Image Classification using small Convolutional Neural Network

Shyava Tripathi
Department of Computer Science Engineering, Amity School of Engineering & Technology, Amity University
Noida, Uttar Pradesh, India
shyava.tripathi8@gmail.com

Rishi Kumar
Assistant Professor, Department of Computer Science Engineering, Amity School of Engineering & Technology, Amity University, Noida, Uttar Pradesh, India
Rishikumar182000@gmail.com

Abstract— ***With an upsurge in the rate of data production, pervasive usage of cameras for automation and surveillance and the requirement of visual input for artificially intelligent devices all across the globe, there has been a rapid increase in the mass of image data being generated today. This gives rise to the essentiality of automated image processing required to simplify image related tasks. Automated image processing bridges the gap between the human visual system and the pixel level data of images. Deep Convolution Neural Networks are being deployed expansively to analyze, detect and classify images for a diverse number of tasks. These neural networks, similar to the human neural network, contain neurons with learnable weights and biases, which are trained to identify and classify different objects or features across the image. This paper presents a functional implementation of image recognition using a small convolutional neural network, proposing less complexity and yielding good classification accuracy for all tested data sets.***

Keywords-Image Classification, Object Detection, Deep Learning, Convolutional Neural Networks

I. INTRODUCTION

The utilization of software for digital image processing has soared due to the drastic rise in the volume of images database, wider availability of cost-effective image databases and the need for human-level object classification accuracy. Enormous progress has been accomplished in the field of image recognition and classification using deep convolutional neural networks and machine learning in the past few years.

The human neural system consists of a vast interconnected network of neurons that communicate and exchange inputs with each other for processing the information around us. Convolutional Neural networks (CNNs), analogous to human neural system, contain neurons with learnable weights and biases, as shown in figure 1. This system is trained with a variety of data sets to extract, analyze and classify visual patterns from image pixels. For example, Content Based Image Retrieval neural networks are capable of extracting visual features of image data such as patterns, edges, colors, shapes et-cetera and classifying these features to determine visually similar objects or images. Convolving the image layer by layer is the principal behind image recognition using this neural network. CNNs broadly incorporate convolutional layers, pooling layers, hidden layers and fully connected layers. The convolutional layer embodies a set of self-sufficient filters and every filter is autonomously convolved with the input image. We start by choosing a filter and sliding it across the whole image sequentially, in parts, and taking the dot product of the filter and each part of the image simultaneously. These filters become classifying parameters that are learned by the CNN. The output from one convolutional layer, called feature map, is passed on further to other convolutional layers for deeper convolution. The weighted sum of input values are passed to the activation function, which determine the output of a given neuron after a given set of inputs [1]. Pooling layers work on each feature map independently and reduce the overall computations and avoid overfitting by decreasing the spatial size of the representation matrix. This layer removes redundancy and smoothens computations. The fully connected layers are responsible for classifying and mapping the learned features into the sample datasets. The optimization of the network model and increased accuracy can be achieved with the help of loss functions. These functions determine the degree of variability between the predicted and observed values. Smaller loss functions represent better models [2]. The input, which is linearly transformed by the neuron's weights and biases and non-linearly transformed by the activation function is then passed to the hidden layers for further processing to obtain the output, giving rise to feed forward propagation.

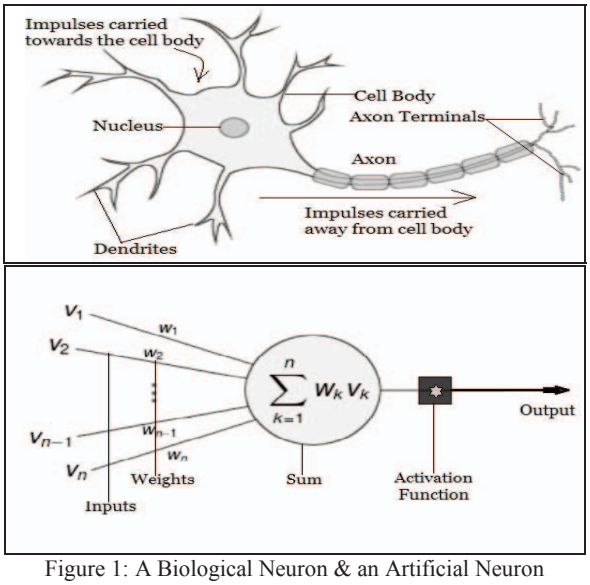

Figure 1: A Biological Neuron & an Artificial Neuron

978-1-5386-5933-5/19/$31.00 ©2019 IEEE

We have proposed a simple Convolution Neural Network for image recognition. Our model consists of 200 classes with 13 layers, namely, 5 Convolution layers, 3 Max Pooling layers, 1 Dropout layer, 1 flattening layer, 2 Fully Connected layers along with one Softmax Layer. The dataset has been divided into the three categories of training, validation and test-set to avoid overfitting. Initially, our training accuracy increases rapidly while rate of increase of validation accuracy is not so high. Meanwhile our validation loss doesn't follow the linear decrease, instead it sometimes decreases and sometimes increases.

