Scene recognition with Deep neural network

Shivam Gupta
Department of Computer Science
The University of Texas at Dallas
Richardson,Texas
SXG190040@utdallas.edu

Kavin Kuppusamy
Department of Computer Science
The University of Texas at Dallas
Richardson,Texas
KXK190026@utdallas.edu

Prachi Vats
Department of Computer Science
The University of Texas at Dallas
Richardson,Texas
PXV180021@utdallas.edu

Bhavya Sree Bombay
Department of Computer Science
The University of Texas at Dallas
Richardson,Texas
BXB180036@utdallas.edu

Abstract*—As Machine learning and Artificial Intelligence have been witnessing a monumental growth in bridging the gap between the capabilities of machines and humans, the Convolutional Neural Networks (CNN) a class of Deep neural network tends to be one of the robust algorithms in the framework of Scene and Video recognition. CNN has the capability of training huge datasets with millions of parameters and convolve it with filters to produce the desired outputs. In this article, we experiment with CNN models built with ConV layer, maxpooling layer, softmax layer and fully connected layer to evaluate its performance on scene recognition and detection datasets. The algorithm is implemented on Intel image classification datasets from Kaggle and we obtained the accuracy of 19%. We genuinely hope that our effort on this article will serve as a productive guide for the readers to fathom the concepts involved in CNN.

***Keywords*— machine learning, artificial neural networks, deep learning, convolutional neural networks, computer vision, scene recognition, Image classification.**

I. Introduction

Image classification, which plays vital role in computer vision field, is the process including image processing, feature extraction and matching recognition. As the traditional neural network algorithms succumb to faster scene recognition and feature selection, the Convolutional Neural Network tends to give the ground breaking results in the performance of pattern recognition; from image processing to voice recognition. The latest image classification techniques is being applied to scientific experiments, facial identification,self-driving cars, facial emotion detection, robotics and other fields.

Deep neural network is an implementation of neural networks with hidden layers to imitate the functions of the human cerebral cortex. The layers in the deep neural network will perform the multiple feature extraction to give multiple levels of abstractions.CNN is a good deep learning algorithm where it can deal with huge number of parameters and convolve the images with the help filters and produce the output volume.

In recent years, the optimization of CNN is mainly focused in the following components: the construction of Convolutional layer and pooling layer, the activation function, loss function, regularization term. Considering the efficacy of the Convolutional neural network, this learning can be applied the real-world practical problems.

In this report, we proposed a simple Convolutional neural network on Scene Recognition. On the basis of the CNN, we have experimented the network with different learning rate and incorporated optimization algorithm to solve the optimal parameters which has the high influence on the image classification.

II. Convolutional neural network components

Technically, in the process of training and testing the deep learning CNN models, the variable input size will be pass through the series of layers namely Convolutional layer, ReLu activation function, Maxpooling layer, Flatten layer ,fully-connected layer and SoftMax layer which will have many tuning parameters to classify the Image with the probabilistic values ranging from 0 to 1. Fig.1 shows the complete flow of Convolution neural network to process the image and classify the object based on the feature map extracted from the image.

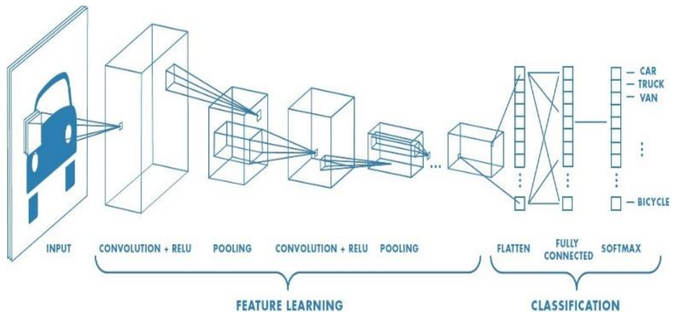


Fig. 1: Building blocks of CNN

A. *Covolutional Layer*

As the CNN involves the number of convolution, activation function and max-pooling layers and optionally followed by other layers to classify the image probability, the Convolution layer is the first layer to extract the feature maps from the input image. This layer will convolve the image to find the convolved feature map by dot product the Image matrix with the kernel or filter matrix [1]. The input image can be convolved with different filters to find edge detection, sharpening and blur etc. The kernel will take parameter "Stride" to assume the number of pixels to shift over the input image matrix while performing the dot product. The convolution layer will use the padding method if the applied filter does not fit the input image properly.

