International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 12 | Dec 2019 www.irjet.net p-ISSN: 2395-0072

Image Classification – Cat and Dog Images

Tushar Jajodia[1], Pankaj Garg[2]

[1]Student, Department of Information Technology, Maharaja Agrasen Institute of Technology
[2]Assistant Professor, Department of Information Technology, Maharaja Agrasen Institute of Technology

---

Abstract - *Many problems in computer vision were saturating on their accuracy before a decade. However, with the rise of deep learning techniques, the accuracy of these problems drastically improved. One of the major problem was that of image classification, which is defined as predicting the class of the image. Cat and Dog image classification is one such example of where the images of cat and dog are classified. This paper aims to incorporate state-of-art technique for object detection with the goal of achieving high accuracy. A convolutional neural network is been build for the image classification task.*

***Key Words*: Image Classification, Convolutional Neural Network, Keras, Deep Learning, Callbacks.**

1. INTRODUCTION

The Dogs vs. Cats image classification has been around for a long time now. The Dogs vs. Cats competition from Kaggle is trying to solve the CAPTCHA challenge, which relies on the problem of distinguishing images of dogs and cats. It is easy for humans, but evidence suggests that cats and dogs are particularly difficult to tell apart automatically.

Many people has worked or are working on constructing machine learning classifiers to address this problem. A classifier based on color features got 56.9% accuracy on the Asirra dataset. An accuracy of 82.7% was achieved from a SVM classifier based on a combination of color and texture features.

In my project I am going to build a convolutional neural network to solve the problem and achieve higher performance and better results.

In my project instead of using the Kaggle data set comprising of total 25000 images, I would be working on subset of these images. My dataset would be comprising of total 10000 images. Keras would used for model building and all the code would be implemented on google colab.

2. Convolutional Neural Network

The name "convolutional neural network" indicates that the network employs a mathematical (convolution) operation. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that *convolve* with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. The final convolution, in turn, often involves backpropagation in order to more accurately weight the end product.

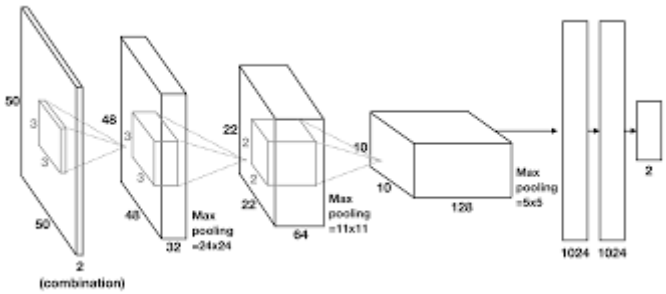

Fig 1. Basic CNN Architecture

3. Dataset and Data Augmentation

The dataset in keras is divided into folders for each class. The dataset is divided into training and testing set. The training set and the test set compose of 2 folders one for cat images and other for dog images. There are 4000 images of each cat and dog for training and 1000 image of each cat and dog for testing. The images are of varing shape and sizes, but in order to train a CNN the images should be of same size.

Data Augmentation is being carried out by using the ImageDataGenerator module provided by Keras. Using it the images are resized to 64 x 64. Also, for training of a convolutional neural network large set of images are needed. So, data augmentation is applied on the existing set of images to increase the dataset size. Various data augmentation technique such as rescaling, shear range, zoom range are being used to do the same.

4. Model Architecture

Classifier is the name given to the Sequential model. The model's first layer is a Conv2D layer. Since, it is the first layer of the model, input shape of the images that are going to be supplied to the model is being mentioned. Next layer is a

batch normalization layer. Then one activation layer corresponding to the conv2d layer. Further there is another set of conv2d, batch normalization and activation layer with different number of kernels in the conv2d layer. After that a max Pooling layer is there and then a dropout layer is there.

The same set of layers is again repeated with different number of kernel's and dropout rate. The convolution layers end with this set. Next are the fully connected layer.

The Sequential model API is used to build model. The sequential API allows you to create models layer-by-layer. The 'add()' function to add layers to our model. The model needs to know what input shape it should expect. For this reason, only the first layer in a Sequential model needs to receive information about its input shape.

Dropout layer consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting.

Activation layer are used to apply the activation function to the output of that layer. The purpose of the activation function is to introduce non-linearity into the output of a neuron. Relu and sigmoid activation are used in the model.

Batch Normalization is used for improving the speed, performance, and stability of artificial neural networks. Batch normalization is a method we can use to normalize the inputs of each layer and achieve faster convergence.

Maximum pooling, or max pooling, is a pooling operation that calculates the maximum, or largest, value in each patch of each feature map. The results are down sampled or pooled feature maps that highlight the most present feature in the patch.

In Conv2D layer, a kernel, convolution matrix, or mask is a small matrix. It is used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between a kernel and an image. This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs.

Then comes the fully connected layers. It contains only 2 layers. First one is the global average pooling layer to minimize overfitting by reducing the total number of parameters in the model. Second layer and the final layer is the Dense layer with sigmoid activation.

Global Average Pooling 2D layer is used to minimize overfitting by reducing the total number of parameters in the model. GAP layers perform a more extreme type of dimensionality reduction, where a tensor with dimensions h×w×d is reduced in size to have dimensions 1×1×d. GAP layers reduce each h×w feature map to a single number by simply taking the average of all hw values.

Dense layer implements the operation: output = activation(dot(input + kernel) + bias, activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True).

There are total 131,457 total parameters, out of which 130,881 are trainable parameters and 576 are non- trainable parameters. The major number parameters are form the conv2d layer. Batch normalization and dense layer also contribute few of the parameters.