II. THEORETICAL BACKGROUND

2.1 TensorFlow

Machine learning frameworks have made the demanding task of implementing machine learning models much simpler. These frameworks help in acquiring datasets and provide pretrained models with better refining. One such framework is Google's TensorFlow. Released in 2015, it's an open-source machine learning library that allows dataflow programming over various platforms. It can be implemented in multiple languages such as C#, C++, Java and R. TensorFlow's principle lies in deploying tensors to power the learning. This framework is capable of training deep neural networks (In our case, a Convolutional Neural Network) to perform copious tasks which can solve real world problems, such as image recognition and classification, word embeddings, speech recognition, sentiment analysis, natural language processing and so on [3].

TensorFlow enables the users to illustrate the movement of data through a progression of processing nodes. Distinct mathematical computations are represented by distinct individual nodes in the graph and nodes are connected to each other with edges, forming a network. Edges are multidimensional data arrays, also known as tensors. These mathematical computations are written in C++ and the nodes as well as the tensors are python objects. TensorFlow allows its users to design neural networks line by line, using python to conveniently couple high-level abstractions.

2.2 Machine Learning Algorithms

Supervised learning, Unsupervised learning and Reinforcement learning can be categorized as the three different classes of machine learning algorithms. Our model implements supervised learning.

In supervised learning, the dependent (target) variable is to be predicted using independent (predictor) variables with the help of mapping functions which map inputs to the desired outputs. The algorithm forges predictions based on the identified patterns and observations over the cycle of training which is continued until the desired accuracy is reached. In simple words, this category of machine learning algorithms trains the machine with the help of examples and the model is based on both input and output data [4], as presented in figure 2. Classification and Regression are the two implementations of supervised learning.

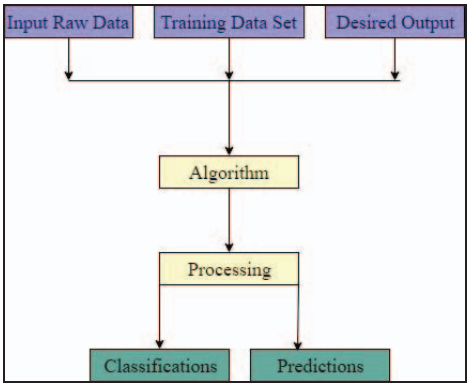

Figure 2: A Supervised Learning Algorithm Flowchart

- Linear Regression: In linear regression, we estimate the values of target variables on the basis of predictor variables and the relationship between them is established by fitting a best line, called the regression line.

> Regression line Equation: **Y= m*X + c**
> where, 'Y' : 'dependent variable'
> 'X' : 'independent variable'
> 'm' : 'slope'
> 'c' : 'intercept'

- Logistic Regression: We use logistic regression to predict discrete values based on the independent variables. These discrete values are probabilities, and thus the range of the output lies between 0 and 1.

> The logistic function is given by:
> h(x)= 1/ (1 + e^ -x)

2.3 ConvNet

We learned how to implement ConvNet, first by Convolution layer followed by Max Pooling to down sample the layer. Various activation functions were studied and the ReLU Activation function was chosen.

- **ReLU Activation function:** y=max(x,0)

If in convolution layer, input image size = 'n*n', filter size = 'f' and stride = 's'.

Input is added with 0 pad of size 'p' then [5]:

> Output: (n-f+2p)/(s+1)

- **In max pooling:**

If input is of size 'w1*h1*d1', filter= 'f*f', stride = 's, then [6]:

> Output: w2=(w1-f)/(s+1)
> h2=(h1-f)/(s+1)
> d2=d1

III. PROPOSED SYSTEM

We built a Convolution Neural Network for image recognition. Our dataset contains 200 classes (Subset of ImageNet dataset). Our model has 13 layers and consists of 5 Convolution layers out of which 3 are followed by 1 Max Pooling Layer, 1 Dropout Layer, 1 Flattening layer and 2 Fully Connected Layers along with one Softmax Layer.
The dataset has been divided into 3 categories: Training, validation and test-set to avoid overfitting. The convolutional layers are victualled with input images one after the other from all the classes, during training. After convolution, some neurons are dropped at the Dropout rate of 0.8 and the output is flattened before being finally passed to the fully connected layers. The number of outputs in the second fully connected layer match the number of classes which represent the probability of an image for each class.
The model is retained during training and is utilized to operate on our input image dataset to predict whether the given image belongs to the classes our model is trained on. Since labels are input along with the training image dataset during training, the accuracy achieved during training will be greater than the validation accuracy. However, it is important to report the training accuracy in every iteration so as to be constantly ameliorating the accuracy in the training dataset. Each iteration, or Epoch, ends with saving the model for re-iteration and reporting the accuracy. The input image is required to be read and pre-processed in a manner identical to the training, so as to obtain predictions. The saved model is restored, and the values of weights and biases learned from the previous iterations are used to predict the probability of the input image belonging to every class.

3.1 Requirement Analysis

The requirement essentials for our project are as follows:

- **Software and Hardware:**

Linux, Docker are used to implement containers to use TensorFlow without disturbing our computer environments. Python is used to implement our Neural Network and train our model. We use laptop with good CPU to run the model.

- **Functional Requirements:**

Our system needs a dataset of images to train the model and then user is required to give a random image for prediction of the class the image belongs to. The image is preprocessed according to the image provided in the training. If it is successful then it should provide the correct prediction to the user, otherwise prompt error.

- **Non-Functional Requirements:**

Our system requires high maintainability and our system should be secure. Our system should not raise error while reading images i.e. our dataset should be reliable.

- **User Requirements:**

We want our model to predict the correct label for the image with minimal error possible in appreciable time.

Figure 3 represents the UML diagram of the proposed model.

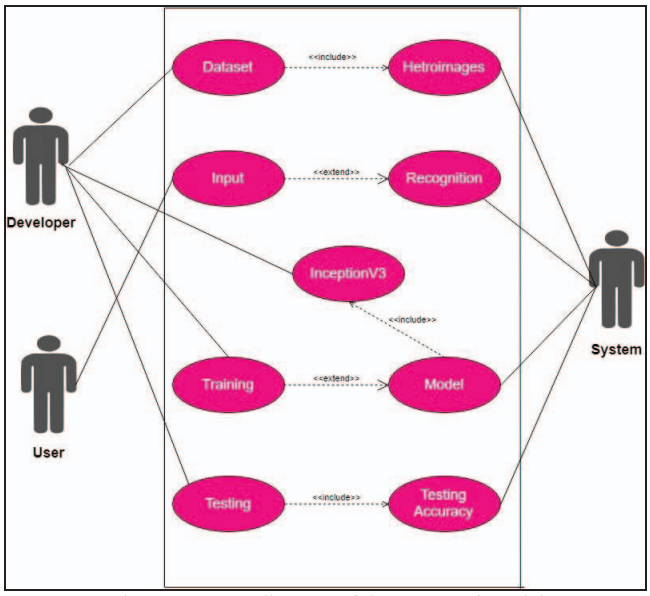

Figure 3: UML diagram of the proposed model

3.2 Architecture

We built a Convolution Neural Network for image recognition. We take only 200 classes. The model will identify and separate images of different classes.
Here Instead of initializing our parameters with zeros, we initialize them with Random Normal distribution with mean 0 and very small standard deviation of 0.05.
Our model is made up of 13 layers and has the following architecture:

- **Convolution Layer 1**: It takes images as input and applies convolution with 3*3 Filter Size and a total of 32 Filters.
- **Max Pooling Layer 1**: It takes the output of Convolution Layer 1 as input applies Max Pooling with Kernel of Size 2*2 with stride 2*2 which reduces the image height and width to half.
- **Convolution Layer 2**: It takes Output of Max Pooling Layer 1 as input and applies convolution with 3*3 Filter Size and a total of 32 Filters.
- **Convolution Layer 3**: It takes Output of Convolution Layer 2 as input and applies convolution with 3*3 Filter Size and a total of 32 Filters.
- **Max Pooling Layer 2**: It takes the output of Convolution Layer 3 as input applies Max Pooling with Kernel of Size 2*2 with stride 2*2 which reduces the image height and width to half.

- **Convolution Layer 4**: It takes the Output of Max Pooling Layer 2 as input with 3*3 Filter Size and a total of 64 Filters.

- **Convolution Layer 5:** It takes Output of Convolution Layer 4 as input and applies convolution with 3*3 Filter Size and a total of 64 Filters.

