MO444 - Pattern Recognition and Machine Learning
Practical Assignment #4

Renato Xavier Dias
Osvaldo Xavier Dias Junior*
Prof. Anderson Rocha†

Abstract

This report is regarding the fourth assigment of the module MO444 (Pattern Recognition and Machine Learning), and it presents results of a image multiclass classification problem related to dog breeds. In this assingment was proposed to build a feature extractor and classifier to recognize 83 different dog breeds using any avalible solution. The solution chosen was to use convolution neural networks (CNN), with fine-tunning of a known CNN the Inception V2, which can perform feature extraction and classification at the same time. The results achieved were very good, presenting a normalized accuracy and F1 score above 94% for testing.

1. Introduction

The convolution neural networks are very similar to the regular neural networks and thus the former are also constructed with neurons, which have weights and biases. Each neuron receives an input, performs a dot product and, in most cases, perform a non-linear operation on top of the dot product. The non-linear operation can be chosen arbitrarily according to the type of the machine/deep learning problem, one example of this is the function is the rectifier linear unit (ReLU) as follows:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta x}} \tag{1}$$

$$z = \theta x \tag{2}$$

$$f(z) = max(0, z) \tag{3}$$

Where x is the input vectors, θ is the parameter vectors, h_θ is the predicted target vectors and $f(z)$ is the rectifier linear unit.

The CNNs have a loss function (e.g. softmax) on the last (fully-connected) layer. It follows the formula of the softmax function:

$$f(x_i) = \frac{e^{x_i}}{\sum e^{x_i}} \quad i = 0, 1, 2.., k \tag{4}$$

Where x_i is the i-th input vector and k is the number of classes. In general, this function will calculate the probabilities of each target class over all possible target classes.

The major differente between regular neural networks and convolutional neural networks is that the later assumes that the inputs are 3D (e.g. images) that allow enconding certain properties into the architecture. There are some known CNNs which represents important developments in the field of computer vision and convolutional neural networks: AlexNet, ResNet, Inception, VGG and so on.

The figure 1 shows a representation of a CNN, it can be seen a CNN with its three dimension neurons and 4 layers: input layer, two hidden layers and the last fully connected layer.

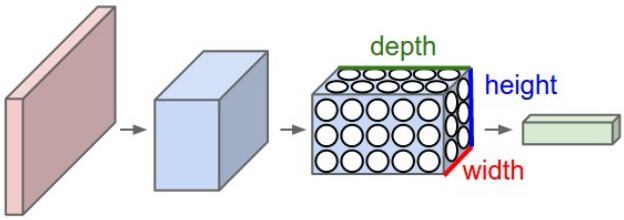

Figure 1. Representation of a CNN with 2 hidden layers

The CNN has three main types of layers to build its architectures: convolutional layer, pooling layer, and fully-connected layer. The convolutional layer is a set of learnable filters that performs a convolutional operation, slide over the image spatially computing dot products, into a local area (local receptive field) of the input image or 3D data. If it is chosen a large set of filters the data's depth increase significantly, for exemple if the input are images of size 32x32x3 (32 pixels x 32 pixels x 3 channles RGB) and

*Is with the Institute of Computing, University of Campinas (Unicamp). **Contact**: osvaldoxdjr@gmail.com
†Is with the Institute of Computing, University of Campinas (Unicamp). **Contact**: anderson.rocha@ic.unicamp.br

the convolutional layer is composed by 6 filters 5x5, then the output of the convolutional layer has a size of 28x28x6, which represents a decrease in the image size but an increse in the image depth. Another important aspect of convolutional layers is the padding, it tells how to perform the convolution operation in the boundaries of the data.
The convolutional layers, in general, adds complexity to the model, nevertheless the pooling layer makes it smaller and more manageable, operating over each activation map independently. Thus, the pooling layer helps avoiding overfitting and to reduce the use of computation resources during the CNN trainning. The pooling layer can perform operations of max pooling, average pooling and other variations, this kind of operation should be chosen carefully to each type of problem.
The fully connected layer have full connections to all activations in the previous layer, as performed in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset, this layers can represent a significant part of the computation performed in the network because all of its neurons are connected to the previous layer.
There is a technique that can be used during the CNN training, validation or testing to try to increase the size of the dataset, and this is called data augumentation, basically this technique consists of applying transformation into the original dataset. For example, when dealing with images it is possible to flip the image horintally or vertically, apply filters to the image, apply zoom to the image and other augumentation techniques. The data augumentation can be powerfull when there is a lack of data to train.
One of the main advantages of deep neural networks is that many of previous works can be reused or reimplemented to try to solve a new problem. Considering this fact, there is a technique called transfer learning which uses pre-trained models as a starting point to a new problem, hence it takes advantage of the learned weights and architecture of the pre-trained model. To maximize the efficiency of the transfer learning is recommended to choose a pre-trained model that was used to solve a problem that is similar to the new problem. Once the transfer learning is applied it is mandatory to train again the network on top of the data of the new problem, fine tuning is the process to achieve this task. There are several ways of performing fine tunning, it is possible to train part of the network, therefore freeze some layers and do not freeze others. It is possible to add new layers and classifiers at the end of the transfered CNN.
In this assigment it was performed all of the techniques described previously in order to classify 83 different dog breeds, seeking the maximum possible score. The images in the training set were captured with different camera models. There are 83 classes with 100 images for training for each class. In the validation, there is 73 images per class.

