MO444 - Pattern Recognition and Machine Learning
Final Project

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

Abstract

This report is regarding final project of the module MO444 (Pattern Recognition and Machine Learning), and it presents results of a image multiclass classification openset problem. In this assingment was proposed to build a feature extractor and classifier to recognize 3 different classes: dogs, cats and unknown, using any avalible solution. The solution chosen was a 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 good, presenting a recall measure for the unkown class around 70%.

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 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 vector of the inputs, θ is the vector of the parameters, h_θ is the vector of predicted targets 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 (FC) layer.

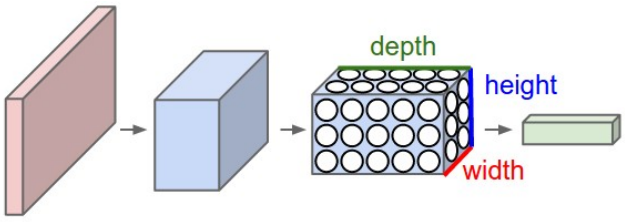

Figure 1. Representation of a CNN with 2 hidden layers

The CNN has three main types of layers to build its architecture: 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

*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

significantly, for exemple if it the input are images of size 32x32x3 (32 pixels x 32 pixels x 3 channles RGB) and the convolutional layer is composed by 6 filters 5x5, then the output of the convolutional layer is 28x28x6 which represents a decrease in the image size, but an increse in the depth. Another important aspect of convolutional layers is the padding which tells how to perform the convolutional operation in the boundaries of the data.

The convolutional layers, in general, adds complexity to the model, in the other hand the pooling layer makes them smaller and more manageable, operating over each activation map independently. Thus, this layer helps avoiding overfitting and to reduce the use of computation resources during the CNN training. 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 the starting point to a new problem, hence it takes advantage of the learned weights and architechture 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 can be added new layers and classifiers at the end of the transfered CNN.

In this final project were performed all of the techniques described previously in order to classify dogs, cats and unknown, trying to achieve the maximum possible score. The unknown class is any possible image that is not a dog or a cat. It will be explained further in this report how it was done to build this dataset, specially the unknown 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 achieve the best classification:

1. Choose CNN archtechture
2. Build the dataset
3. Build CNN model
4. Fine tunning with data augumentation
5. 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's enviroment which provedes a GPU Tesla K80. First of all, the proposed solution to the problem envolves finding suitable pre-trained CNN archtechtures for image classification to do fine tunning. In order to help in this task it was analyzed the figure 2:

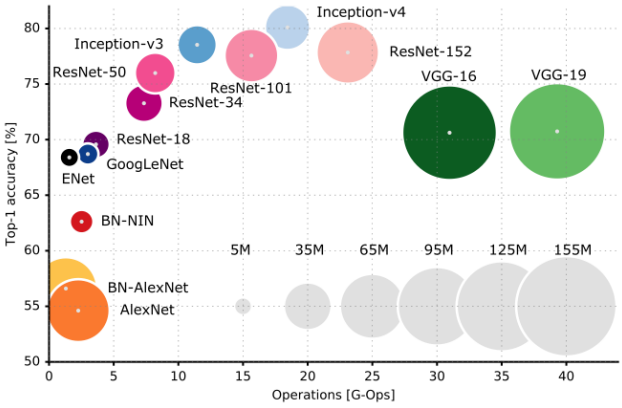

Figure 2. A comparison chart that shows top 1-accurracy versus operations to several known networks

It can be seen in figure 2 several CNNs: AlexNet, ResNet, Inception, VGG and so on. The size of the blob in the respective figure 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 final project.

