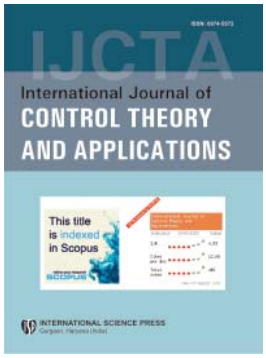

International Journal of Control Theory and Applications

ISSN : 0974–5572

Volume 9 • Number 40 • 2016

Image Classification using Transfer Learning

Kushagra Pandya[a] and Pranay Singhal[a]

[a]*Computer Science and Engineering Student, SRM University, Kattankulathur, TN 603203, India*

E-mail: kushagra.pandya.98@gmail.com, Corresponding Author

Abstract: A convolutional deep neural network was used to create a flower classifier which classifies 5 different types of flowers. The model used is built on top of Google Inception which provides an architecture for improved utilization of the computing resources inside the network. This model aims at reducing the training time while delivering the same results as that of Inception. This is achieved by reprogramming the bottleneck layer (layer just before the output) of the Inception model, and dockerizing the entire application so that it can run on any Linux machine without any issues. This model is very modular and one only needs to change labels (file names) and images inside files. The model deduces the features on its own and then returns scores for each possibility.

Keywords: *Artificial Neural Networks, Bottlenecks, Convolutional Neural Network, Google Inception*

1. INTRODUCTION

In the modern era, dependence of mankind on machines has increased exponentially with the advent of machine learning and artificial intelligence. Machine Learning is a branch of Artificial Intelligence, which enables computers to make decisions without being explicitly programmed. It focuses on development of computer programs that can teach themselves to grow and change their decisions whenever new data is encountered. Artificial Neural Network tries to replicate the neural network present inside a human brain and tries to generate similar results as the brain for a given problem. Deep learning has many advantages over other conventional machine learning algorithms like statistical training, detecting relationships between dependent and independent variables, detection of possible interactions between predictor variables, pattern recognition and many more. There are limitless applications of neural networks, to name a few – colorization of black and white images, adding sound to silent movies, automatic machine translation, emotion analysis, character text generation, image captioning and many more.

This model uses a type of artificial neural network known as the Convolutional Neural Network. In this model, the connectivity of neurons is inspired by the organization of the animal visual cortex. When compared to regular feed-forward networks with similar sized layers, CNNs have much fewer connections and parameters. The reason being, the local connectivity and shared filter architecture in convolutional layers, this makes them far less prone to over-fitting which is one of the most common problem in training models. Another noteworthy property of CNN is that the pooling operation provides a form of translation invariance and thus benefits generalization[1].

Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. Finally, the whole network expresses a single differentiable score function: from raw data on one end and scores on the other end. It still has a loss function(eg. SVM/Softmax) on the last layer. These traits make the forward function efficient and immensely reduce the amount of time required to train the network.

2. ARCHITECTURE

2.1. Convolutional Neural Network(CNN)

A Neural Network[2] receives an input (a single vector), and passes it through a series of hidden layers. Every hidden layer is made up of a set of neurons, where each neuron is connected to other neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections with others. The last layer is called the output layer and it contains the class scores.