2. Proposed Solutions

The solutions proposed were based on convolution neural networks, the model executes on top of input images. It follows a list of techniques applied, in chrological sequence, to the dataset to achieve classification:

1. Choice of CNN architecture
2. Fine tunning without data augumentation
3. Fine tunning with data augumentation
4. Modification in the model parameter
5. Modification in CNN architecture
6. Testing the model

3. Experiments and Discussion

To build the algorithms the programming language used was Python using the framework Keras with TensorFlow in backend, inside an Google Colaboratory enviroment which provides a GPU Tesla K80. First of all, the proposed solution to the problem envolves finding a suitable pre-trained CNN architecture for image classification to do fine tunning. In order to help in this task it was analyzed the figure 2:

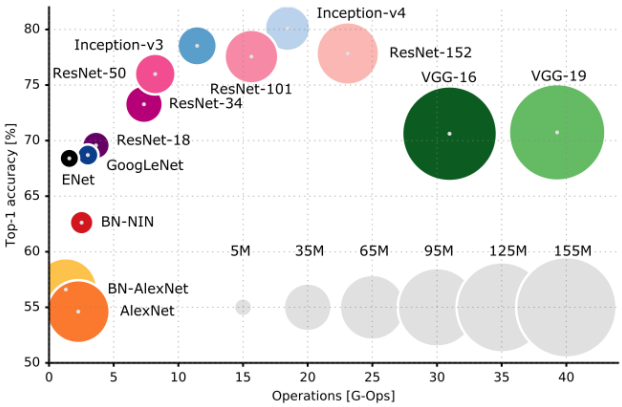

Figure 2. A comparison chart that shows top 1-accurracy versus operations to several known networks

It can be seen in figure 2 several CNNs: AlexNet, ResNet, Inception, VGG and so on. The size of the blob in for each network represents the number of parameters of the CNN. Therefore, analyzing the figure it was concluded that Inception V2, which is similar to Inception V3 is a good choice, because it has a high accuracy and not too high number of operations, which could be a feasible network to be fine tunned in the assignment.

With the selected Inception V2 network, then the CNN was downloaded (without the fully connected layers) and it

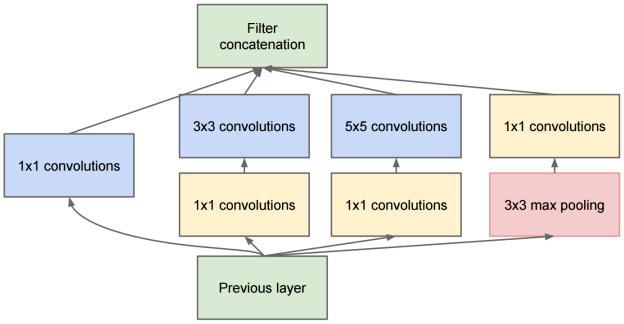

Figure 3. Inception module

was chosen arbitrarily to freeze all the layers of the CNN (do not update its layer's weights), exception made to the last four layers. This choice was made because the Inception V2 is a CNN to classify images trained for ImageNet competition (Large Scale Visual Recognition Challenge), so this network without any post fine tunning is capable of recognizing dogs, therefore it is not needed to train again most of the network. Another argument to not train again the major part of the network is that networks like Inception V2, the first layers do general recognition (edges, corners and etc.) and the last layers are more specific to the problem under analysis, so knowing that the Inception V2 recognizes dogs efficientily, probably it can recognize dog breeds without updating its first weights, being needed only to perform a fine tunning in the model.

Furthermore, it was added new fully connected layers at the end of the CNN. It was added two fully connected layers: the first was one with 1024 nodes, using ReLU as activation function, that has dropout of 0.3, which means that during training the algorithm will drop randomly 30% of the connections between this fully connected layer to the next fully connected layer. The second fully connected layers was a layer with 83 nodes and with a softmax classifier on top, the number of nodes for the last layer must be 83, because the problem under analysis has 83 diffente classes or dog breeds.

Once the architecture is defined the model can be trained. For the baseline solution it was not implemented data augumentation, but it was done a image rescaling dividing the pixels of each image channel by 255.