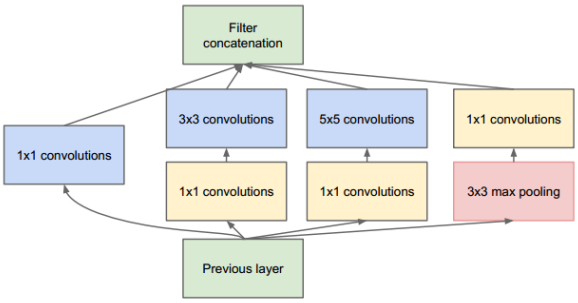

Figure 3. Inception module

At this stage, it was necessary to build the dataset in order to do this task it was downloaded two different datasets and then combined them together. The datasets downloaded was "Dogs vs. Cats" and "UKBench". The former is from a Kaggle competition and has 25000 images, within different sizes, equally distribute between dogs and cats, the images for both species contains different breeds. The later dataset is from Henrik Stewenius and David Nister and contains 10200 images of 2550 different classes with each four images at size 640x480. The images are rotated, blurred and have a tendancy for computer science motives. The UKBench dataset is special because it will be used to try to simulate an open set scenario, due to its large set of different classes. Theoretically a real openset scenario will present infinity different classes but this is not feasible in practice, so the approach here is to simulate the class unknown using some different types of image. The dataset distribution among trainning, validation and testing dataset is presented in table 1:

Table 1. Dataset distribution

	Samples		
	Dogs	**Cats**	**Unknown**
Training	5000	5000	4080
Validation	1250	1250	1020
Testing	6250	6250	5100

All three sub datasets were divided following these steps - get all the images and divide equally into two groups: training plus validation and testing. The training plus validation sub dataset is divided again equally into two: training and validation, being 80% for training and 20% for validation. Following the procedure just described, the class unknown will be trained with 1020 different types of images, validated with 255 different types of image and tested with 1275 different types of images, this is because each different class of UKBench has 4 images.
With the selected Inception V2 network, then the CNN was downloaded (without the fully connected layers) and it was chosen arbitrarily to freeze all the layers of the CNN (do not update its layer's weights). 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, cats and other images, therefore it is not needed to train again most of the network. Another argument to not train again the network is that networks like Inception V2, the first layers do general recognition (edges, borders and etc.) and the last layers are more specific to the problem under analysis, so knowing that the Inception V2 recognizes different imagens efficientily, probably it can recognized dog, cats and other images without updating its first weights and doing some adjustments in the model.
Furthermore, it was proposed four different configurations to be concatenated after the Inception V2 with its weights freezed, these can be seen in figure 4. Basically, it was added fully connected layers on top of Inception V2, gradually, to observe the convolution network behavior.

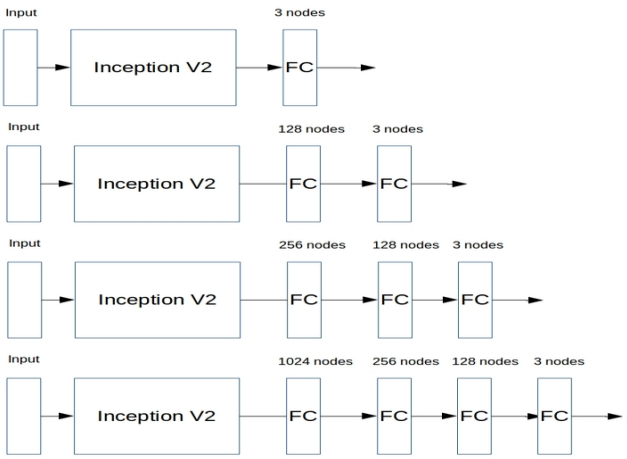

Figure 4. The four archtechture proposal

From the top to the bottom of figure 4 it was nominated each model from 1 to 4, respecting an ascending order. The training process of each model was done using data augumentation: horizontal flip, rotation of image until 20 degress at maximum, width shift and height shift. Also, for training, validation and testing it was done a image rescaling, dividing the pixels of each image's channel to 255.
The objective function chosen was categorical crossentropy and a RMSprop optimizer. To evaluate the results of training, validation and testing 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. The F1 score is calculated using the following formulas:

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

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

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

Where P is precision, R is recall, TP is true positive, FP is false positive anda 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, validation and testing of the four models is represented in table 2:

Table 2. Results of models

		Train	Val	Test
M1	**F1 Score**	0.806	0.816	0.77
	Norm. Acc	0.808	0.819	0.766
M2	**F1 Score**	0.953	0.939	0.908
	Norm. Acc	0.952	0.938	0.903
M3	**F1 Score**	0.939	0.926	0.89
	Norm. Acc	0.94	0.926	0.889
M4	**F1 Score**	0.942	0.934	0.901
	Norm. Acc	0.943	0.935	0.901

All of the four models were trained through four epochs, Model 2 presented the best values regarding the two metrics used to evaluate them. On the other hand, the Model 1 is the worst model among all of the four. These results are interesting, because the Model 1 is the simpler model, which would be the worst intuitively, however the Model 2 is the second more simpler model and provided the best result, thus increasing the complexity (adding hidden fully connected layers) did not enhanced the performence of the convolution network.