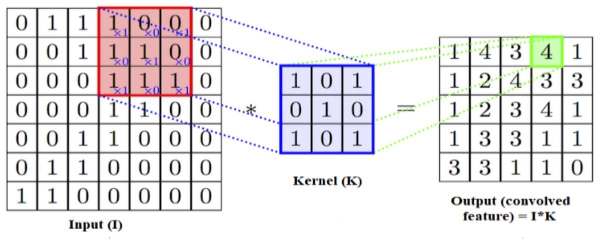


Fig. 2: Convolutional Layer

B. *ReLu Activation Function*

The Rectified Linear Unit (ReLU) is an non-saturating activation function to perform the non-linear operation to the convolved feature map. The ReLU will computed after convolutional layer to remove the negative values from the activation map without affecting the receptive fields of the convolution layer.

The output of the ReLU is $f(x) = \max(0,x)$.

C. *Max pooling layer*

In order to reduce the dimensionality of the input representation from the convolutional layer, pooling layer is incorporated to reduce the number of parameters when the image feature map is too large, without curtailing the important information from the image. Max-pooling layer is one of the down-sampling spatial pooling method which takes the maximum value from the rectified feature map. This layer will reduce the number of parameter and also the computational complexity of neural network thereby controlling the overfitting.

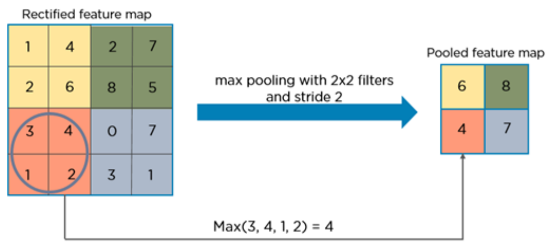


Fig4: Max pooling layer

D. *Flatten layer*

Flatten layer is present between Convolution Layer and Fully connected Layer. It basically converts a two dimensional matrix into a 1-D matrix that can be further used by fully connected layer i.e. we put all the pixel data in one line and make connections with the final layer. Hence as our pool map is ready, it flattens pool feature map into a vector. We convert pooling matrix into 1-D vector so that result can be further used by Convoluted neural network. Convoluted network stores the information in 2D matrix . But the last layer it is expected that output should be in terms of probabilities that cannot be interpreted using 2D matrix, thus we convert 2D matrix into 1-D vector. Flattening layer is in turn connected to Fully connected Layer, which further processes output of flatten layer.

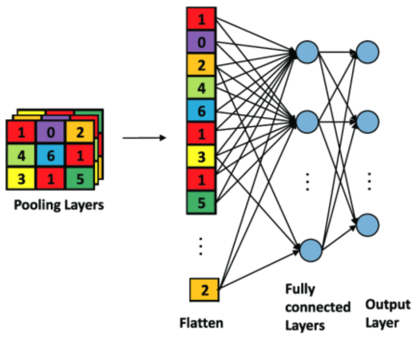


Fig 3. Transition between Flatten layer and FC layer

E. *Fully Connected layer*

Fully connected layer basically takes the output of the previous layer and then converts the output into single vector i.e. it converts a N * N matrix into N * 1 matrix where N is the number of classes. In simplified words it takes input from neurons from previous layer and operates with every single neuron to produce output.

It takes the output of previous layer and finds which feature correlates to a particular class. For example, if program predicts that a object is Tree then it would have high values in the activation map that shows high level features as trunk, leaves etc. So it considers which high level feature maps to a particular class, and has specific weights so the product of previous layer matrix and weights gives the probabilities for different classes.

