



Convolutional Neural Networks for Object Recognition

EVALUATION OF THE CNN CLASIFIERS' PERFORMANCE

Abstract - This paper addresses the subject of *object recognition* and covers the basic concepts of Convolutional Neural Networks. Indeed several pre-trained CNN were taken into consideration in order to analyze both qualitatively and quantitatively their performance and accuracy. It was also implemented a code that computed the confusion matrix of the predicted classes for each CNN.

Furthermore, a particular CNN was trained upon a subset of video frames and upon different parameters in order to detect, in terms of true positives and false positives, if it quantifies well the specified targets. At last, it were applied various distortions to a character image in order to achieve the data augmentation which supplies the training stage with variations that may be present in the test dataset.

INTRODUCTION

The neural networks realization was inspired by studies in biology of nervous system. Its main purpose is to perform pattern classification in a similar way to human's brain functioning. In fact, deep neural networks with many-layer structure tries to imitate the humans' intelligence and are mainly used when it is required to create a complex hierarchy of concepts.

The two most important characteristics of neural networks are their capacity to learn and to generalize. Learning consists in programming the machine to recognize certain patterns by showing it several examples of the same class while indicating the correct answer. Whereas testing or generalization consists in using the extracted general properties of different classes of patterns to give the correct answer when new examples are shown to the trained classifier.

The impressive performance of the neural network is somewhat mysterious because the weights and biases of the network are learned automatically from the training data. Typically, this is achieved through the adjusting of weights based on the information flowing through the layers.

The neural network is a complex adaptive system which can change its internal structure based on provided input parameters. Therefore it is important to train the network on different parameters in order to estimate the best combination of inputs that offer an acceptable recognition accuracy. Thus, it might be important to worry about the learning rate, the right initialization of weights and biases, the size of the training data or the number of epochs etc.

I. QUALITATIVE ANALYSIS OF “ALEXNET” AND “VGG VERY DEEP 16 LAYERS”

ConvNet (Convolutional Network) architectures assume that the inputs are images. Thus, the layers are arranged in 3 dimensions (**width, height and depth**) and every layer performs basic image processing operations. To build a ConvNet architecture there are used four main types of layers, such as Convolutional layer, Max Pooling layer, Relu layer and Fully-Connected layer. It is also important to notice that the final output layer of the ConvNet architecture is reduced into a single vector of class scores ($1 \times 1 \times n$), arranged along the depth dimension.

The CONV layer's parameters consist of a set of learnable filters. It computes the dot product between the entries of each filter and the region it connects to in the input volume and produces a 2-dimensional activation map of that filter. Afterwards all filters' activation map are stacked along the depth dimension. Whereas, the activation map is used during test stage in order to activate distinct filters when some specific type of feature are seen at some spatial position in the input.

On the other hand, the RELU layer applies an elementwise transfer function, such as the $\max(0, x)$ which clamps negative values of convolutional layer to zero. While the Max Pooling layer provides a way to subsample the input image along the height and width dimensions. In addition, the network could have also a Normalization layer which regularizes the values and ensures that all weights inside the layer sum to one. This step allows to avoid the problem of expensive constraints implementation on conv layers. Finally, the FC layer computes the

class scores for each category. It should be clear at this point that the CONV/FC layers apply a function of both activations, weights and biases parameters to the input volume, while the RELU/NORM/MAXPOOL layers implement a fixed function to each image pixel.

Moreover it is important to mention that the size of the output volume is also controlled by other three parameters such as stride and zero-padding. The stride allocates depth columns around the height and width dimensions. Therefore, having a stride equal to one leads to overlapping receptive fields between the columns and to large output volumes, while higher strides allows to obtain a smaller output volume and a less overlapped fields. On the other hand, the zero-padding allows to control the size of the output volume. In fact, by padding the border of the input volume with zeros it is possible to balance the subsampling produced by the Max Pooling layer.

