolutional layer is the part of the work where experimentation was done.

2.2.1 The Convolutional Layer

The network was built with 2 convolutional layers, 4 convolutional layers, 6 convolutional layers, and 8 convolutional layers, with results compared. A pooling layer was added after each convolutional layer to downsample the parameters and save some computation while also preventing overfitting.

All the convolutional layers were made with similar parameters; 32 (and 64 alternating) filters each with a size of 2, stride of 1, ReLU activation function.

All the pooling layers were made with kernel size of 2

2.2.2 The Fully Connected Layer

The fully connected layer has only three layers - the input layer, the hidden layer, and the output layer.

The first (input) layer of the fully connected layer has 1024 number of units and ReLU activation function as parameters. Then dropout was added immediately after as the downsampling method for regularisation of the units added, also to prevent overfitting, and it is good for reducing computational complexity. The dropout layer was given a "keep probability" of 0.8

The second fully connected layer has 3 units (same as the number of output, one-hot encoded binary values for the classes). The activation function chosen for this layer is softmax.

The output layer is a regression layer. The optimiser used for this layer is adam optimizer, learning rate is 0.001, and the loss function is categorical cross entropy

NOTE: All the parameters used for the network were based on the generally accepted good values in the state of the art

3.0 Evaluation of Results

The number of images for the network was reduced in the way described, hence the test data was also used for validation

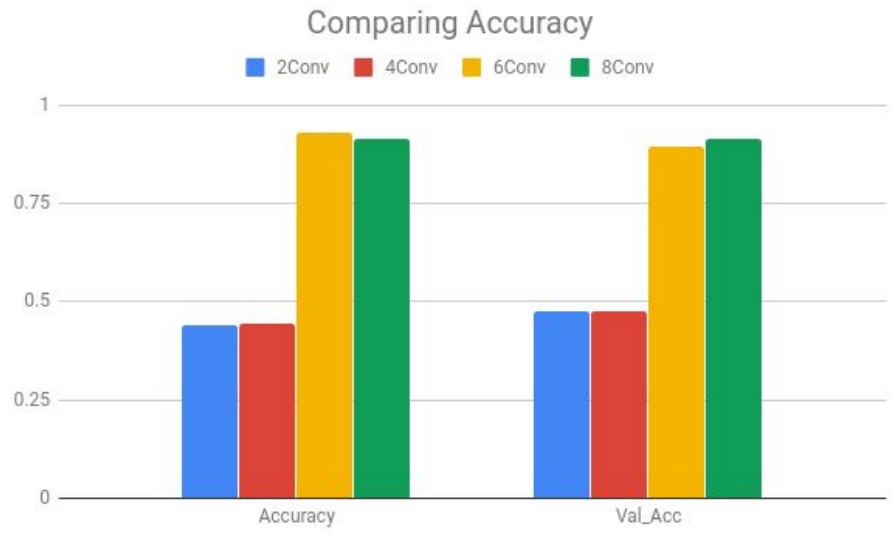
3.1 Experiment 1: Exploring the Effect of Number of Convolutional Layers

No of Epochs	10
Learning Rate	0.001
Convolutional Layers Activation Function	ReLU
Output Layer Optimizer	Adam
RUN 1	
No of Convolutional Layers	2

Final Loss Value	12.92922
Final Accuracy Value	0.4385
Final Validation Loss Value	12.06919
Final Validation Accuracy Value	0.4758
RUN 2	
No of Convolutional Layers	**4**
Final Loss Value	12.75813
Final Accuracy Value	0.4459
Final Validation Loss Value	12.06919
Final Validation Accuracy Value	0.4758
RUN 3	
No of Convolutional Layers	**6**
Final Loss Value	0.21486
Final Accuracy Value	0.9300
Final Validation Loss Value	0.37415
Final Validation Accuracy Value	0.8931
RUN 4	
No of Convolutional Layers	**8**
Final Loss Value	0.19144
Final Accuracy Value	0.9149
Final Validation Loss Value	0.22406

