University of Macedonia

Applied Computer Science

Assignment on Convolutional Neural Networks

Konstantinos Pasvantis

November 14, 2022

Contents

List of Figures

1 Introduction

In this assignment we will test the power and the usage of the convolutional neural network, a deep learning neural network that is widely used when we have to deal with image classification problems, where the typical artificial neural network is not capable of producing the results that we want.

This assignment consists of 3 different parts. In the first part we will use a simple convolutional neural network after we explain how this model is trained using our data, and the way it predicts the classes of the images in our test set. The dataset that we will work with in this assignment is the CIFAR-10, a dataset that consists of 60000 different images that has shape (32,32), grouped by 10 different classes. We first have to split our data in 3 different sets, train, validation and test set. After we split the data, we have to plot 4 random images from the train set so that we can have a look in the given dataset. These images are a subset of the set that we will use to train our model. After this, we will build the model with all the necessary parameters so that it will be in position of fitting well in the train data. The parameters of the model, and the usage of them will be commentated later on this assignment. Furthermore, we will save the trained model using a built-in function from tensorflow, so that we will not have to train again the model after we are done in the first place. In addition, after the training phase, we will keep on testing the data to observe how well the model predicts images that never saw before. To achieve this, we will obtain metric scores, such as precission, recall etc. about the predicted data, and we will be able to have an idea on how these scores were calculated, as we will see the confusion matrix of predicted classes of images and the actual class. Lastly, we will plot 4 random images for each class, this time in respect to what the convolutional neural network predicted, instead of the actual class of the image.

In the second part, we will repeat the same process, only this time we will use a different loss function in the compiling phase of our model, and, in the third part, we will repeat the first experiment but we will build the model to be more complicated, as in terms of more convolutional layers.

After the end of these experiments we will be in position to have a look at the metric scores of all trained models in the test data, so that we will be able to compare the three different models.

2 Explaining the convolutional neural network

As explained in the introduction of this assignment, the first section will be all about explaining the way the convolutional networks work with our data in order to produce the right predictions.

Artificial Neural Networks (we will refer to them as ann), fall shortly for image classification when compared to Convolutional Neural Networks (we will refer to them as cnn), for the simple reason that images could have too many pixels for the ann to work with. For example lets say we feed the ann with images of shape (1080,1250,3), meaning that an image is 1080 pixels long, 1250 pixels wide, and 3 channels of colour (RGB). The typical ann will have to work with 1080x1250x3= 405×10^4 neurons in the input layer. If we also take account the neurons in the hidden layers we know that the computation of this model will take a decent amount of time to train with a dataset with lots of images. The aim of the cnn is to reduce the dimension of images, so that the computation won't be as expensive as this from ann, and to extract the features of classes from the images.

First of all cnn is named as such, beacause the first layer is always a convolutional layer. The first part about convolutional layers we must put into account is the kernel, or the filter. The kernel plays the part of the feature extractor in every image. Kernel is a matrix with low number of dimensions, consists of weights, and hovers over every subregion of an image. For example, if the kernel is a (3,3) matrix, it moves over every subregion of the image with shape (3,3), from top-left corner to bottom right, row by row.The amount of pixels this kernel moves in every row and in every column on the image is definited by the stride value.The kernel performs the dot product between itself and every subregion it passes over, and gets a matrix as output that has values the dot product with every subregion of the image. The output matrix is referred to as a feature map of the image, using the specific kernel. The numbers of the kernels to use is specified by the user in every problem of image classification using cnn. For example, if we use 32 kernels in our convolutional layer, then the output will give us 32 different feature maps, that were calculated by the dot product between the 32 different kernels and every subset of the image they were moving over. Once this computation is completed, we pass onto the activation layer. The activation layer simply takes every feature map in our convolutional layer, and applies the Relu (Rectified Linear Unit) function, or another activation function of our choice.

Except of the convolution layers there is something called pooling layer. Pooling is used in order to lower dimensions of the feature mappings. The pooling layer applies a function in every feature map, after it is recomputed with the ReLu application, and downsamples the feature mappings. The filter it applies is specified by the user and a common pooling layer uses a (2,2) filter, with a stride of 2. This means that it applies its function in every (2,2) subset in all of the feature maps, and downsampling them by half. There are multiple functions for this to happen but the most common are these of Mean pooling and the Max pooling. If we use the mean pooling for example, with a (2,2) filter and a stride of 2, the filter will find the mean of every (2,2) submatrix in every feature map, and use this value in the output, that will be exactly the half size as the feature maps.

