Scene Recognition using Deep Learning Networks

Satyam Bhikadiya
CS Department, Erik Jonsson School of Engineering
The University of Texas at Dallas
Richardson, TX, USA
sxb180124@utdallas.edu

Tejas Gupta
CS Department, Erik Jonsson School of Engineering
The University of Texas at Dallas
Richardson, TX, USA
txg180021@utdallas.edu

Abstract*— In the recent past, *Machine Learning has experienced a paradigm shift in ways unexpected. One of the major building bricks in this area is the development of Convolutional Neural Networks (CNN). In today's world, the consumption of audio and visual data has increased tremendously, and CNN is one of the best deep learning algorithms which can be run on huge datasets and gives good results. In this project, we are implementing a Scene Recognition on a dataset from Kaggle. This dataset contains a wide range of images classified in 6 groups viz. Buildings, Street, Glacier, Mountains, Sea and Forests. The main aim of this project is to be able to correctly classify which image belongs to the six of the mentioned categories. We obtained a training accuracy of around 40% and testing accuracy of around 30%.

Keywords—Convolutional, Neural, Network, Activation, Classification, image recognition, deep neural networks, max pooling layers, SoftMax layer

Introduction

In this world of social media, we find that images and videos are a medium of expression of sentiments and how a person feels. Nowadays, we live among images and videos. Image classification specifically, has many applications which we use in our day to day activities. Straight from unlocking our phones to reading manuscripts which are handwritten. This is accomplished by viewing it as a supervised learning problem. From the images that we already have, we try to figure out the images and what the images try to convey by using Convolutional Neural Networks. CNNs have proved to be the best method of image classification in the history of ML algorithms. In this project, the task is to train a machine to be able to distinguish between 6 types of scenes and further test it's accuracy. The algorithm we have implemented is the Convolutional Neural Network.

I. Convolutional Neural Networks

Convolutional Neural Networks (also CNN, ConvNets) is a class of deep neural networks which is mainly used to analyze images and videos. They are based on the principle of shift invariance. CNNs are a regularized and more complex versions of multilayer perceptrons. These perceptrons are usually fully connected. A fully connected neural network is the one in which all the neurons of a layer are connected to the next layer. CNNs are derived by synonymity to biological processes. The neurons are organized as such in an animal.

Due to the fully connectedness behavior of the CNNs, they are prone to overfitting. Such a problem is tackled by regularizing the data. For regularization in CNNs, the hierarchical nature of the data is taken advantage of. The more complex patterns are considered in their simpler forms and patterns. Thus, making it easy for analysis.

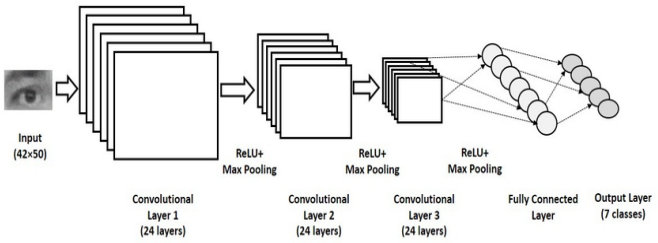

Fig.: Simple CNN

In the figure above, we can see a block diagram for a simple CNN. CNN consists of an input layer output layer and a series of other hidden layers. Hidden layers is consisted of major components such as the Convolutional Layer, Activation Function. MaxPooling Layer, Flattening Layer, Fully Connected Layer, Softmax Layer. The hidden layers are in multiple of series of arrangement of Convolutional Layers, Activation Layers, Pooling Layers and more if required by the system for further refinement.

II. Advantages of Convolutional Neutral Networks over Conventional Neural Networks

Although, convolutional neural networks and conventional neural networks (CoNN) are Multilevel perceptron fundamentally. Convolutional neural networks have many advantages over the conventional neural networks, mainly due to the architectural differences. In CNNs, there is a requirement of convolutional layer, fully connected layer, softmax layer, etc. In the CoNNs, there is a requirement of stochastic gradient descent, activation function (such as tanh, sigmoid, ReLu, etc.), Batch Normalization, etc.