The constantly increasing computing power of GPUs has permitted to design and implement various multi-layers CNNs. However this paper will mainly discuss and analyze the AlexNet and the VGG-Very Deep (VD) Convolutional Networks, whereas the CaffeRef and the VGG-m-2048 networks will be used only to get results that could be useful to compute the AlexNet and VGG VD performance.

The Alex Krizhevsky et al. architecture, which won the ImageNet challenge in 2012, accepts input images of size $[227 \times 227 \times 3]$. It contains eight learned layers — five convolutional and three fully-connected. Each convolutional layer contains a CONV layer followed by a RELU layer, though the first and second convolutional layers have also MAXPOOL and NORM layers, while the fifth layer has only an additional MAXPOOL layer. The fully-connected layer are also made of CONV and RELU layers, except the last layer which has a SFTM layer instead of a RELU layer. In fact, the SoftMax layer allows to interpret better the obtained results by transforming the class scores vector into a probability vector. However the main peculiarities of convolutional layers design is the insertion of a 11×11 filter with a stride of 4 at the beginning followed by a 5×5 filter with a padding of 2 in the second convolutional layer, while in the first fully-connected layer the filter has a 6×6 filter. The rest of the CONV layers have a 3×3 filters with a padding of 1 and the MAXPOOL layers have a 3×3 windows with a stride of 2.

The VGG VD, instead, contains sixteen layers – thirteen convolutional and three fully-connected (see Figure 1). It has a homogeneous architecture with all convolutional layers performing a 3×3 convolutions with a padding of 1 and a 2×2 polling with a stride of 2 from the beginning to the end, while the first fully-connected layer has a 7×7 filter. Moreover its performance is very good in extracting features from images due to the depth of the network. However it was noticed that VGG Very Deep network is much slower and computationally expensive to run.

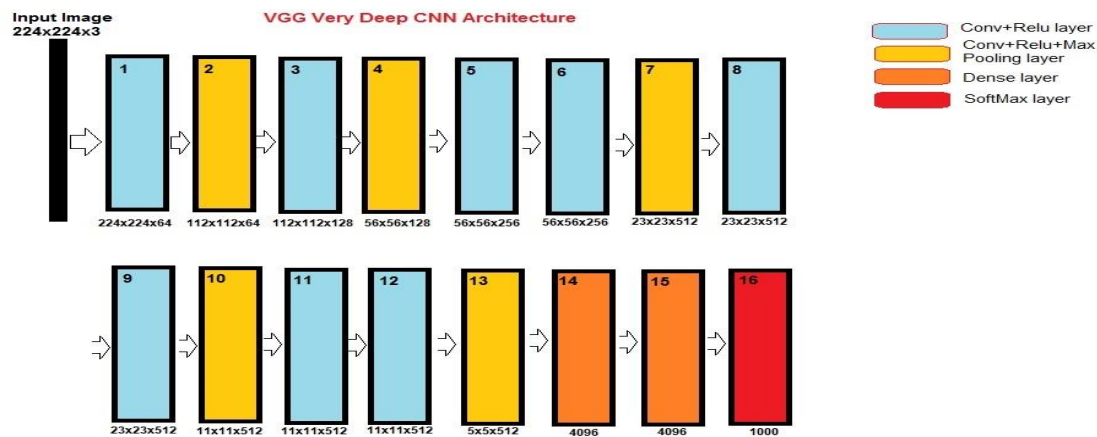


Figure1. VGG Very Deep CNN Architecture – 16 layers

As each CNN has its peculiarity, the best way to evaluate them is by comparing the test results obtained for the same categories. In fact, as it could be seen from the Figure2, different CNNs have different results. In this case,

the VGG VD 2048 offers a higher number of good results respect to AlexNet which shows the worst results in each category. However these results may be misleading because the number of categories taken into account is very low and does not allow to make a good average. Furthermore it is worth to point out that each of the networks was trained on a different number of images, for example AlexNet was trained on 1.2M images while VGG VD 16 was trained on 1.3M images.