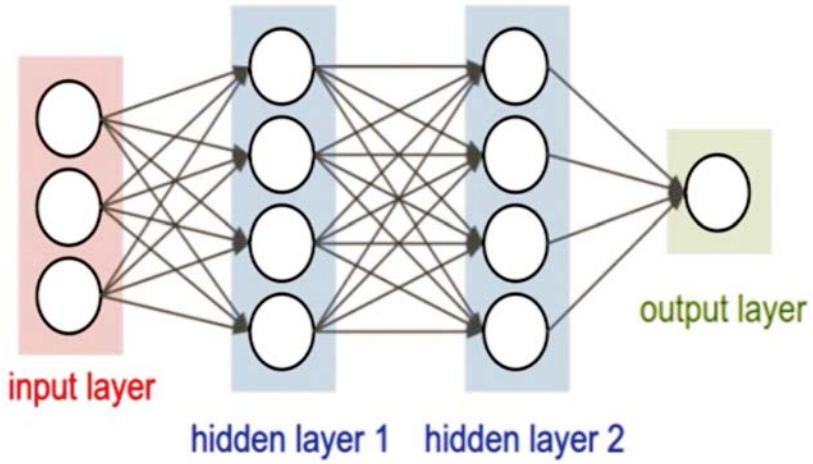

Figure 1: A regular 3-layer Neural Network

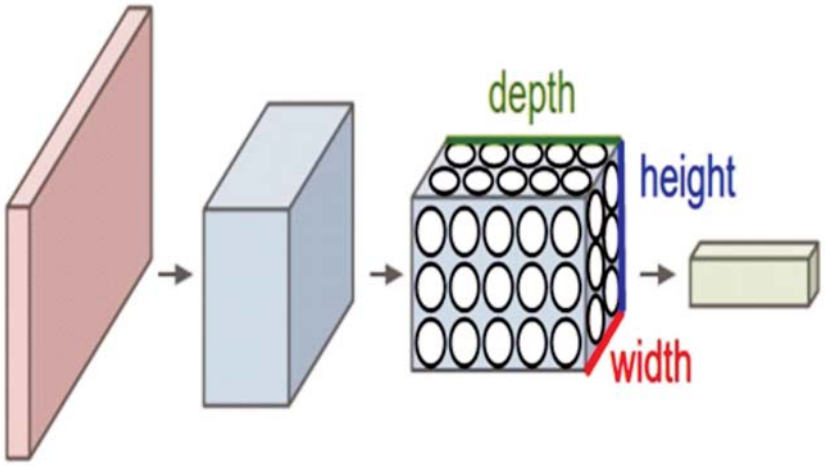

Figure 2: A ConvNet arranges its neurons in 3-D as visualized in one of the layers

Every filter h_i is reproduced covering the complete visual field in CNNs. A feature map is formed by sharing parametrization among replicated units.

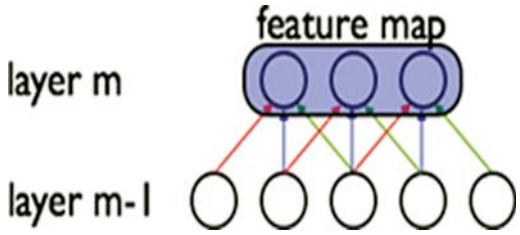

Figure 3: Hidden layer and weights

In the above figure, hidden units belong to the same feature map. Weights of the same colour are shared—constrained to be identical. For learning such parameters, gradient descent should be used, with only a small change to the original algorithm. The gradient of a shared weight is the sum of the gradients of the parameters being shared. Replicating units in this way allows the features to be detected *regardless of their position inthevisualfield.* Furthermore, weight sharing increases learning efficiency by vastly reducing the number of free parameters to learn. Thus, CNNs achieve better generalization on vision patterns using these constraints.

2.2. Regularization

With limited training data, however, many of the complicated relationships will be the result of sampling noise, so they will exist in the training set but not in real test data even if it is drawn from the same distribution. This leads to over-fitting and many methods have been developed for reducing it. One of the method is to stop the training as soon as performance on a validation set starts getting worse, and to introduce weight penalties such as L1 and L2 regularization.

L2 Regularization[3] is one of the most common form of regularization. It is implemented by penalizing the squared magnitude of all parameters directly in the objective, i.e., for every weight W in the network, the term $1/2\lambda W^2$ is added tothe objective, where λ is the regularization strength. It is common to see the factor of ½ in front because then the gradient of this term with respect to the parameter$W$ *is* simply λW instead of $2\lambda W$. Whenever peaky weight vectors are encountered, the L2 regularization imposes a penalty on it, on the other hand it prefers diffuse weights. Due to multiplicative interactions between weights and inputs, it has the appealing property of encouraging the network to use all of its inputs a little rather than some of its inputs a lot. Every weight is decayed linearly when using the L2 regularization technique:

$$W+ \;=\; \textit{-lambda*W towards zero.}$$

Every weight is penalized in L1 Regularization[4], where for each weight W a term $\lambda |W|$ is added to the minimizing function. It is also possible to combine the L1 regularization technique with that of L2 regularization:

$$\lambda 1|W| \;+\; \lambda_2 W^2$$

The L1 regularization has the property that it leads the weight vectors to become sparse during optimization i.e. very close to exactly zero. That is, neurons with L1 regularization end up using only sparse subset of their most important inputs and become nearly invariant to the "noisy" inputs. The final weight vectors that come out of L2 regularization are generally found to be diffuse and very small numbers. It is expected by L2 regularization to give a better performance than L1.

Dropout[4] is another proposed technique to fight against over-fitting. It is a regularization method that sets to zero the activations of hidden units for each training case at training time. The term "dropout" refers to dropping out units (hidden and visible) in a network temporarily. The choice of which units to drop is random. This breaks up co-adaptions of feature detectors since the dropped-out units cannot influence other retained units. Another way to interpret dropout is that it yields a very efficient form of model averaging where the number of trained models is exponential in that of units, and these models share the same parameters .

