Classification with Convolutional Neural Networks

Mahdi Zerara*
*Department of Electrical and Computer Engineering
University of New Mexico

I. Abstract

In this work, a convolutional neural network was designed and trained using a large RGB image dataset. The neural network decides weather the image that we feed to the system, is of a cat or a dog. Two models of the convolutional neural networks were generated. We examine the performance of each model and then compare. We use Python programming language with some useful libraries to build, train and test the models. The CPU performed all the processing in a reasonable amount of time. The model that consists of a deeper neural network gives a better and satisfying results with almost the same computational time as the first model.

II. Introduction

Deep learning is an area of machine learning that was introduced for the first time in the 1960s. Many researchers embraced the idea of deep learning and published papers that describe algorithms with multi-layer neural networks [1]. Convolutional Neural Networks (ConvNets or CNNs) which process data and extract features in an extreme similar way humans do, came after to make a revolution in the area of deep learning. They are a category of neural networks that have proven high classification efficiency. CNNs are commonly used in image recognition tasks. In some cases, they can learn to play video games far better than any human ever could, by forming patterns from the pixels on the screen. CNNs also can be used to train a robot to cook by just watching a set of videos [2] [3]. According to Google trends comparison, CNNs gained a massive popularity increase in the past two years and are taking over artificial neural networks.
This paper presents the results of using CNNs in an experiment that was carried out on a large RGB image dataset using the TensorFlow Python library. We design two architectures and construct the convolutional neural network models that we will be using. Using the dataset that contains color images of cats and dogs, we train, test and compare the two CNN models. The comparison was established based on the prediction accuracy on both the training and test sets of each model. This paper is organized as follows. In Section III, a brief description of convolutional neural networks is given and an explanation of the different layers of the networks is presented. In Section IV, we state all the tools and methodologies that we relied on to obtain the presented results. Finally, we conclude our work by presenting and discussing the results of our experiment as well as the references of the resources that we have used.

III. Convolutional Neural Networks

In this section, we briefly explain how CNN solves classification problems. Assume that we have an *input image* and we want to build a system that classifies images according to certain features. The network will take the input image and pass it through different steps to finally recognize it as an *output label* which will in turn be used for classification purposes.

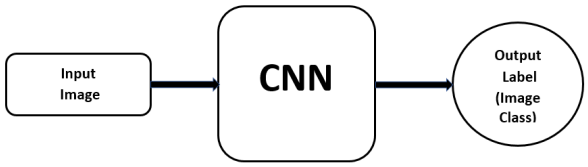

Fig. 1. A simplified graphical representation of the CNN model.

The steps that the image goes through while being processed in the network are the following:

A. Convolution

Convolution is the very first step in the CNN pipeline. Similarly to signal processing, the convolution operation is given by the the following equation:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(x)g(t-x)dx \tag{1}$$

The operation begins by extracting some features from the input image. Usually, we use 3-by-3 or 5by5 feature detectors (filters). Then we use each of these kernels and convolve them with the input image to get multiple feature maps which have a smaller size than the input image. The main role of the convolution step is to reduce the size of the input image and accelerates the processing in the network.

B. ReLU layer

In this step we apply the rectified Linear unit function on the feature maps to increase the non-linearity. Increasing the non-linearity helps the network distinguish the features more accurately [4]. The rectifier function is represented in figure-2.

C. Max pooling and flattening

Pooling is a very important next step in the CNNs pipeline. It reduces the size of the feature maps even more and preserve the most significant features that allow the network to make accurate predictions. The flattening operation is simply putting the the elements of the pooled 2-D features maps into a single large 1-D array [5].

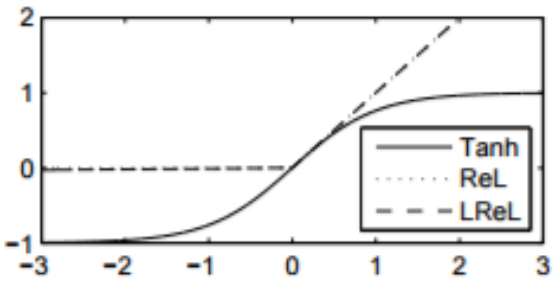

Fig. 2. Rectifier functions used in neural network hidden layers. The hyperbolic tangent (tanh) function is the typical option while some recent work has shown improved performance with rectified linear (ReL) functions.

D. Fully connected:

Finally, after all the previous steps, the reasoning part in the neural network is done via the fully connected layer(s). A fully connected layer takes all neurons in the previous layer and connects it to every single neuron it has. Then based on the highest probability of the value that we have, we make the prediction.

IV. Experiment

In order to test the performance of convolution neural network, we first had to download the data from the Internet, use the Python programming language with all the needed libraries to construct our predictor, have access to the appropriate hardware that allows us to train the model and construct the convolutional neural network. In this section, we will be describing each step of the experiment that we led to classify the images.

A. Dataset:

The dataset set was obtained from superdatascience.com. It contains 10000 64 by 64 RGB images of different dogs and cats and which were taken in various situations. We have divided the data into 8000 images for training and 2000 images for testing purposes. Both sets have an equal number of images for the two classes. In figure-3 we show some examples of the downloaded images.

B. Python libraries:

In order to perform deep learning with Python, there is 3 special libraries that we have to setup in the workspace. The first one is "Theano" which is an open source numerical library that was developed by a machine learning group at the University of Montreal. It is very efficient for fast numerical computation and runs in both CPU and GPU which will speed up the calculation process in our experiment. TensorFlow is another open source computational library that was developed by Google. the Keras library will use the computation methods from the previous libraries and construct the convolution neural networks in a very few lines of code.