F. *Softmax layer*

Softmax can be regarded as squashing function that limits the output of CNN to be a probability between 0 and 1. It is the final layer of the conv net. In scenarios where we have multi class problem it classifies the object by taking the maximum probability being assigned for all the classes. For example

Class	Probability
Mountain	0.04
River	0.001
Building	0.95
Tree	0.001
Bridge	0.008

Softmax gives us the tendency or likelihood of the object to belong to a class [3]. The sum of output is 1. It is used for the mapping of non-normalized output to probability distribution on predicted classes. The normalization can be calculated by dividing exponential value of particular output with the summation of exponential value of each possible output. This layer is the soft version of max-output layer so it can be differentiated and it is resilient to outliers. Softmax should be used when the classes are independent i.e. they are mutually exclusive.

III. Model Implementation

CNN is a deep learning algorithm, wherein weights are assigned to features and are updated during backward passes. Convoluted network requires very less preprocessing in comparison of other classification techniques. Generally conv-nets are composed of various layers including certain blocks of convolution, pooling layers, fully connected layer and an output layer.

A. *Image Dataset Preprocessing*

We have used the Intel Image classification datasets from Kaggle which contains 6 classes: building(label:0), forest(label:1), glacier(label:2), mountain(label:3), sea(label:4), street(label:5). We took 1200 Images (taking 200 images for each class). The testing datasets consist of 1500 images (taking 200 images for each class). The Images were of different sizes so we resized all the images to 100 X 100 X 3(as the image has 3 bands:- Red(R), Green(G), Blue(B)) as we have to maintain the shape of each image while giving input to the model. The pixel intensities were divided by 255 (maximum RGB intensity) in order to normalize the image training data (Normalization). The mean values were also subtracted from all the pixel values in order to standardize the image data (Standardization).

B. *Model Training Phase*

The images (150X150X3) are pre-processed before giving input to the model. They are normalized and subtracted by the mean values of the pixel intensities. The architecture used is the improvised LeNet in which we stacked together the convolution layer (5X5 Kernel), ReLU Activation, Maxpooling layer (pooling-2) twice in order to achieve better features.

Another convolution layer (with 72 filters of size 5X5) was stacked to increase the number of feature maps/depths. These feature maps consist of all kind of features like edges, texture, corners, and others which we cannot even imagine of what a CNN can extract. More the feature maps are there the better chances of extracting the information from the images. This flattening is done as we are doing the image classification as we need a single dimensional array in order to classify the whole image as we are doing scene classification (recognition). Then we used Fully connected layers which do the matrix multiplication unlike convolution so that we can get required number of feature maps which should be equal to the number of classes(6).

At the end we have used softmax which calculated the probabilities using the exponential functions. We can decide for the classification based on the probabilities out of the 6 classes (depths) that whichever will be the maximum probability we will assign the image to that particular class.

All the layers are stacked together, and the forward propagation occurs. The cross-entropy loss is calculated to evaluate the model. The backpropagation occurs which updates all the weights of all the layers. We have used the batch of 30 images together which will be given to the model, the forward and the backward propagation will occur, then next 30 images will be given as the input to the model. This was continued for all the training images. The weights were dumped after every epochs into a pickle file.

First Layer: Convolution Layer

In the convolution layer the filter slides across the image from left to right giving a modified image output. The given input image i.e. of size (104 x 104 x 3) is fed to to the first convolution layer in order to extract the features from the image. The number of input channels taken for our model in the convolution layer are 3 i.e. the R, G, B channels. The number of filters is decided based on the depth of output image. In our model the depth of the output image is 6 and hence 6 filters are being used in it. In order to increase the depth of the output image a greater number of filters can be used. This implies that as we increase the number of filters, the number of output channels also increases.

In a general scenario an odd sized kernel is used as parameter in the convolution layer. We have used a kernel of size 5 in our model implementation. Padding is used to maintain the dimension of the output image without the image itself being distorted. Hence, we have used padding of size 2 to ensure that the main features and properties of the image remain intact. The size of the step as the filter scans across the image is known as stride. In general, the size of the stride is always maintained as 1 to ensure the image remains the same and is intact. If the stride size is more than one, then this may result in a decreased size of the image. Learning rate deals with the extent of varying of weights throughout the model. We have set the learning rate to 0.01 for our model. The output of the first convolution layer will a output image of size 100 x 100 x 6.