Advantages of CNN over CoNN:

1. Feature generation and classification are unified in a single system.
2. CNNs have the ability to learn from the hierarchical nature of the data
3. Simpler computations of complex features

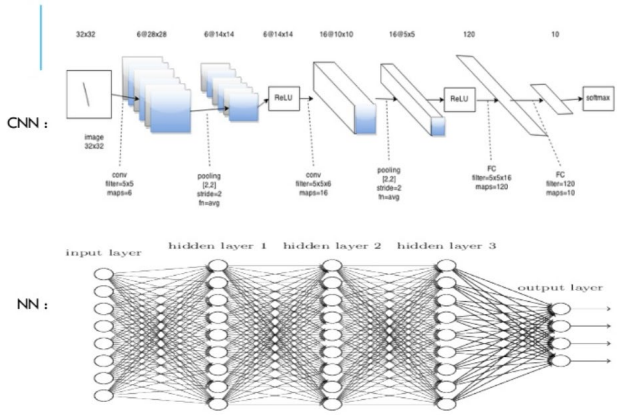

Fig.: Difference between Convolutional Neural Network and Conventional Neural Network

In the following section, we will discuss each of the components in detail.

III. COMPONENTS OF CNN

A. INPUT FORMAT

The input to a CNN is in the form of tensors. A tensor is an algebraic object that describes a multilinear relationship between elements in space. Tensors can be scalar or vectors. The dimensions of an input tensor are given as,

number of images x height x width x depth

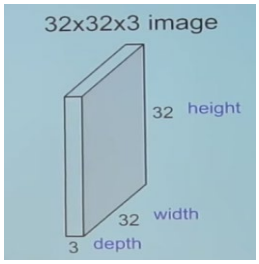

Fig.: Input tensor

B. CONVOLUTIONAL LAYER

The convolutional layer is the very first layer in the CNN and is used to extract features from the input image. The layer convolves the image to find the feature map by taking the dot product of the input image matrix and kernel or filter.

A generally devised convolutional network has the following features:

i. Kernel with defined width and height.
ii. Number of input and output channels
iii. Depth of filter must be equal to number of channels in input feature map

The kernel mentioned above is also called a filter. The main idea of convolution is to find out the number of different ways the filter can be placed on the input image for that layer.

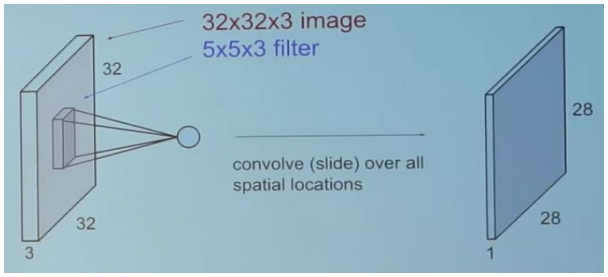

Fig.: Functionality of a Convolutional Layer

In the above figure, we see that the image of dimensions 32x32x3 is passed an input. The filter/kernel of dimensions 5x5x3 is used for convolution. The dot in the middle signifies a scalar value obtained after computing the dot product of the filter with all the possible placements on the input image. The last image with dimensions 28x28x1 is the result of the convolution on the input layer by the filter.

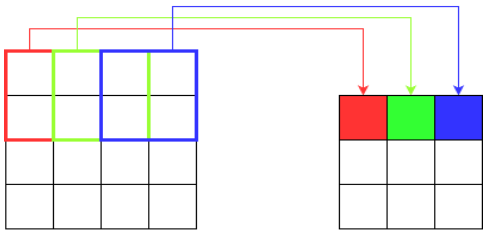

Fig.: Understanding Stride (stride = 1)

There is another important thing to consider, that can change the dimension and overall approximation of the result. It is called the 'stride'. A stride is a parameter that refers to the number of pixels to be shifted. The above convolution is for stride 1. The convolution layer has multiple independent filters and each of the filters are used for convolving the input independently.