	AlexNet	VGG VD 16	VGG VD 2048	Caffe-Ref
	<u>0.830796</u>	0.848102	0.978764	0.991458
	<u>0.420178</u>	0.550372	0.696126	0.484167
	<u>0.546260</u>	0.999942	0.932922	0.552257
	<u>0.955109</u>	0.993839	0.998364	0.988379

Figure2. A comparison of the results of four pre-trained on ImageNet CNNs.

II. QUANTITATIVE ANALYSIS OF PRE-TRAINED CNNs

Usually it is not easy to draw conclusion and make correct hypothesis when a very small subset of categories are used to train or to test the CNNs. In fact, it would be interesting to see how the overall performance of one CNN is going to change respect to another CNN when more categories and more examples for each category are taken into consideration.

Therefore, an experiment of this kind was made upon pre-trained AlexNet, VGG VD 16 and CaffeRef CNNs. Accordingly it was created a small dataset of test images by selecting 10 different object categories from ImageNet dataset and by choosing 20 examples from Google Image for each of these categories. In addition a confusion matrix was computed for each of the CNNs in order to quantitatively estimate their performance on the built dataset.

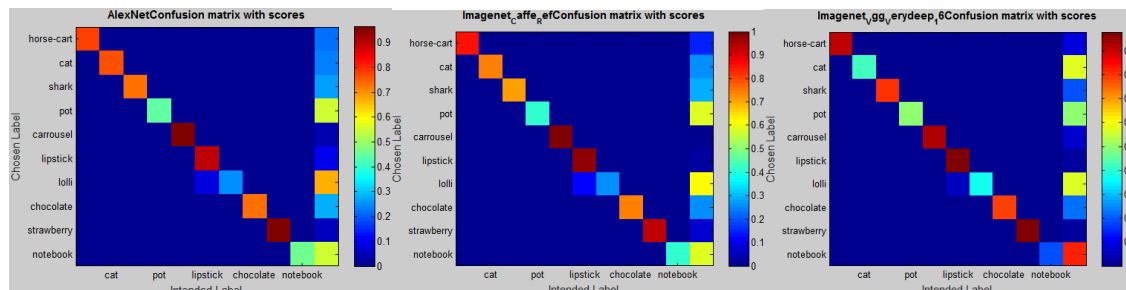


Figure3. Confusion matrix built using the scores offered by each CNN for each image. The pre-trained CNNs are AlexNet, CaffeRef and VGG Very Deep 16 (from left to right).

The confusion matrix was computed by considering the scores assigned to each image during the test session. Furthermore the values inside the matrix were normalized so that the numbers contained in each row sums up to one. In addition, it was added to each matrix an eleventh column, which holds all the other categories present in ImageNet, in order to compute the amount of non-correct categorizations.

In conclusion, by analyzing Figure3, it can be noticed that AlexNet and CaffeRef CNNs offer similar results, with only two categories (pot, notebook) being scored around 0.5 and just one category (lollipop) being scored very low. While VGG Very Deep CNN shows a very good score for six out of 10 categories, a half probability for the categories cat, pot, lollipop, and a very bad score for the notebook category. In addition, the confusion matrix tells us that all CNNs confuse the lollipop with the lipstick category, with only VGG VD16 CNN having a better discrimination of the two categories. Anyway, it seems strange that

the VGG VD16 CNN could not recognize the “tabby cat” category. In fact, VGG VD16 CNN is mostly penalized by the cat and notebook categories’ scores, but, excluding this bad results, its overall performance results better respect to the other two CNNs.

III. TRAINING A CNN FOR TARGET DETECTION

In many cases the CNN output requires a good interpretation based on the parameters chosen for the training session. Therefore, this paper discusses the performance of a very simple CNN, consisting of only a CONV layer followed by a MAXPOOL layer, on several video frames. The main purpose was to evaluate the accuracy in identifying the location of targets during the test session with every variation of training parameters, such as: the size of the filter in the conv layer, the number of epochs, the learning rate and the size of the window in the maxpool layer.