The spatial size of the output volume can be computed as a function of the input volume size W, the kernel field size K, the stride S, and the amount of zero padding P. The output size of the conv layer is [5],

$$\frac{W-K+2P}{S}+1$$

Instead of mean squatted error, we are using Cross-entropy loss to measure the performance of a training model whose output is a probability value between 0 and 1.

$$H(p,q) = -\sum_{x} p(x)\,log\big(q(x)\big)$$

where Entropy H is calculated by true value p(x) and the estimated value q(x).

Back-propagation is the feeding the entropy loss backwards to fine tune the weight based on which the optimization function, we are gradient descent, will find the new weights that will reduce the cross entropy loss for the next iteration.

The updated weight

$$\Delta w_{ij} = \eta \times \frac{\partial H}{\partial w_{ij}}$$

where derivative of Entropy H with respect to w_{ij} is the weight gradient and η is the learning rate.

Second Layer: ReLU Activation Function

ReLu can be summarized as a non-linear function which given an input x, gives an output only if x is greater than 0 else gives

0. The activation function ReLu works well for general approximations and also converges faster in training models. The output of the first convolution layer i.e. of size 100 x 100 x 6 is fed to the activation function ReLu which gives an output of 100 x 100 x 6.

$$g(x) = max(0, x)$$

Third Layer: Max-pooling layer

In this model of object detection, an input image is considered of size 100 * 100 * 6. Now to down sample the size of the image the given image is multiplied with pooling matrix. The size of the pooling matrix is 2 and stride is 2. So, the input image is multiplied with filter of size 2, to produce output of size 50 * 50 * 6. It can be observed that depth of the image is preserved. Here Stride is 2 so filters are moved by 2 pixels at a time. The down sampling is done using MAX operation. In the above case we have filter size as 2 *2 and stride is 2 thus discarding 75% activations. Max operation works by 2*2 space in 6 depth slices i.e. it takes 4 numbers.

Max-pooling is represented as

$$y=max (x1, x2, x3…., xn)$$

where y is the output and xi is the value of the neuron.

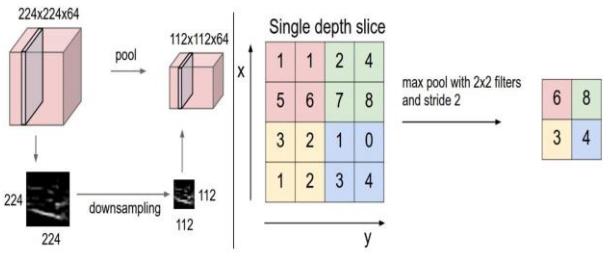


Fig4: Max pooling layer

Forward propagation produces a single value for a 2 *2 pooling block. Backpropagation of pooling layer then finds error which is present in this single unit. In order to find gradient routing during the backward pass index of this unit is stored during the forward passed, so that it can be used backward pass.

Fourth Layer: Convolution Layer

For the second convolution layer an input of size 54 X 54 X 16 is given which is derived from the max pooling layer. The parameters for the second convolution layer are set as follows. The input channel size is taken as 6 and number of filters in this layer are increased to 16 in order to increase the depth of the image. The size of the kernel remains the same as that of the first convolution layer i.e. of size 5. Padding of size 2 is used to maintain the dimensions of the output or modified image and the size of the stride is still maintained at 1 to make sure the image is intact. A learning rate of 0.01 is used in this layer in order to control the changes in the model. The output of the second convolution layer is of size 50 x 50 x 6.

Fifth Layer: ReLU Activation Function

The output of the second convolution layer is fed to the activation function ReLu that approximates the given input to an output of size 50 x 50 x 6.

Sixth Layer: Max-pooling layer

In this layer, the size of the input is 50 X 50 X 16, and this is down sampled to generate an output of 25 X 25 X 16 by using

the stride of 2 and pool size of 2.