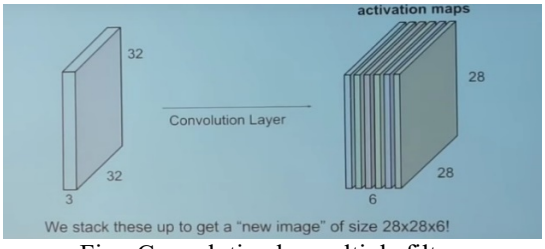

Fig.: Convolution by multiple filters

In the image above, we use 6 independent filters and therefore, we obtain a stacked-up result of dimensions 28x28x6 that is 6 stacks of 28x28x1.

For each convolutional layer, there is a supporting activation layer. The activation layer normally used in CNNs is ReLu. We have implemented Leaky ReLu activation function.

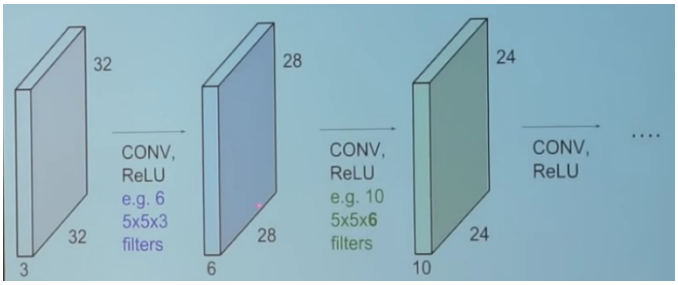

Fig.: Convolution Process by series of Convolutional Layers and ReLu layers.

C. ACTIVATION FUNCTION

An activation function aids the neuron in making decision whether the neuron should fire or not. Firing of a neuron is a term used for considering the neuron's output in the decision. A neuron generally takes the summation of the weighted inputs to come up with an output for that particular neuron. The output of the neuron can be anything between negative infinity and positive infinity. Now, it is upto the acitvation function to decide whther the neuron will be 'activated' or not.

There are a lot of different types activation function used in Machine Learning

1. Step Activation Function
2. Linear Activation Function
3. Sigmoid Activation Function
4. Tangent Hyperbolic Function
5. ReLu Activation Function
6. Leaky ReLu Activation Function

In CNNs, the ReLu activation function is used widely. In our project, we are using Leaky ReLu activation function for finding new results and differences than the conventional method.

ReLu Activation Function is given by:

$$\text{ReLu}(x) = \max(x, 0)$$

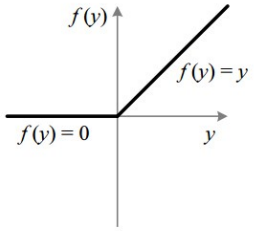

Fig.: Graphical Representation ReLu Activation Function

The Leaky ReLu Activation function is given by:

$$\text{Leaky ReLu}(a, x) = x \quad : x > 0$$
$$= ax \quad : x < 0$$

where 'a' is small negative slope. a is normally considered to be 0.01

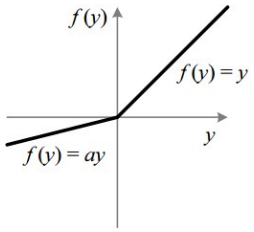

Fig.: Graphical Representation of Leaky ReLu

D. MAXPOOLING LAYER

The output from a convolutional layer is the matrix that has the features of the input. Our goal in the pooling layer is to minimize the amount of data stored for the input while retaining the important structural elements of the data.

For this, the data is stored in a lower resolution version of the input. This can be achieved by down sampling and changing the stride of convolution across the image.

The pooling layer is devised after the activation layer. Thus, forming a set of three layers as one unit in the order Convolutional Layer, Activation Layer and Pooling Layer. This set can be repeated as many times as required by the system for better accuracy.

There are two types of pooling, Average pooling and Maximum pooling. We have used the Max Pooling layer. In the max pooling layer, we calculate the maximum of each part of the feature value.