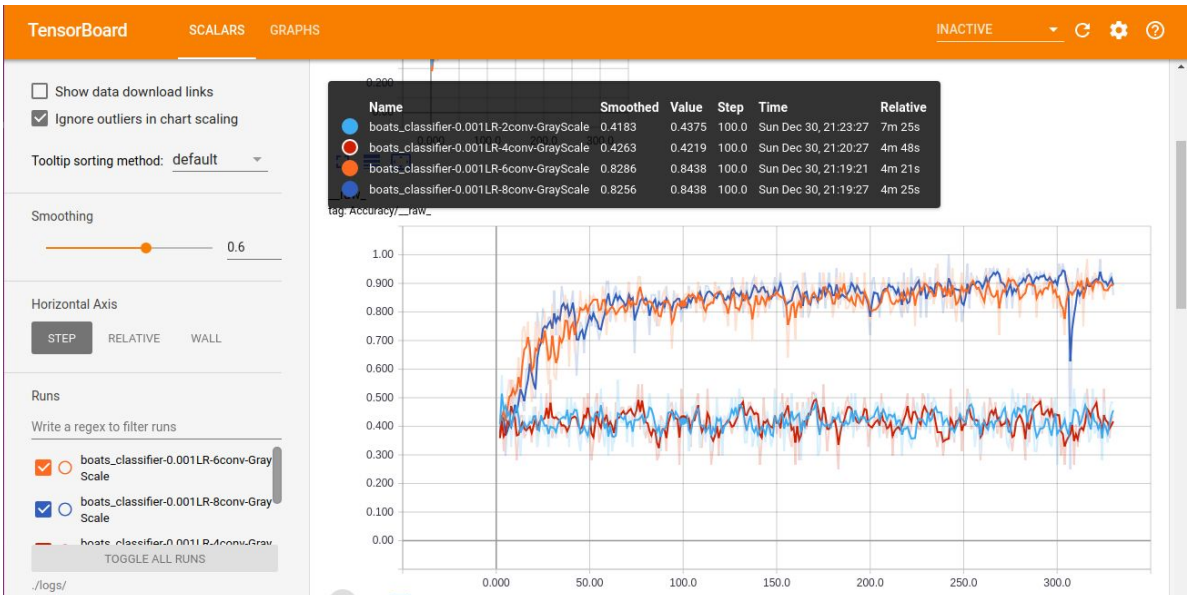
Final Validation Accuracy Value	0.9151

Result Visualisation

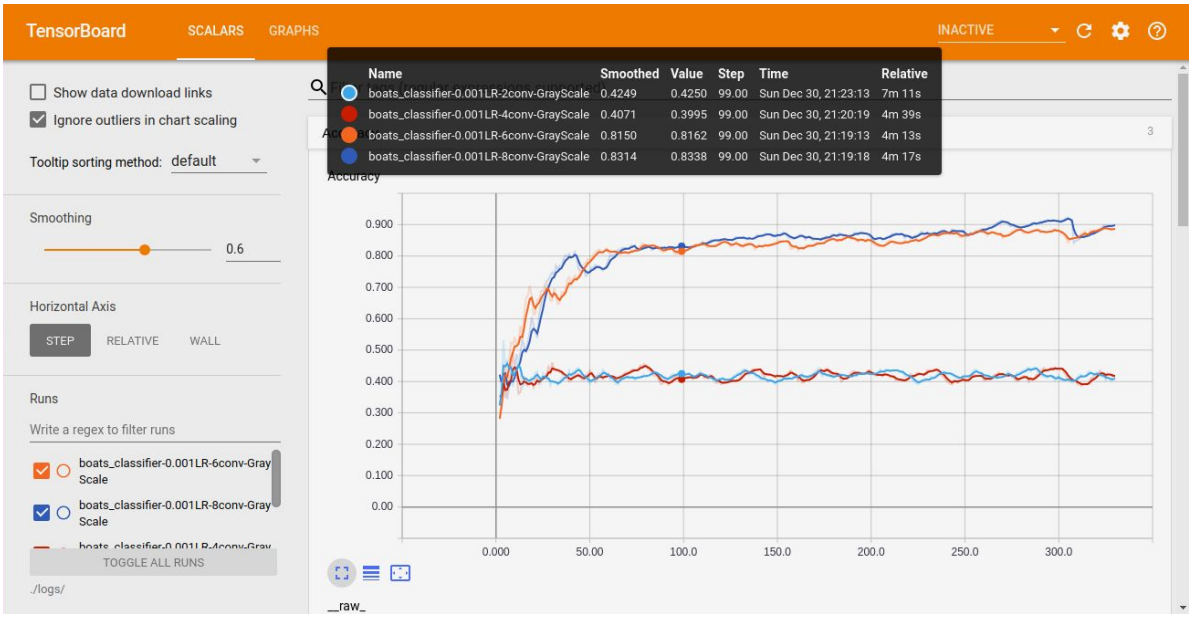


Visualisation with TensorBoard

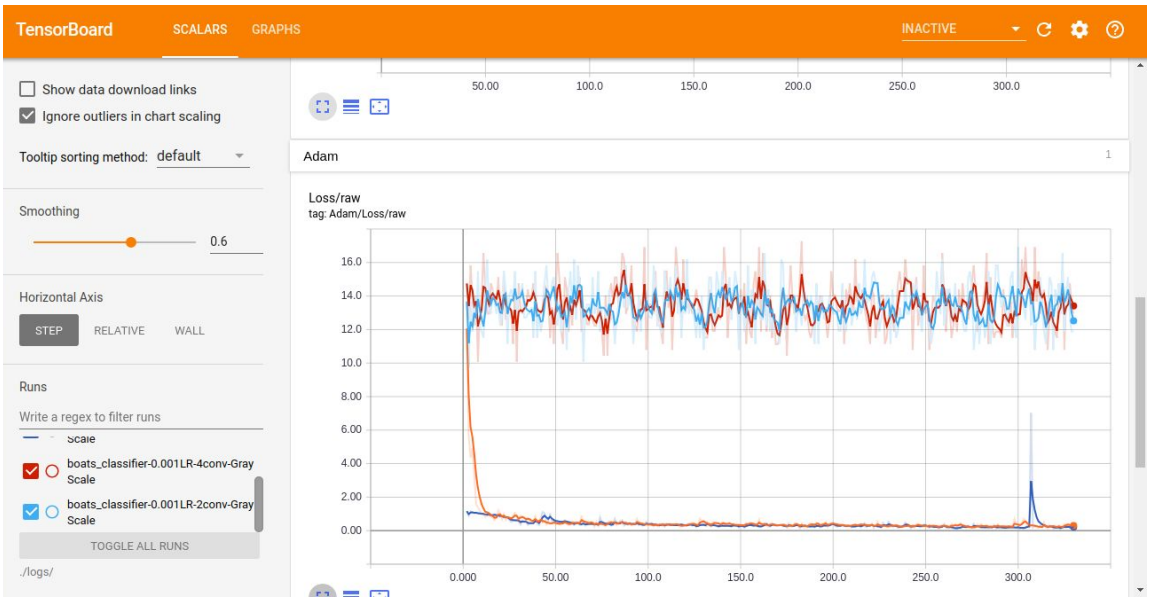
Raw Accuracy Graph



Smoothed Accuracy Graph



Adam Optimisation Graph



Loss Graph

Comments

As evident in the charts above, there is a significant difference between networks with number of convolutional layers less than 6 and networks with 6 or more

convolutional layers. It might be safe to say that a network needs at least 6 convolutional layers to be able to learn the features of images.

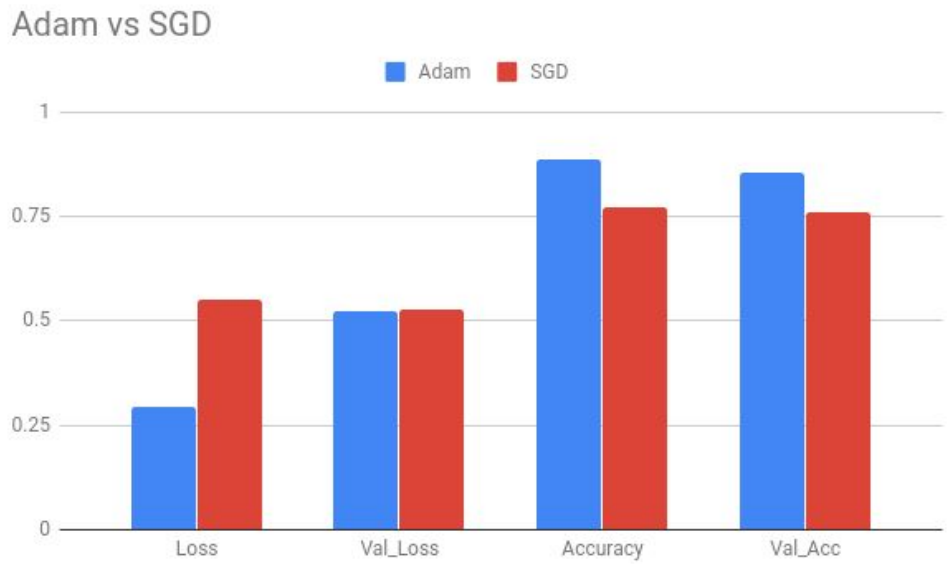
Another important thing to note is that the performance with 4 layers is not that much different for the performance with 2 layers, ditto for 6 and 8 layers. We might be able to draw an inference from this that for networks with layers less than 6, the network is unable to properly learn the features of the images and as such, the bad performance will be quite similar. We might also be able to say that networks with 6 convolutional layers are able to learn enough properties of one channel images and so adding more layers will not necessarily make serious improvements on the result; perhaps tweaking other parameters will

Furthermore, in the TensorBoard graphs shown above, we can see that the behaviour with accuracy, adam optimisation, and loss is irregular for networks with layers less than 6; the values did not increase or decrease with time (as the case may be), with convolutions of 6 and above, however, the correct expected behaviour was seen - accuracy increased while loss decreased with time.