The training of the data was performed on 50 of the 500 frames and the testing was executed on the first 500 frames of the video. In order to estimate the detection rate of the targets, a “Receiver Operator Characteristic” curve and several histograms were plotted. The graphs intend to show the distribution of the true positives matches against the false positives matches.

The true positives were computed by finding the intersection between the Ground Truth mask and the detection mask (with the detected points dilated to a circle of radius 15) divided by the Ground Truth value. While the false positives were computed by intersecting the zero values inside the Ground Truth mask with the detected circles divided by “1-Ground Truth mask”. The obtained values are not accurate because the detected points during the test session should be considered as points and not as circles because otherwise the area taken into account are bigger than it really is. However, this algorithm still offers valid results because we use the same “circle” shape assumption for detections in both computations.

In figure 4 is possible to spot the variations in true positive targets detection caused by parameters modulation. In fact, it can be seen in the first graph that the augmentation of the learning rate causes an evident differentiation of the results offered by a different maxpool window size, namely a 3x3 window tends to increase both the true positive and the false positive rates, whereas 5x5 window do the reverse process. Instead, if it is taken into account a different number of training epochs, it can be noticed that the false positive rate increases significantly, while the true positive rate almost reaches the maximum with a 3x3 maxpool window and a learning rate of 15. Furthermore, it was spotted that the combination: 7x7 conv filter, 300 epochs and 15 learning rate, determines a maximum true positive rate regardless of the maxpool window size, though the false positive rate is also very high (about 0.8 – 0.9). In fact it would be reasonable to choose a learning rate between 5 and 10 and a maxpool window of 3x3, which will give a true positive rate of about 0.98 and also a quite low false positive rate.

In conclusion, by taking also into account the images included in the appendix, it can be said that generally a maxpool window size of 3x3 is preferable to a 5x5 window, except when the lines are disposed horizontally. Moreover, in most of the cases the increase in iterations number could result harmful because of false positive rate augmentation, except when a 9x9 conv filter is used, which tends to maintain the rates value while iteration number changes. In addition, the learning rate could be associated with a proportional bigger differentiation between the results offered by different maxpool window sizes. However, it can be seen that by increasing the size of the conv filter the trend becomes inversely proportional, namely the differentiation becomes smaller with the increase of the learning rate.

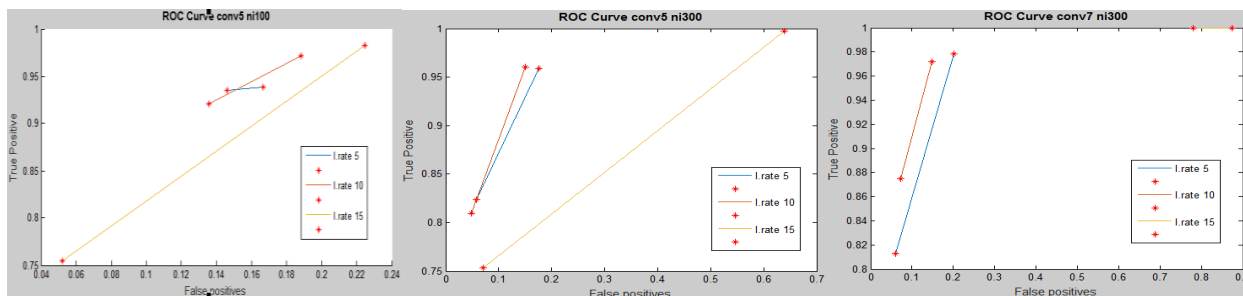


Figure4. The graphs show the 3 curves corresponding to different learning rate parameters (5, 10, 15), while the red stars indicate the maxpool window size (top point of each line corresponds to a 3x3 maxpool window, whereas the bottom point corresponds to a 5x5 maxpool window) First graph has fix 5x5 conv filter and 100 iterations. Second graph has fix 5x5 conv filter and 300 iterations. Third graph has fix 7x7 conv filter and 300 iterations.