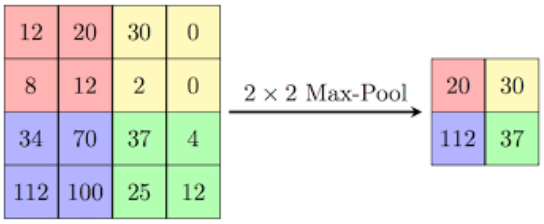

Fig.: Representation of max pooling function

E. FLATTENING LAYER

The flattening layer is used essentially reducing the data from n x n dimensions to 1 x (number of classes) dimension to form an input for the next layer. It is like putting all the pixels of the image in a single line for further ease of calculations in the fully connected layer.

F. FULLY CONNECTED LAYER

Now we have come towards the end of the CNN now we need to think of this problem as a simple multilevel perceptron problem. A fully connected network means that all the neurons on a level are connected to the next neurons on the next layer. We will now generate a fully connected network of n classes received from the flattening layer. The input is the result of the number of pixels from all the layers above. The output of the fully connected layer is the number of classes as defined by the dataset. This number is basically the desired number of classifications to be made in the problem statement.

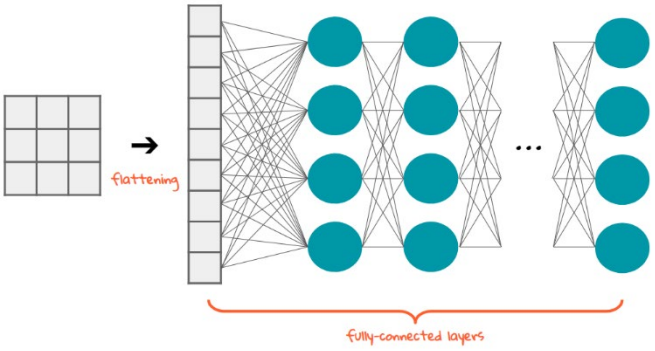

Fig.: Flattening Layer and Fully Connected Layer

G. SOFTMAX LAYER

The final layer of the CNN is the softmax layer. In this layer, we use the softmax function to convert/normalize the vector of real values to a vector of real values such that their sum is equal to 1. These values can be positive, negative or even zero. As these values are greater than -1 and less than +1, these values can also be understood as probabilities for each of the classes. This function is used in multiclass classification. One of the requirements for using it is the classes must be mutually exclusive. The softmax function is given by:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

IV. USE OF MOMENTUM FOR WEIGHT UPDATE

In neural networks, it is very important to put emphasis on the way to update weights of the neurons. In CNNs, we update weights in the convolutional layer and the fully connected layer. We have in this project, used the concept of momentum for updating weights.
The formula for updating weights is given by:

$$\mathbf{v}_t \leftarrow \beta \mathbf{v}_{t-1} + \mathbf{g}_{t,t-1},$$
$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \eta_t \mathbf{v}_t.$$

V. IMPLEMENTATION

In this project, we have implemented the convolutional neural network for scene recognition. It includes the following steps:

- Data Preprocessing
- Generating Class Labels for Training and Testing Datasets
- Training the CNN Model
- Testing the CNN Model

For training and testing the model, we have built the Convolutional Neural Network. For innovative effort, we have done the weight update using the concept of momentum. For the activation layer, we have used the Leaky ReLu activation function. We calculated the error using the cross entropy loss function.

Our model consists of 10 layers.
These layers are:

i. Convolutional Layer 1
ii. Leaky ReLu Activation Layer 1
iii. Maxpooling Layer 1
iv. Convolutional Layer 2
v. Leaky ReLu Activation Layer 2
vi. Flattening Layer
vii. Fully Connected Layer 1
viii. Leaky ReLu Activation Layer 3
ix. Fully Connected Layer 2
x. SoftMax Layer

Now let us discuss each of them breifly with respect to specific implementation in our model.