The objective function chosen was categorical crossentropy and a RMSprop optimizer. To evaluate the results of training and validation it was used two different metrics: normalized accuracy and F1 score. The normalize accuracy:

$$Acc_{norm} = \frac{\sum_{i=0}^{k} H_i}{N_{samples}} \tag{5}$$

Where H_i is the number of correct predictions for the i-th class, k the number of classes and $N_{samples}$ are the total number of samples of the dataset. The F1 score is calculated using the following formulas:

$$P = \frac{TP}{TP + FP} \tag{6}$$

$$R = \frac{TP}{TP + FN} \tag{7}$$

$$F1_{score} = \frac{2PR}{P + R} \tag{8}$$

Where P is precision, R is recall, TP is true positive, FP is false positive and FN is false negative. To evalute the F1 score it was calculeted the F1 score classwise and performed an weighted average according to the number of samples in each class. The results of training and validation after fine tunning are shown in table 1:

Table 1. Verification of metrics of fine tunning without data augumentation

	Train. Set	**Val. Set**
Normalized Accuracy	0.962410	0.921787
F1 Score	0.962642	0.921960

The previous table shows an excelent result, this was expected knowing that Inception V2 is a pre-trained network with several hours of training to classify images, including dogs. The training scores are better than the validation ones, this mean that the model predict better the training dataset, however there is a very good generalization, presenting validation scores above 90%.

The next approach to solve the problem is to include data augumentation on the dataset. This augumentation was applied only in the training dataset and the techniques were: horizontal flip, rotation, width shift and height shift. So, after inserting original and augumented data, the fine tunning was performed and the results were generated:

Table 2. Verification of metrics of fine tunning with data augumentation

	Train. Set	**Val. Set**
Normalized Accuracy	0.929398	0.922783
F1 Score	0.929886	0.922953

Comparing table 1 and 2 it is clear to see a little improvement in validation and a decrease in the training performance score. This is also expected because adding augumentation to the solution your model tends to become more robust, once the model is fed with more data, specifically variations of the original data. Hence, the augumentation is a technique that might decrease the variance of the model, improving the generalization and, therefore, reducing overfitting. In order to keep improving the model it was

proposed a hyperparameter modification of the CNN architecture, it was modified the dropout from 0.3 to 0.6, which means that more connections will be dropped from two last fully connected layers during training. With a new value for dropout it was performed the fine tunning and thus recalculating the weights of the network, the results of this modification can be seen in table 3:

Table 3. Modification in the model parameter: dropout = 0.6

	Train. Set	**Val. Set**
Normalized Accuracy	0.944096	0.938393
F1 Score	0.944144	0.938663

It was possible to see an improvement in the both normalized accuracy and F1 score, and hence in the solution proposed. This happens because of the characteristics of dropout that acts as a regularization to the CNN network, and thus can reduce variance or avoid overfitting. Intuitively, if it is removed randomly the connections between the two last fully connected layers during the training iteraction, it makes more difficult for the model to memorize the dataset, avoiding overfitting.
The last attempt to reach a better model was to modify the architecture itself, it was inserted a third fully connected layer just before that one with 1024 nodes. This third fully connected layer added had also 1024 nodes and tunned with a dropout equals to 0.6. The results after this architecture modification is presented in the following table:

Table 4. Modification in CNN architecture

	Train. Set	**Val. Set**
Normalized Accuracy	0.927831	0.924942
F1 Score	0.928117	0.925231

Table 4 shows a worse results compared to the table 3. This new model has higher complexity than the previous, but it degrades the results. This highlights that this specific architecture is less powerfull than the previous one, considering the same conditions (dataset, hyperparameters and etc.).
Finally, considering the model evindentiated in table 3, the test dataset was tested to see how the model would work under unknwon conditions.

Table 5. Results of testing

	Test Set
Normalized Accuracy	0.947601
F1 Score	0.948130

The evaluation on the test dataset presents a good result both in normalized accuracy and F1 score. The normalized accuracy shows that the results in general was good, desconsidering the precision and recall. On the other hand, the F1 score considers the precision and recall, which means the model is performing good per class.
The confusion matrix was not plotted due to simplicity. The size of this confusion matrix is 83x83 and then is impracticable to show it in this report with reasonable quality.

4. Conclusion and Future Work

After the execution of the experiments it was verified the power of transfer learning, which is one of the most powerfull deep learning tools, due to the fact that it is possible to take advantage of some previous trained network and apply to a new problem with similar characteristics and achieve very good results, without spending days or weeks training a model with a expensive GPU cluster resource. The Inception V2 is a so well trained network that the improments achieved during the assigment, in fine tunning, was not impressive due to the baseline solution already presented great results, it is very difficult to improve a 92% normalized accuracy and F1 result in validation.
Another interesting topic of the assingment was to analyze the intrinsic difficulties of finding a suitable CNN architecture, it is recommended try to find an architecture that already solved one problem similar of yours, rather than start from scratch.
When working with CNNs it is mandatory to be aware of the computation costs related to design decisions that you make, for example, if you try to do transfer learning with a VGG-19 CNN it is way more expensive than using AlexNet. This choice will impact directly in the time needed to train the network, and it is also related on how powerfull is the hardware available to the perform the task.

5. References

1. http://www.ic.unicamp.br/ rocha/teaching/2018s1/mo444/index.html
2. http://scikit-learn.org/
3. https://www.learnopencv.com/keras-tutorial-fine-tuning-using-pre-trained-models/
4. http://cs231n.github.io/convolutional-networks/
5. http://www.image-net.org/challenges/LSVRC/
6. Pattern Recognition and Machine Learning. Christopher M. Bishop. Springer. (2006) Acrononym: PRML