- **Max Pooling Layer 3**: It takes the output of Convolution Layer 5 as input applies Max Pooling with Kernel of Size 2*2 with stride 2*2 which reduces the image height and width to half.

- **Dropout Layer:** It takes the output of Max Pooling Layer 3 as input and randomly drops neurons with dropout rate = 0.8.

- **Flattening Layer**: This layer transfigures the multi-dimensional tensor output from the dropout layer to a one-dimensional tensor.

- **Fully Connected Layer 1**: It takes the output of the flattening layer and returns the layer after applying ReLU to add non-linearity to it.

- **Fully Connected Layer 2**: This layer receives the output from the first fully connected layer and returns the layer without applying ReLU as this contains the probability for each class.

- **Softmax Layer**: It converts the score of each class into Probability Distribution.

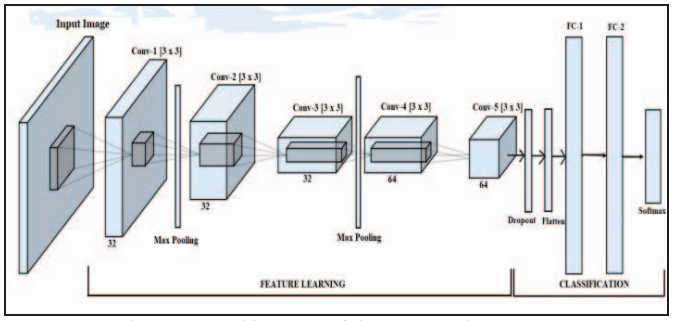

Figure 4: Architecture of the proposed system

The input images size is 64*64 pixels with 3 color channels which makes it a tensor of [64 64 3] which is fed as input into Convolution Layer-1 with 32 filters which then applies the dot product with a filter of Size 3*3 and stride 1. The image is 0 Padded so as to get the output image size same as that of input image size. The output is max pooled with window size 2*2 and stride 2 to down sample the image and finally fed into ReLU function. These are followed by 2 more Convolution layers, first with 32 filters of size 3*3 and stride 1 and again maxpooled. After applying ReLU it is sent to convolution layer having 64 filters of size 3*3 and stride 1. The convolution layers are followed by max pool and then by ReLU activation function to get an activation map. The final output of these convolution layers is fed to the dropout layer with dropout rate of 0.8 and the output is then fed into the flattening layer which converts the multi-dimensional tensor into a one-dimensional tensor with shape [1 Number_of_elements], where *Number_of_elements=Width*Height*Depth*.

The output from flattening layer is fed into the first fully connected layer with 128 neurons and applies the operation ***Y= m.X+c***. ReLU function is implemented on the output from the first fully connected layer so as to add non-linearity to the model. The output of the activation function is then treated as input to second fully connected layer which has 128 neurons as well. The output of this layer is not treated with ReLU function as the output of this layer is the probability score of each class.

We use Softmax Classifier to convert scores of each class into probability distribution and then use the Cross Entropy as our loss function. For calculating the gradient and optimizing the weights, we deploy AdamOptimizer. We are trying to achieve minimum cost with a learning rate of 0.0001 (1e-4).

The training consists of 100 Epochs with an input of mini batch of 32 images using dataset class which provides next batch of images for training. After each epoch, we calculate training accuracy, validation accuracy and the validation loss (value of cost function).

3.3 Implementation

100000 images from 200 classes (500 per class) are taken as input to train the network and are fed to the first Convolution Layer. The convolution layer itself consists of convolution followed by max pooling.

The first convolution layer consists of 32 filters of size of 3*3 and gives an activation map of 64*64*32 which is fed into max pooling with stride 2,2 decreases the height and width of the image by half. We then apply ReLU Activation function to introduce non-linearity to the model.

The output from previous layer is fed to the second convolution layer and whose output is fed into third convolution layer and the previous process is repeated. The output from these layers are fed into fourth convolution layer and fifth convolution layer where number of filters are increased to 64 giving an activation map of 8*8*64.

The output is now passed to dropout layer with dropout rate of 0.8. Dropout is applied to 'drop-out' or obviate randomly selected neurons during training. When these randomly selected neurons are 'dropped-out', weight updates are not applied to these neurons on the backward pass and they do not help in the activation of downstream neurons on the forward pass temporarily.