Seventh Layer: Convolution Layer

In the third convolution layer an input of size 29 X 29 X 16 is given which is derived from the max pooling layer. The parameters for the third convolution layer are set as follows. The input channel size is taken as 16 and number of filters in this layer are increased to 72 from 16 in order to increase the depth of the image. The size of the kernel remains the same as that of the first and the second convolution layer i.e. of size 5. Padding of size 2 is used to maintain the dimensions of the output or modified image and the size of the stride is still maintained at 1 to make sure the image is intact. A learning rate of 0.01 is used in this layer in order to control the changes in the model. The output of the third convolution layer is of size 25 x 25 x 72.

Eighth Layer: ReLU Activation Function

The output of the third convolution layer is fed to the activation function ReLu that approximates the given input to an output of size 25 x 25 72.

Ninth Layer: Flatten layer

Flatten layer is used to flatten the output, i.e. it is the arrangement of the 3D volume of numbers into 1D vector. Thus 25* 25 *72 is flattened to 1 x 45000.

Tenth Layer: Fully Connected Layer

Fully connected layer (FC) takes the inputs from the feature analysis and applies weights to predict the correct label. Fully connected layer acts like normal matrix multiplication connecting every neuron with each other so as to identify each pixel in the image independently unlike convolution layer. Fully connected layers are responsible to change the number of features into the required number which could be number of classes at the end of CNN, then we can take the maximum value in order to predict the class label. In our model the FC takes the input of shape 1x 45000 and outputs the shape of 1x36.

Eleventh Layer: ReLU Activation Function

The output of the fully connected layer is fed as an input of size 1 x 36 to the activation function ReLu that approximates the given input to an output of size 1 x 36.

Twelfth Layer: Fully Connected Layer

The last fully connected layer is responsible to break the number of features maps (depths) to the number of classes. This layer takes the input of shape 1x36 and outputs the shape of 1x6. Now we got a single value with depth of 6(number of classes) which then can be broken into probabilities using softmax and the index of the maximum probability can be used to predict the class label.

Thirteenth Layer: Softmax Layer

Softmax layer will produce the probabilistic output vector that corresponds to the the probability distributions of potential outcomes of the label. The output vector 1X6 from the fully connected layer is fed into softmax layer to get the probabilistic vector of 1 X 6.

The Softmax function

$$S(y_i) = \frac{e^{y}i}{\sum_j e^{y_i}}$$

Where y is the input vector values.

C. *Model Testing Phase*

During testing phase in the model, the test images of variable is resized to 100 X 100 X 3 as we have to maintain the shape of each image while giving input to the model Each test image from the dataset will passed in to the forward propagation of thirteen layer and the final vector 1 X 6 probabilistic values will be produced.

D. *Tabulation of Model Input,Parameters and Output.*

Layers	Input Size	Parameters	Output Size
INPUT	Variable Size Image		100 X 100 X 3
Convolution	104 X 104 X 3	No of filters=6, Kernel size=5, Padding=2, Stride=1, Learning rate=0.01	100 X 100 X 6
ReLu	100 X 100 X 6		100 X 100 X 6
Max-pooling	100 X 100 X 6	Pool size=2, stride=2	50 X 50 X 6
Convolution	54 X 54 X 6	No of filters=16, Kernel size=5, padding=2, stride=1, Learning rate=0.01	50 X 50 X 16
ReLu	50 X 50 X 16		50 X 50 X 16
Max-pooling	50 X 50 X 16	Pool size=2, stride=2	25 X 25 X 16
Convolution	29 X 29 X 16	No of filters=72, Kernel size=5, padding=2, stride=1, Learning rate=0.01	25 X 25 X 72
ReLu	25 X 25 X 72		25 X 25 X 72
Flatten	25 X 25 X 72		I X 45000
Fully Connected	I X 45000	No outputs=36, Learning rate= 0.01	1 X 36
ReLu	1 X 36		1 X 36
Fully Connected	1 X 36	No outputs=6, Learning rate= 0.01	1 X 6
SoftMax	1 X 6		1 X 6
OUTPUT			1 X 6