3.2 Experiment 2: Comparing Adam and SGD Optimizers

No of Epochs	10
Learning Rate	0.001
Convolutional Layers Activation Function	ReLU
No of Convolutional Layers	6
RUN 1	
Output Layer Optimizer	Adam
Final Loss Value	0.29322
Final Accuracy Value	0.8880
Final Validation Loss Value	0.52397
Final Validation Accuracy Value	0.8536

RUN 2	
Output Layer Optimizer	SGD
Final Loss Value	0.55026
Final Accuracy Value	0.7736
Final Validation Loss Value	0.52814
Final Validation Accuracy Value	0.7599

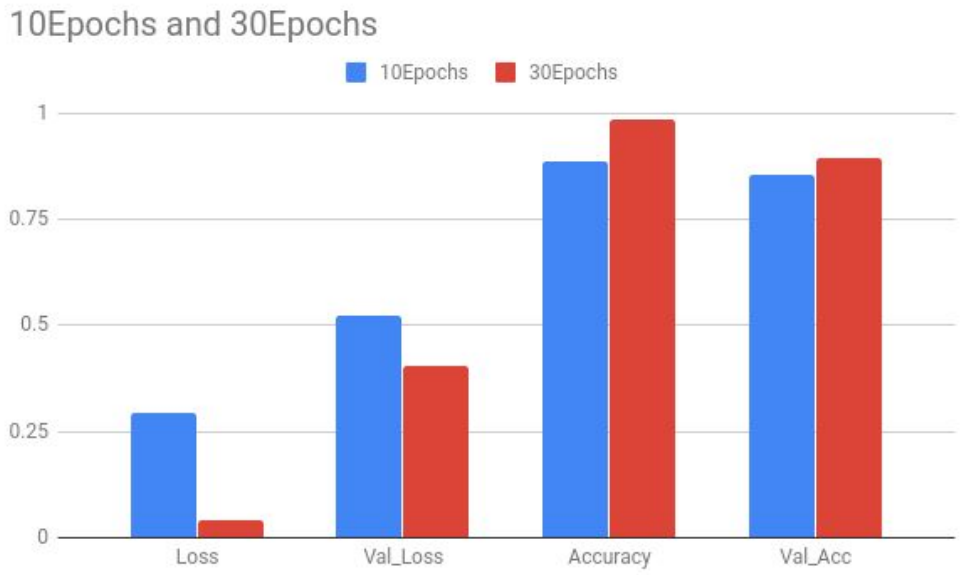


Comments

This second experiment was to compare Adam Optimisation and Stochastic Gradient Descent (SGD) optimisation (in the output layer). With all other parameters kept constant, Adam optimisation performed better than SGD optimisation on this dataset

3.3 Experiment 3: Comparing Epochs

Learning Rate	0.001
Convolutional Layers Activation Function	ReLU
Output Layer Optimizer	Adam
No of Convolutional Layers	6
RUN 1	
No of Epochs	10
Final Loss Value	0.29322
Final Accuracy Value	0.8880
Final Validation Loss Value	0.52397
Final Validation Accuracy Value	0.8536
RUN 2	
No of Epochs	30
Final Loss Value	0.04160
Final Accuracy Value	0.9856
Final Validation Loss Value	0.40386
Final Validation Accuracy Value	0.8931



Comments

Here, we see another interesting thing; the model performed much better with more epochs, we see that the loss saw a really good improvement and the accuracy got to a whooping 98%!

NOTE: Graphical confusion matrices were generated for all these experiments, but they have not been added to this report to avoid making this report verbose. All the confusion matrices are accessible in the jupyter notebook files attached.

4.0 Limitations

Learning features from images is a computationally expensive task that requires multiple iterations of matrix operations. Attempting to train on the full coloured images, with full sizes and containing all the classes on a local computer (Intel Core i5 CPU, Ubuntu OS) took more than 24 hours without having completed the training. It is hard to conduct experiments this way and a trade-off had to be made. This limitation, meant that we could not see how important colours could be, and how more varying data would affect the network's confidence.

Google Colab was tried out, but the other limitation with that was that it was really hard to get it to work with TensorBoard; TensorBoard was built to work locally.

5.0 Conclusion

This work shows how good Convolutional Neural Networks with the right parameters are in learning important features from an image, and that this power is embedded in the non-linearity in the convolutional layers. We also know that we need to have at least 6 of those non-linear layers to be able to properly learn features from images **for classification**.

It will be interesting to see how this is extended to object recognition.

References

1. https://www.researchgate.net/figure/The-baseline-FCNN-architecture-CNN-base-composed-of-3-convolutional-layers-with_fig1_316244066