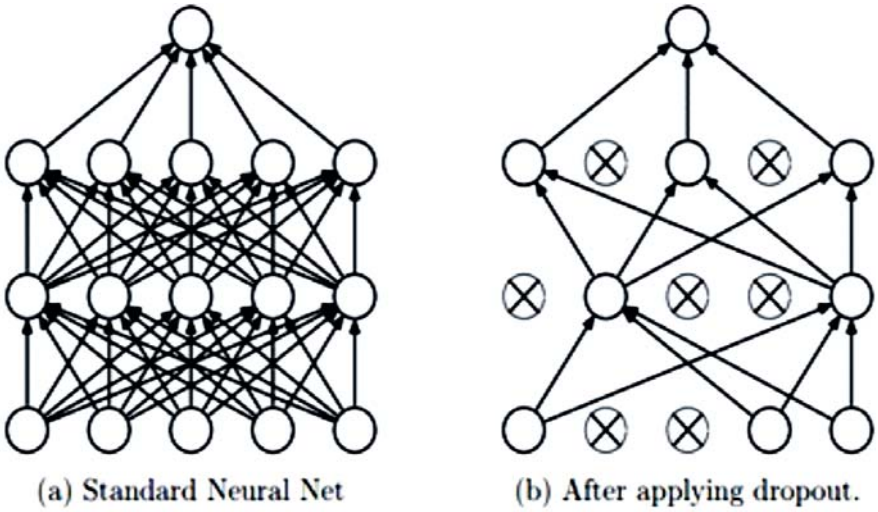

Figure 4: Standard v/s Dropout

The dropout can be imagined as a sample neural network inside a fully connected neural network (while training) which only updates their parameters based on the input data. Since they share the parameters, the exponential number of possible sampled networks are not independent of each other. During testing there is no dropout applied, with the interpretation of evaluating an averaged prediction across the exponentially-sized ensemble of all sub-networks [4].

On each presentation of a training example, if layer l is followed by a pooling layer, this forward propagation (without dropout) can be described as

$$a_j^{(l+1)} = Pool(a_1^{(l)},\ldots,a_i^{(l)},\ldots a_n^{(l)}), i \hat{I} R_j^{(l)}$$

The pooling region at j at layer 1 is $Rj^{[l]}$ and is the activity of each neuron within it. n= $|R_j^{(l)}|$ is the number of units in (l) *Rj* . *Pool ( )* denotes the pooling function.

With dropout, the forward propagation becomes :

$$a'^{(1)} \sim m^{(l)} * a^{(l)}, A_j^{(l+1)} = Pool(a'^{(l)},\ldots,a'^{(l)},\ldots,a'^{(l)}_n), i \hat{I} R_j^{(l)}.$$

2.3. Activation Function - ReLU

An activation function is used to determine whether or not a signal in a neural network will go to the next layer or not depending upon the input. It can be seen as either "ON" (1) or "OFF" (0). The non-linear activation functions are used to compute non-trivial problems using small number of nodes. This type of function is also called a transfer function.

One such activation function is ReLU i.e. Rectified Linear Unit which has been used in this mode of neural network. The function can be approximately given by:

$$F(x) = \ln (1 + e^x)$$

The use of the rectifier as a non-linearity has helped the training of deep-supervised neural networks without requiring unsupervised pre-training. The biggest advantage of using ReLU over other activation functions is that it is faster and gives a better range. It is also very functional when training on deep neural networks.

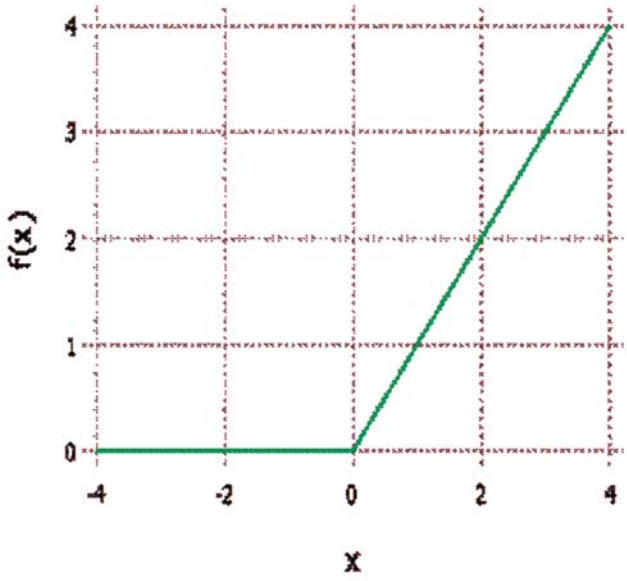

Figure 5: ReLU Graph

2.4. Google Inception

The Inception model is an architectural case study performed by Google for assessing the hypothetical output of a network constructed algorithm that tries to give out sparse structure implied for vision networks. All we need is to find the optimal local construction and to repeat it spatially for our model.

The clusters form the units of the next layer and are connected to the units of the previous given layer. We assume that each unit from an earlier layer corresponds to some region of the input image and these units are grouped into filter banks. In the lower layers (the ones close to the input), correlated units would concentrate in local regions. Thus, we would end up with a lot of clusters concentrated in a single region and they can be covered by a layer of 1×1 convolutions in the next layer.