IV. TRAINING A CNN WITH DATA AUGMENTATION

Overfitting is one of the problems that may occur during neural network training. It is mainly due to a low capacity of the network to generalize the training examples to new situations. In fact the network returns large errors each time new data are presented during test sessions. Therefore, in order to minimize this negative effect, it is good practice to randomly distort training input to the CNN. This practice is also called Data Augmentation and it was applied to a string of characters.

In figure 5 are shown the errors trend during both training and validation sessions of every change made to the training parameters. First of all it must be noticed that the training error is always lower than the validation error. Next it can be deduced that, by increasing the number of images used in each iteration, the error increases because the network finds difficult to generalize many data. However, the result is not so bad if it is taken in consideration the higher steepness of the error function and the smaller discard between the training error and the validation error. In addition it can be seen that playing with the learning rate is not a very good idea because the network offers worse results. Moreover, it is quite obvious that by increasing the number of epochs the error becomes smaller because the network has more time to extract important features which may be useful in generalizing the objects. Finally, it can be seen that, training the data without augmenting it, generates very small training errors, however the error increases significantly during the validation session, when new data are presented to the network.

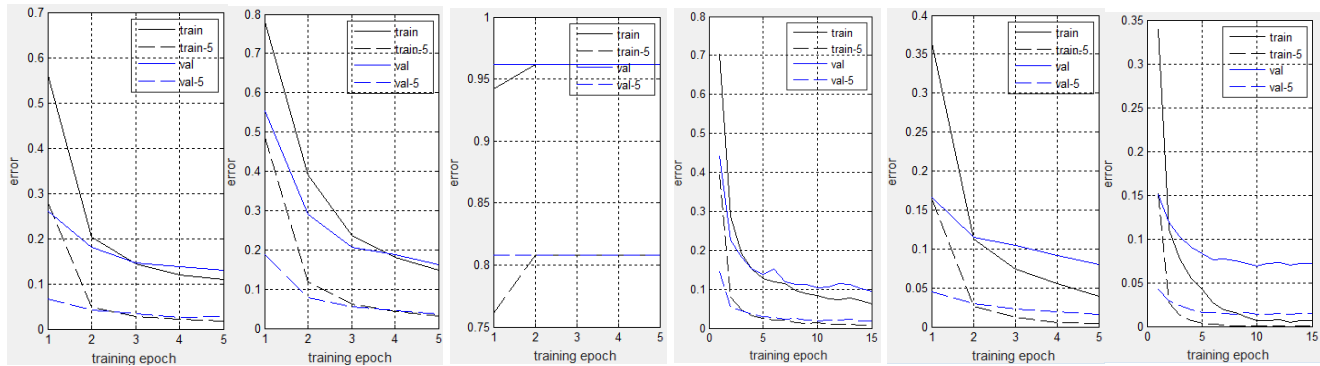


Figure5. (from left to right) Error results for character training with: 1graph – 5 epochs, 50 batchsize, 0.001 learning rate; 2graph – 5 epochs, 150 batchsize, 0.001 learning rate; 3graph – 5 epochs, 100 batchsize, 0.01 learning rate; 4graph – 15 epochs, 100 batchsize, 0.001 learning rate; 5graph – 5 epochs, 100 batchsize, 0.001 learning rate, with NO augmentation data; 6graph – 15 epochs, 100 batchsize, 0.001 learning rate, with NO augmentation data.

Accordingly to the obtained results the data augmentation is an efficient way to minimize the difference between the training error and the validation. It could be interesting to see how the results change when different affine transformations are applied to the input image. In order to make a quantitative estimation, a rotation of $\pi/4$ and a scaling up of 2 were used to plot the errors trend (see Appendix, figure4). Hereafter, it was found that the rotation transformation offers the best result, with an error equal to about 0.15, while the scaling up gives a twice as large error for the same input image. In addition, the network was trained to do the affine transformation by using a bilinear interpolation and it was noticed that it generates a high error (about 2.9), but the error function is steep, which means that the error decreases rapidly during the sessions, and the training error is almost equal to the validation error.