A. Data Preprocessing

Our dataset is divided in two folders one for training and testing each. Under each of them, we have six folders of specifically each of the categories of scenes we wish to classify the images with. We first append all the images in the training folder and make a large array of all the images. Then we convert the images into grayscale and subtract the mean of each column from each element in the column. We do the same with the testing

B. Generate Labels for Training and Testing Dataset

After getting a combined array of the training dataset, we now generate labels for each of the rows in the array. We do this in a similar fashion as we did with combining the training dataset. We take all the class label from each of the folders and combine them to form an array of all class labels. Same is done for generating testing labels.

C. *Training the CNN Model*

For training the CNN model, as mentioned above, we have implemented 10 layers.
The input and output for each of the layers is described in the following section:

i. Convolutional Layer 1

Input : 100 x 100 x 3
Kernel size : 5 x 5 x 6
Stride : 1
Padding : 2
Output : 100 x 100 x 6

ii. Leaky ReLu Activation Layer 1

Input : 100 x 100 x 6
Output : 100 x 100 x 6

iii. Maxpooling Layer 1

Input : 100 x 100 x 6
Kernel size : 2 x 2
Stride : 1
Pool Size : 2
Output : 50 x 50 x 6

iv. Convolutional Layer 2

Input : 50 x 50 x 6
Kernel size : 5 x 5 x 16
Stride : 1
Padding : 2
Output : 50 x 50 x 6

v. Leaky ReLu Activation Layer 2

Input : 50 x 50 x 16
Output : 50 x 50 x 16

vi. Flattening Layer

Input : 50 x 50 x 6
Output : 1 x 40000

vii. Fully Connected Layer 1

Input : 1 x 40000
Output : 1 x 36

viii. Leaky ReLu Activation Layer 3

Input : 1 x 36
Output : 1 x 36

ix. Fully Connected Layer 2

Input : 1 x 36
Output : 1 x 6

x. SoftMax Layer

Input : 1 x 6
Output : 1 x 6

D. *Testing the CNN Model*

For testing purpose, we use the dataset which is preprocessed at the beginning. Then we called the test function to test the model against the training it recently obtained.

VI. RESULT AND ANALYSIS

After multiple runs of the model on different epochs and learning rate, we found out that our system was able to identify the scenes in various cases.
The results for the model are given as follows:

Epoch	Learning Rate	Error	Training Accuracy	Testing Accuracy
2	0.01	1.84	36%	21%
	0.1	1.84	34%	21%
3	0.01	1.79	46%	26%
	0.1	1.80	44%	28%
5	0.01	1.71	51%	31%
	0.1	1.72	49%	29%

We have observed some insights about the dataset and also maybe the reason as to why the accuracies are lower than expected. Our system was supposed to classify among 6 classes such as Streets, Buildings, Forests, Glaciers, Sea and Mountains. Upon checking the dataset, we found out that the main reason for lower accuracies is that they model got mistaken for the images of streets which has trees and buildings for forests and building instead of streets themselves. In the same way, the glaciers and seas, mountains and forests were confused for each other.

VII. CONCLUSION AND FUTURE SCOPE

In this project, we have implemented a Convolutional Neural Network for Scene Classification using Leaky ReLu and momentum updates for weight updates. In this report, we have started with giving an idea of convolutional neural networks, difference between conventional and convolutional neural networks, detailed information about the key components of CNN, implementation of our model and it's results and observations. For further implementations, we can optimize the propagation functions for better results. Also, with the use of other activation functions such as Bent Identity, Square Non-Linearity, SReLu, etc.

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, Dec 1989.

[2] S. Lawrence, C. L. Giles, Ah Chung Tsoi and A. D. Back, "Face recognition: a convolutional neural-network approach," in IEEE Transactions on Neural Networks, vol. 8, no. 1, pp. 98-113, Jan. 1997, doi: 10.1109/72.554195.

[3] https://en.wikipedia.org/wiki/Convolutional_neural_network

[4] https://d2l.ai/chapter_optimization/momentum.htmlR. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[5] https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

[6] https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8