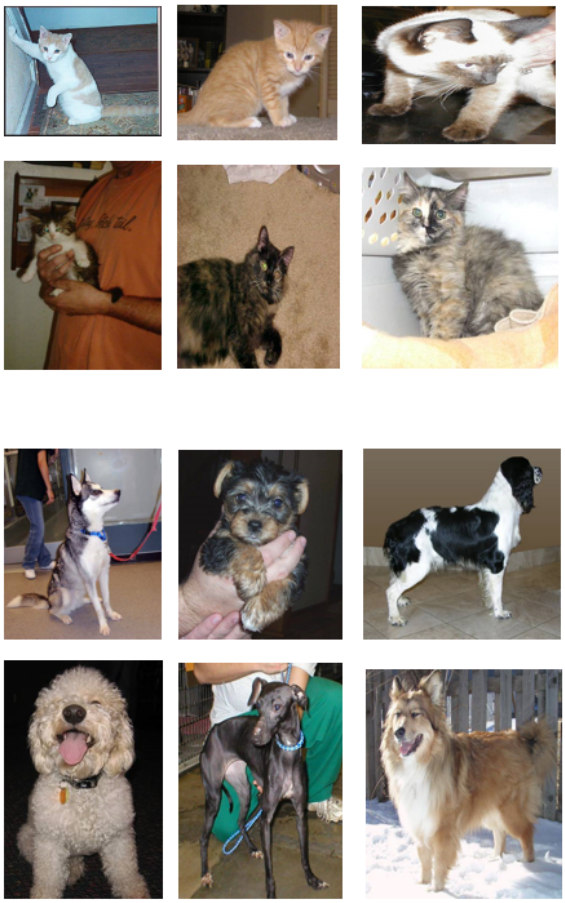

Fig. 3. Examples of condidate image from the downloaded dataset.

C. hardware:

Although a GPU is a much better choice in most recognition tasks, we have used an Intel core i7 CPU in our experiment. The reason behind this choice is the relatively small size of the dataset that we are working with and the simplicity of neural networks that designed.

D. Convolutional neural network architecture:

The architecture of our network is very simple. the image passes through a convolutional layer with 32 feature detectors of size 3x3. A max pooling layer comes next. It is performed with 2x2 pixel window, with a stride of 2. Finally we add the fully connected layer with 128 channels. We also tried another experience by using the first two layers twice to make our network deeper and compare the results that we get from the two models.

V. Results

In this section we present the results of the CNN models that we generated in our experience. The learning process has been performed in 25 epochs. Meaning that the training and testing steps are repeated 25 times in each model to improve their accuracy. For each epoch, we measure the accuracy of the models. Table-1 summarizes the results of the first CNN

model that we used:

TABLE I
PREDICTION ACCURACY (%) FOR CLASSIFYING RGB IMAGE DATASET WITH A SIMPLIFIED CNN MODEL

Epoch	*Prediction Accuracy for the training set*	*Prediction Accuracy for the test set*
1	53.1%	54.2%
2	66.23%	54.7%
10	71.54%	60.7%
15	77.13%	66.81%
20	82.41%	74.75%
25	84.52%	75.1%

Table 2 summarizes the results of the second CNN model that we used:

TABLE II
PREDICTION ACCURACY (%) FOR CLASSIFYING RGB IMAGE DATASET WITH A MORE COMPLEX CNN MODEL

Epoch	*Prediction Accuracy for the training set*	*Prediction Accuracy for the test set*
1	53.29%	55.9%
2	64.56%	68.75%
10	72.11%	71.36%
15	79.1%	75.43%
20	82.8%	80.45%
25	85.16%	81.8%

As indicated in the tables, the results that we presented include 6 epoch sequences from all the epochs that we ran the models through, along with the accuracy of the predictions. The first column shows the the epoch sequences, the second column groups the prediction accuracy on the training set that was measured during the six epochs and the third column groups the prediction accuracy on the test set. The goal was to improve the model in order to get better accuracy with the test set. Therefore, by using the two tables above, we can compare between the results.

VI. DISCUSSION

The results presented in the previous section shows that the architecture of the the convolution neural networks plays an important role in the efficiency of the model. We see that in the first model, we have reached an accuracy of 84.5% with the training set and 75% with the test set. It is clear the we had to add some improvements to the model to increase its accuracy since 75% is not satisfying to us.

After adding a convolution and max-pooling layers to the artificial network, we notice that the accuracy on the training set didn't change much. However the accuracy on the test set increased by approximately 7% without a big difference in the computational time that the CPU took to do the processing and which was 25 minutes.

The two experiences show that the architecture of the neural networks decides the efficiency of the prediction. We can add as many layers as we want to the CNN. However, adding many layers can make the computation very expensive and time consuming even with a use of a GPU. Also, a neural network with many layers can result in over-fitting. In order to avoid this two problem, we usually need to test different CNN architectures and keep the most efficient ones.

VII. CONCLUSION

in this paper we have presented the results of using the CNN model to perform recognition and classification tasks. This was done by using Python tensorFlow library. The model was trained and tested and with different real images of cats and dogs. The accuracy of the CNN was determined by the complexity of the neural networks. The deeper the network is, the more likelihood to get a better prediction. In our experience we reached an accuracy of 82% in a computational time of 25 minutes while using only the CPU. To conclude, the convolutional neural network method is very powerful for solving classification and recognition in computer vision problems.

REFERENCES

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[4] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, 2013.

[5] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," *Artificial Neural Networks–ICANN 2010*, pp. 92–101, 2010.