CONCLUSION

This paper sought to explain the design and implementation of several convolutional neural networks. The main intend was to deliver to readers the awareness of the neural networks unpredictability when modifying the training parameters without any prior evaluation of the network's architecture or previous results. Therefore a step by step

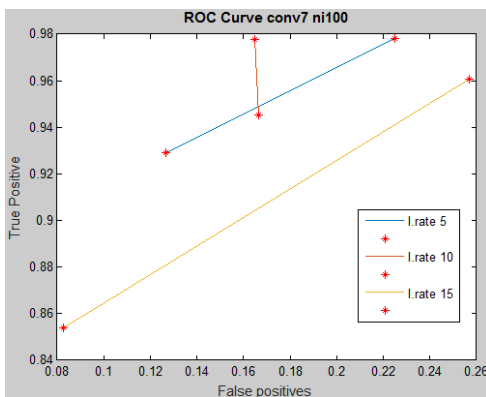
investigation was conducted on the performance of object recognition of some CNNs and each result was explained by offering intuitive and plausible answers.

REFERENCES

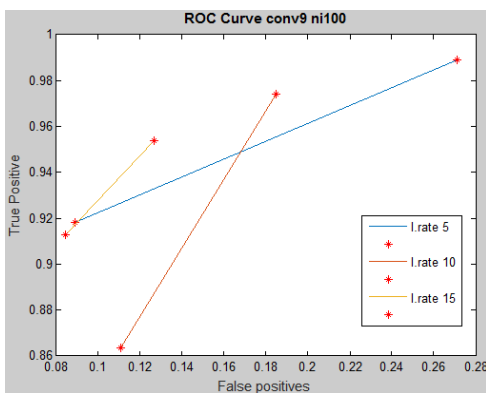
- J. Collomosse, *Image Processing and Vision*, Convolutional Neural Networks Lecture Notes, Guildford: University of Surrey, 2015.
- A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", University of Toronto, NIPS, 2012.
- K. Simonyan and A. Zisserman, "Very Deep convolutional networks for large-scale image recognition", Conference Paper at ICLR, 2015.

APPENDIX

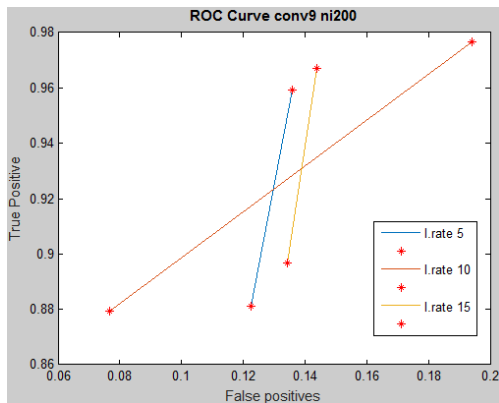
App. fig 1. The graphs show the 3 curves corresponding to different learning rate parameters (5, 10, 15). It has fix 7x7 conv filter and 100 iterations.



App. fig 2. The graphs show the 3 curves corresponding to different learning rate parameters (5, 10, 15). It has fix 9x9 conv filter and 100 iterations.



App. fig 3. The graphs show the 3 curves corresponding to different learning rate parameters (5, 10, 15). It has fix 9x9 conv filter and 200 iterations.



App. fig4. The graphs show the errors trend during both training and validation sessions while different affine transformations are applied to the input image. All graphs have the following parameters: 5 epochs, 100 batchsize, 0.001 learning rate. 1graph – applied a rotation of $\pi/4$ and a random translation; 2graph – applied a scaling up of 2 and a random translation; 3graph – applied a bilinear interpolation on a random translation.