Despite the fact that the best score on training was above 90%, it is probable that the unknown class would get more wrong classifications than the dogs or cats class, to analyze such behavior it will be plotted two confusion matrix of the testing dataset: one for the best model (Model 2) and one for the worst model (Model 1), considering the normalized accuracy and the F1 Score.

The confusion matrix of the Model 1 can be seen in figure 5, it shows that the model is classifing as cats wrongly several images of dogs and unknown images, this is a biased behavior to classify cats to the detriment of other classes. For the class unknown the precision is high and the recall is low, around 47%, which indicates that the model is misclassifing 53% of the unknown class samples. Althought

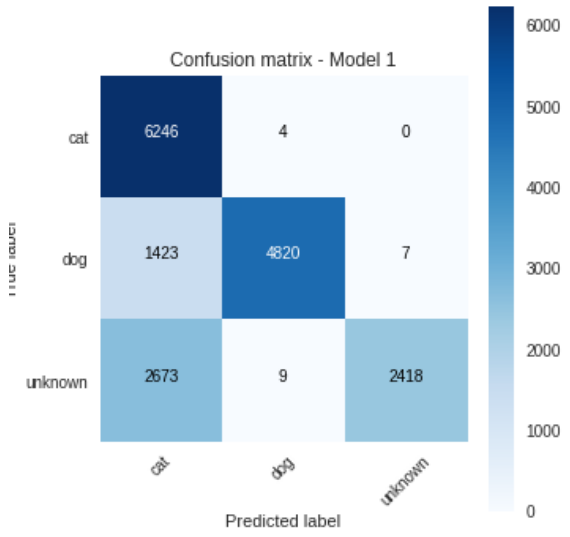

Figure 5. Confusion matrix of model 1

this result is not outstanding, it is better than chance probability, for this kind of problem chance probability is 33% considering that is the probability of select the right class given a random choice.

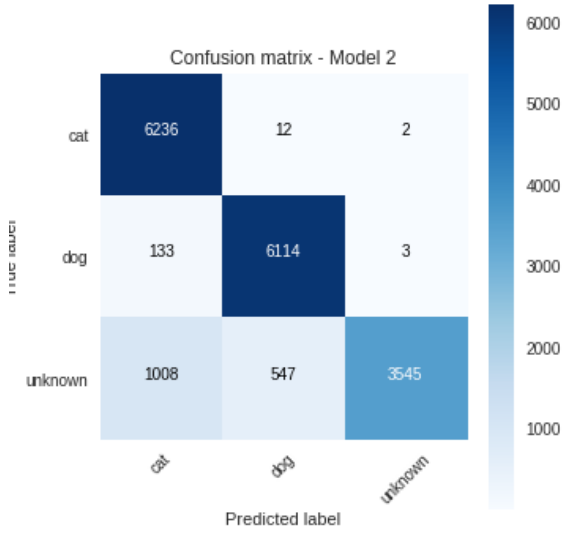

Figure 6. Confusion matrix of model 2

Once viewed the worst result, the best result presented a considerable improvement. Taking a closer look into the figure 6, which is the confusion matrix of Model 2, it cab be verified that the true positives of classes dogs and unknown ehanced, improving the recall and precision for the three classes. The recall of the unknown class went from

47% (Model 1) to 70%, so the model is getting 70% of the known class samples right. This is a huge improvement in the openset scenario, given that all samples of the testing were not seen during training, this result is two times better than chance probability. However, it must be done the consideration that for other openset samples for testing the model might not provide such good results.

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. Because 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. Another interesting topic of the final project was to analyze the emperical subject that is to find a CNN archtechture, it was shown that is not always that increasing complexity will provided better results, in fact this can took the solution to an overfiting scenario.
All problems in machine learning can be considered an open set problem, it may not be necessary to remove contantly open set samples, but all problems might present a failure in data extractor that could lead to inputs with labels that were not seen during the training process.

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