The cnn consists of the convolutional and pooling layers we talked about above. We pass into the cnn a convolution layer with a ReLu application, and then the pooling layer so that we can reduce the dimensions even further. Then we repeat the same procedure until we manage to downscale the size to a reasonable number. This doesnt mean we have to use a convolutional layer and right after it a pooling layer. We may use for example three convolutional layers in row, followed by one pooling layer and achive even better results than if we used the pattern of one convolutional layer followed by one pooling layer. In the case we use more than one convolutional layer in the row, each filter moves over every feature map of the previous convolutional layer.

There is a technique with which we can regularise our model called dropout. If we pass a dropout parameter in our neural network, the model has to ignore some neurons chosen randomly during the training of the model. By 'ignoring' we mean that the neural netowrk has to remove the contribution of random neurons in the feed forward part of the algorithm, and to not apply any weight changes of the specific neurons in the back propagation part.

After this downscaling of the feature maps the next thing is to flatten them . With the word flatten, we mean that we transform all of the feature maps we have into a single column. For example, if we feeded the neural network with 4 convolutional layers and 4 pooling layers and this gave us an output of 64 feature maps that each of them has a size(3,3), then if we flatten all of them we will take a column with $64 \times 3 \times 3 = 576$ values. Finally, after we flatten the feature maps, the column that is formed is passed into a classic neural network for processing. The next things are considered trivial, compiling the model with a loss function and an optimizer while taking account a metric such as accuracy of the model, fitting the model in our train set while we use the validation set for better results, and predicting the

values of our test set to see how well our model trained.

3 Building the first CNN

The first model that we build to predict the classes of our test set follows the typical flow of Convolutional Layer-Pooling layer. In this case we pass in the first convolutional layer with 32 kernels, each of them having size (3,3). The activation function that we are using in this assignment is ReLu. Also, as we see in the script file, the pooling function that we are using is the MaxPooling, meaning that instead of calculating the mean in every disjoint (2,2) submatrix of images and then projecting this value in the feature map, it is calculating the maximum value. After this sequence, we pass in another convolutional layer with same parameters, only this time we are using 64 kernels, followed by another MaxPooling layer and then a dropout parameter of 0.25. After we flatten the feature maps (creating $64 \times 6 \times 6 = 2304$ values for our column), we pass in a Neural network with a hidden layer of 128 neurons and the output layer with 10 neurons(to predict which class out of 10 possible classes the image belongs to).
This model, after the trainingm, manages to predict the class of an image that never saw before with relatevly high percentages, as we can see in the above figure. In the confusion matrix shown in the script,

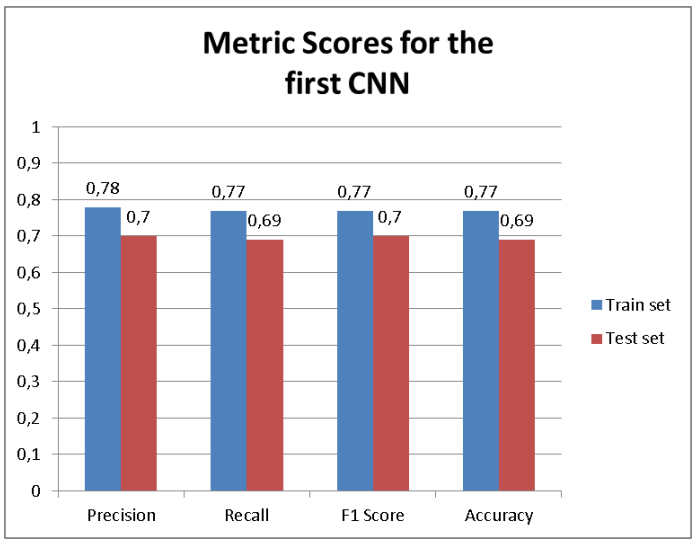

Figure 1: Metrics for our first Convolutional Neural Network

we can see that the lowest percentage the model predicted the actual class of images, is the percentage that is about the deers. In fact, the model predicted only 407 out of 1000 deers, while it predicted 889 automobiles out of 1000 in our test set.