5. Model Compilation and Training

During the model compilation, the optimizer algorithm, loss function and the list of metrics are parameters which are to be taken care of. Adam is used as the optimization algorithm, binary cross entropy is used as the loss function and accuracy is the only metric used. Early Stopping and ModelCheckPointer callbacks are used to prevent overfitting and save the best state of the model. These callbacks are mentioned as a list during the training. Sequential models fit_generator() is used to train the model. Model is trained for 100 epochs with EarlyStopping and modelCheckPointer callbacks.

6. Model Evaluation

The model trained for 15 epochs after which it stopped due to the presence of EarlyStopping callback which had the patience parameter set to 5. The training accuracy kept on increasing but the validation accuracy started to decrease which might be due to overfitting. That was the reason EarlyStopper check pointer was used to prevent results obtained due to overfitting. Below is the table showing the end result of training i.e train accuracy, test accuracy, train loss, test loss and epochs.

Table -1: Training Result Per Epoch

Epoch	Train Accuracy	Test Accuracy	Train Loss	Test Loss
1	0.771102	0.790454	0.472153	0.453161
2	0.851020	0.822045	0.340384	0.419899
3	0.878109	0.848286	0.284974	0.344910
4	0.896172	0.817980	0.246997	0.438386
5	0.907215	0.809476	0.223019	0.506150
6	0.916441	0.852508	0.202851	0.368489
7	0.923234	0.893444	0.189446	0.275266
8	0.928559	0.891424	0.175463	0.281190
9	0.932105	0.871992	0.167532	0.379654
10	0.936773	0.901068	0.158170	0.252944
11	0.939715	0.882986	0.150208	0.315709
12	0.942430	0.898951	0.144202	0.277554
13	0.944715	0.890657	0.137761	0.285958
14	0.946852	0.882450	0.132328	0.303598
15	0.949430	0.895224	0.126789	0.286201

The graph below shows training accuracy and testing accuracy vs the number of epochs.

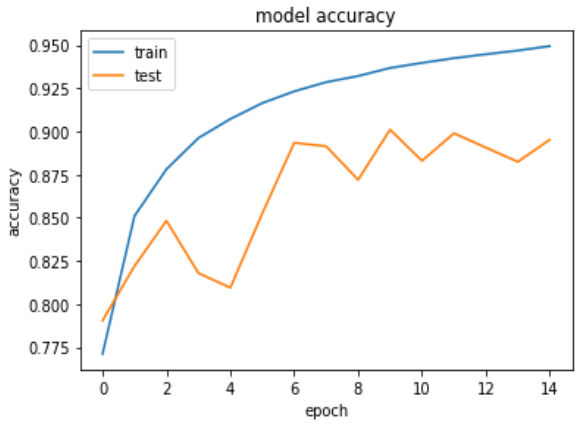

Chart -1: Accuracy vs Epoch

It is evident from the graph that the train accuracy kept on increasing with the number of epochs but the test accuracy become near constant after the 11th epoch.

The graph below shows the train loss and test loss vs the number of epochs.

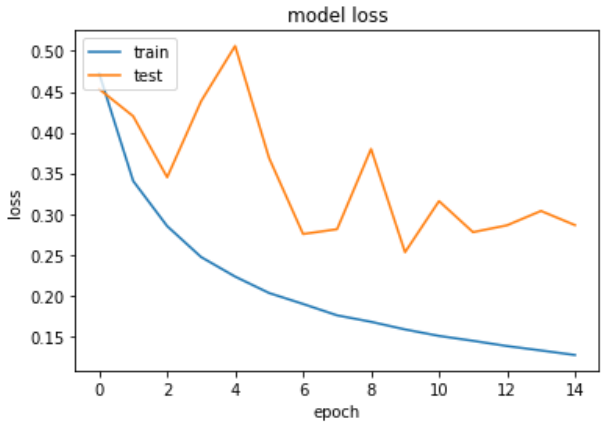

Chart -2: Loss vs Epoch

It is evident from the graph that the train loss kept decreasing with the number of epochs but the test loss become near constant after 11th epoch.

The final accuracy was obtained by initializing the model with the weights that were stored during training by the use of ModelCheckPointer callbacks. Final accuracy of 90.10% was obtained on the testing data.

7. CONCLUSION

In this paper we build a deep convolutional neural network for image classification (cat and dog images). Despite of using only a subset of the images an accuracy of 90.10% was obtained. If the whole dataset was being used the accuracy would have been even better.

REFERENCES

[1] Golle, P. (2008, October). Machine learning attacks against the Asirra CAPTCHA. In Proceedings of the 15th ACM conference on Computer and communications security (pp. 535-542). ACM.

[2] J. Elson, J. Douceur, J. Howell and J. Saul. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. Proc. of ACM CCS 2007, pp. 366-374.

[3] Muthukrishnan Ramprasath, M.Vijay Anand and Shanmugasundaram Hariharan, Image Classification using Convolutional Neural Networks. International Journal of Pure and Applied Mathematics, Volume 119 No. 17 2018, 1307-1319

[4] Parkhi, O. M., Vedaldi, A., Zisserman, A., & Jawahar, C. V. (2012, June). Cats and dogs. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on (pp. 3498-3505). IEEE.

[5] Zeiler, M. D., & Fergus, R. (2013). Visualizing and Understanding Convolutional Neural Networks. arXiv preprint arXiv:1311.2901.

[6] Bang Liu, Yan Liu, Kai Zhou “Image Classification for Dogs and Cats”.