Table 1

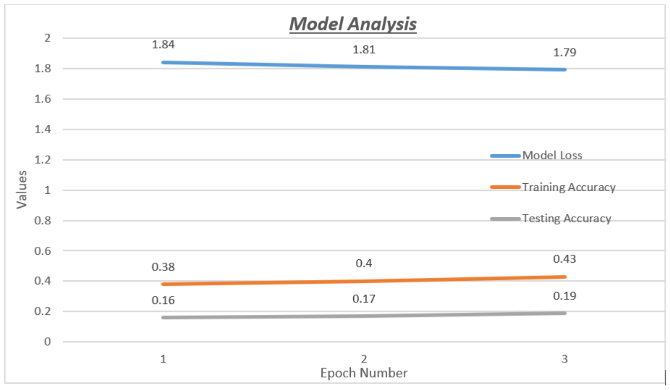
IV. RESULT AND ANALYSIS

We trained our CNN model with the learning rate of 0.01, batch size of 30, stride of 2, padding size of 2. All the kernel size of the convolution layer was kept as 5X5. We build the model initially by increasing the feature maps to 72 and decreased at the end to 6 to predict out of those 6 classes.

We tried with different kernel sizes of 3X3, 5X5, 7X7 but 5X5 was giving the best training accuracy. We trained our modified LeNet model with 3 epochs on 1200 Images (taking 200 images for each class). The losses fluctuated in the range of 1.5-1.8. At the end of the 3rd epoch we were getting the loss of 1.79. The table 2 below shows the cross-entropy loss, training accuracy, testing accuracy with respect to the epoch number trained.

Epoch Number	Loss	Training Accuracy	Testing Accuracy
1	1.84	0.38	0.16
2	1.81	0.40	0.17
3	1.79	0.43	0.19

Table 2



Graph 1

We observed that after the 3rd epoch that the losses got stabilized and predicted some of the images and getting their probabilities. We concluded that the model was able predict the street class with high confidence index or probability of around

0.329 as compared to the other classes. We also observed that there were a lot false positives which were happening because of the class mismatching. As in the images of street classes it was also containing some buildings and trees which was getting confused with the building and forest class. Similarly it was happening between sea and glacier class. The reason behind the class mismatching is that the features are getting shared among the classes which is causing issue thereby decreasing the testing accuracy while prediction and also reducing the prediction confidence values or probabilities. We tried to change the learning rate with 0.001 but observed that the training got slow with no significant impact on the training accuracy and the model loss.

Graph 1 shows the variation of model loss, training accuracy and testing accuracy with respect to the epoch number. We got the testing accuracy of 0.19 on all the 1200 images. We could have achieved a much higher accuracy if we would have trained the model for a greater number of epochs but the model was created from scratch (using only NumPy) so it was quite slow to train so we trained for only 3 epochs.

V. Conclusion and Future work

In this report, we have mentioned about the how the convolutional neural networks can be utilized in the image classification. We have used the improvised version of the LeNet architecture (which was created by Yann Lecun [2]). The Convolutional neural networks are the best to extract the features from the images and to train on it. We trained a CNN from scratch for the scene classification into 6 classes. The accuracy of the results can be improved if could create a denser model and use some other optimization techniques like Adagrad and Adams optimizers for weights updating. The training could have been done for a greater number of epochs in order to achieve lesser loss and better accuracy. Nowadays a lot of famous architectures like ResNet [3], AlexNet [6], InceptionNet [3], etc have been created to give good results which are considered as the state of the art algorithms in the field of deep learning. This project can be extended to classify multiple scene in a single image which is called semantic segmentation or pixel-wise classification and into the image captioning using recurrent neural networks. Convolutional neural networks are in great advancements in the field of computer vision and this work can also be extended in the real-time videos for the multiple scene recognitions. One of the major challenges this project is the accuracy as we require a lot of data to train the model and need to have a denser model and a lot of fine tuning the parameters to achieve that good level of accuracy.

References

[1] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 5 2015.

[2] 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, Dec 1989.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions, in The IEEE Conference on Computer Vision and Pattern Recognitio (CVPR), June 2015.

[4] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, “Handwritten digit recognition with a back-propagation network,” in Advances in Neural Information

[5] Processing Systems 2, D. S. Touretzky, Ed. Morgan-Kaufmann, 1990, pp. 396–404. [Online].

[6] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.