The output of the dropout layer is fed into the flattening layer to convert the multi ranked tensor to a 1 rank tensor which is fed into first fully connected layer with 128 neurons and ReLU function is applied to the output of the first fully connected layer which is then fed to the second fully connected layer which then passes the data into softmax classifier which then finally gives the class of the image provided to the network. We run the model with learning rate

of 1e-4 and we use AdamOptimizer to minimize the cost function and the values of weights and biases are updated through backpropagation.

IV. Results

Our model started with low training accuracy as expected due to vast number of classes but as the model trained itself, the accuracy started to increase. By 30 Epochs our model reached 50% training accuracy, a maximum of 45% validation accuracy and a maximum of 5.1 validation loss.
We trained our model for more Epochs and finally after our model reached 99% training accuracy and a minimum of 0.12 validation loss, as shown in figure 5, which is a good achievement considering the number of classes and the size of dataset per class.

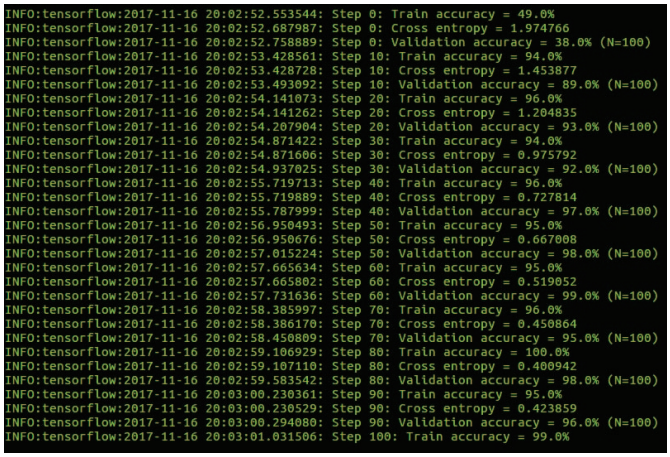

Figure 5: 99% training accuracy achieved by the proposed model

V. Conclusion

Our model is able to 99% accuracy which is good achievement. The model reaches 96% Validation accuracy. The Validation loss goes up to 0.12. Initially our model saw overfitting to a great extent which was reduced by introducing a dropout layer before the flattening Layer. The validation loss could be decreased by introducing randomization in dataset.
Even though the number of classes were high, and the number of images were less, our model achieved a good accuracy in the end with only 13 layers in the architecture. We could save the model and use it for prediction of images belonging to any of these 200 classes and get the probability score for image belonging to each class.
Our model saw the dataset for first time and was trained from scratch, so it required a greater number of Epochs to achieve a good accuracy which it did.

VI. Future Scope

We can use our model for transfer learning and train on new classes by crawling images from internet and retraining our model on those classes.
We can increase number of classes in our model to up to 1000 classes and train it on full ImageNet dataset (256*256 px14,197,122 images).
We can increase our accuracy as follows:
1. Including the Feature extraction on our model instead of giving raw input images to model.
2. We can increase our dataset from 500 images per class to at least 1000 images per class but at the cost of time.
3. Image Preprocessing - Randomize data by rotating the image, increase/decrease contrast and brightness, shifting the image from its axis.
4.We can decrease the time taken to train our model by training it on distributed GPU's and faster CPU with high memory.

VII. References

[1] Yang, J., & Li, J. (2017). Application of deep convolution neural network. *2017 14Th International Computer Conference On Wavelet Active Media Technology And Information Processing (ICCWAMTIP)*. doi: 10.1109/iccwamtip.2017.8301485

[2] Amrutkar, B., & Singh, L. (2016). Efficient Content Based Image Retrieval Using Combination Of Dominant-Color, Shape And Texture Features And K-Means Clustering. *International Journal Of Engineering And Computer Science*. doi: 10.18535/ijecs/v5i1.10

[3] Rampasek, L., & Goldenberg, A. (2016). TensorFlow: Biology's Gateway to Deep Learning?. *Cell Systems*, *2*(1), 12-14. doi: 10.1016/j.cels.2016.01.009

[4] Ma, J., Wen, Y., & Yang, L. (2018). Lagrangian supervised and semi-supervised extreme learning machine. *Applied Intelligence*. doi: 10.1007/s10489-018-1273-4

[5] Multimedia Data Classification Using CNN. (2018). *International Journal Of Recent Trends In Engineering And Research*, *4*(4), 557-561. doi: 10.23883/ijrter.2018.4274.relqz

[6] A. Hirano, A. Tsukada, K. Kasahara, T. Ikeda and K. Aoi, "Research on construction and verification of a color representation system of a cataract to assist design", *The Japanese Journal of Ergonomics*, vol. 54, no., pp. 2B1-1-2B1-1, 2018.