4 Building the second CNN

The Topology for our second model is exactly the same as the first one. The only thing that we changed is the loss function when we compile the model. In the first model we used the categorical crossentropy loss function, and now we are using the Mean Squared Error function. The metric scores of this CNN is lower than these of our first model, showing us that the categorical crossentropy function worked better for this dataset.

5 Building the third CNN

In this model we are adding extra convolutional layers and the kernels we use in each layer are much more than before. More specifically, we have 3 convolutional layers at the beggining, each with 128 kernels, followed by a max pooling layer. Moreover, we add 2 more convolutional layers with 64 kernels each, and then, after we apply a maxpooling layer again, we add another convolutional layer with 128 kernels. While this model is much more complicated and much more expensive in terms of computations, we can observe that we have only a slight increase in the metric scores about the test set.

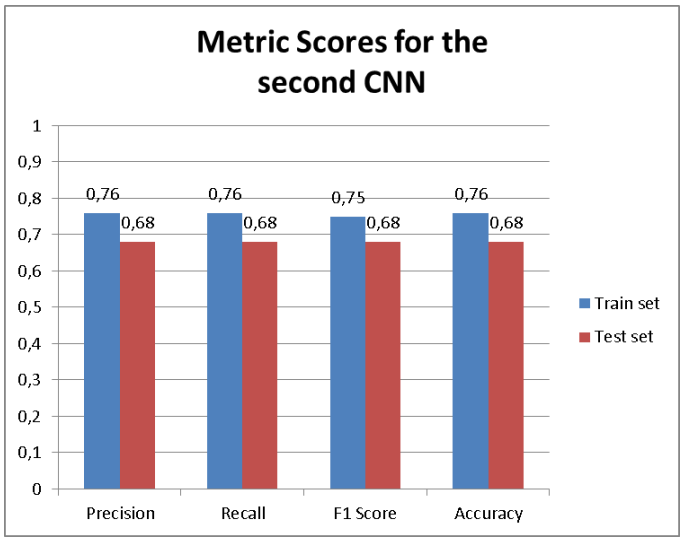

Figure 2: Metrics for our second Convolutional Neural Network

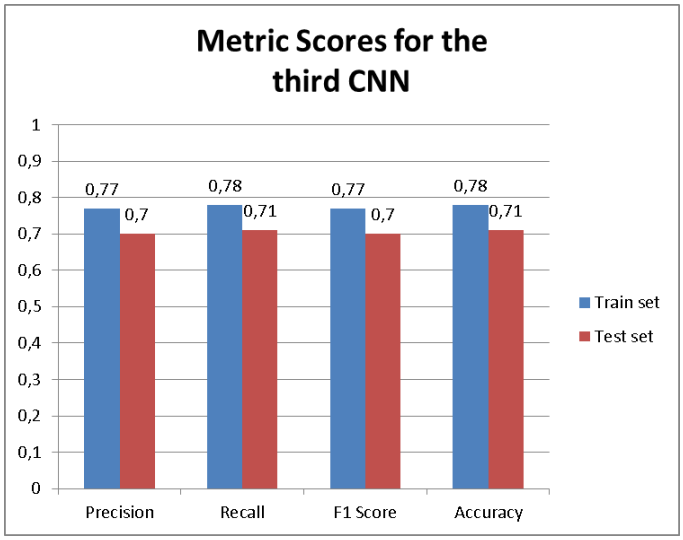

Figure 3: Metrics for our third Convolutional Neural Network

6 Summary

Convolutional Neural Networks are the type of neural networks you can definetly count on, in terms of image classification algorithms. With the use of kernels and pooling layers they manage to reduce the computational speed, in addition to give better results.

In our assignment we saw how the choice of a different loss function, or the addition of multiple convolutional layers may help or not an image classification problem. The model that worked the best with the given dataset, is the third cnn. If we want to further increase the performance of cnns, we may add more kernels in every convolutional layer, followed by MaxPooling layers. Also we can try and change the max function of the pooling layer to a mean function, or with the choice of another loss function except from Mean Squared Error and Categorical Crossentropy, the model may had better results.