For the residual versions of the Inception networks, cheaper Inception blocks are used than the original Inception. A filter expansion layer which is used to scale the dimensions of the filter bank occurs after each and every Inception block, before the addition is performed to match the depth of the input. It has to be done to reimburse the reduction which was induced.

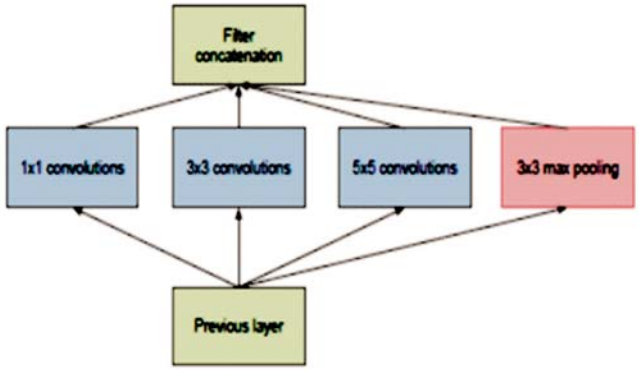

Figure 6: Google inception, naive version

A decision was made to conlove in restricted filter sizes of 1x1 3x3 5x5 to avoid patching problems in the model. It was based more on convenience than the necessity. Hence it must be noticed that all the layers of different convolutions are concatenated and these outputs are in turn fed as inputs to the next layer.

3. DOCKER

Docker is a tool which is designed to make it easier to create, deploy and run applications by using different containers.Containers package up an application with all of the parts it needs, including libraries, dependencies, etc. by doing so the developer is given an assurance that their application will work on any other Linux machine regardless of the customized settings that the machine may have.

Unlike a virtual machine, which creates a complete virtual operating system, the tool Docker allows the applications to use the same Linux kernel as the system that the user is running on and it only requires shipping the host computer running the application.

4. TRAINING METHODOLOGY

In this model we have used transfer learning. Training a deep neural network can take days to do, however transfer learning can be done in a shorter period of time.

This model doesn't train a whole neural network, rather it retrains an already trained one. Google Inception is a CNN which is trained on millions of images. A script was written to remove the old final layer, and train a new one on the flowers.

The inception v3 model is made up of many layers stacked on top of each other. These layers are pre trained and are very valuable for summarizing the information that will help classify images.

Bottleneck is an informal term given to the layer just before the output layer that actually does the classification. Every image is used multiple times during training. The bottlenecks are saved in a temporary file, so they can be reused.

While training, each step shows training accuracy, validation accuracy and cross entropy. Training accuracy shows the percentage of the images used in the current training batch that were labeled correctly with the correct class. Validation accuracy is the precision (percentage of correctly labeled images) on a randomly-selected group of images from a different set. Cross entropy is a loss function that gives an idea of how well the learning process is progressing.

The training's objective is to make the cross entropy as small as possible. The script runs 4000 times, each step chooses 10 images at random from the training set, finds their bottlenecks from the cache, and feeds them into the final layer of prediction.

In the output, you will see a list of flower labels, with the most confident prediction on the top. You can also train the network to recognize categories you want. All one needs to do is run this software, specifying a particular set of sub folders. Each sub folder is named after one of your categories and contains only images from that category.

The script than uses the folder names as the label names, and the images inside each folder should be corresponding.

5. RESULTS AND CONCLUSIONS

Rose, Dandelion and Daisy got exceptional accuracy scores, i.e. more than 99.5%. Therefore, this shows that the given model is very flexible. It can be used to classify any given set of images as per requirement taking comparatively very less time to train.

Table 1
Score of different flowers

Flower	*Score*
Rose	99.72%
Dandelion	99.96%
Sunflower	95.55%
Daisy	99.84%
Tulips	95.20%

The aforementioned results were obtained after the training of the given model of Convolutional Neural Network. Hence, it can be concluded that transfer learning is a very viable, as well as time bound method for image classification.

REFERENCES

[1] Alex Krizhevsky, I. S. (n.d.) ImageNet Classification with Deep Convolutional.

[2] Andrew L. Maas, A. Y. (n.d.). Rectifier Nonlinearities Improve Neural Network Acoustic Models. Christian Szegedy, W. L. (n.d.).

[3] Haibing Wu and Xiaodong Gu. Max-Pooling Dropout for Regularization of Convolutional Neural Network.

[4] Dropout: A Simple Way to Prevent Neural Networks from Overfitting Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever , Ruslan Salakhutdinov Department of Computer Science University of Toronto 10 Kings College Road, Rm 3302 Toronto, Ontario, M5S 3G4, Canada