Cs512- Computer Vision

Kinjal Kachi

Hawk id: A20449343

Abstract

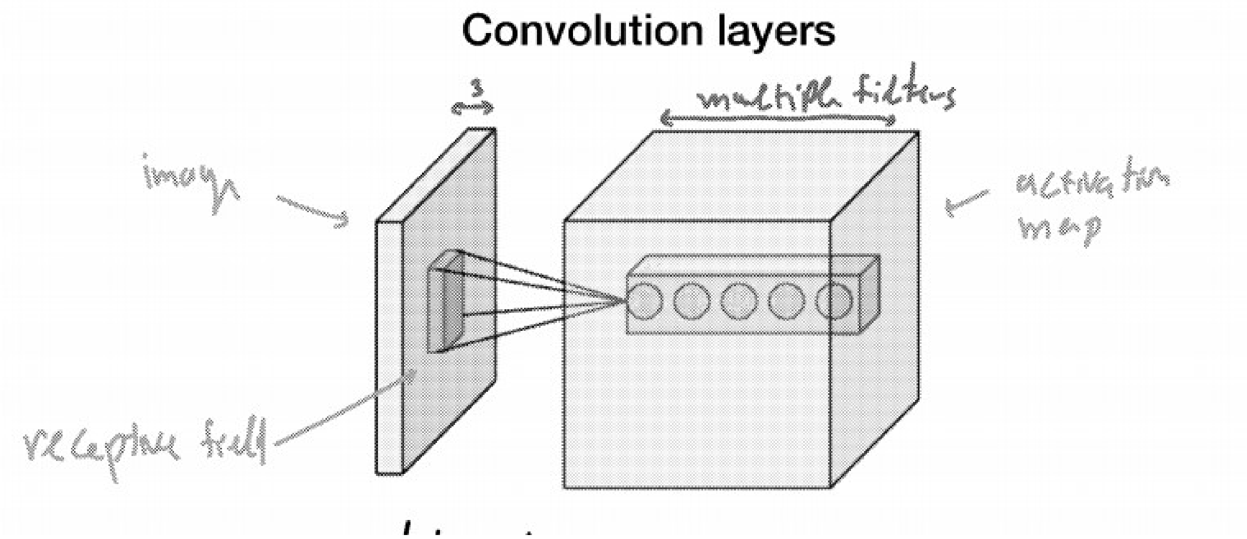
The coding part of this assignment aims at making students understand the concept of constructing convolution neural network, adding different parameters like activation function, loss function, the concept of hyper parameter tuning using strides, receptive field, batch and layer normalization, weight initializers. This coding assignment gives students an opportunity to test and evaluate their models using the above-mentioned concepts.

Problem Statement:

1) Construct and train Convolution Neural Network.
2) Hyper-parameter Tuning: Evaluate different variations of the basic network and measure performance.
3) Inference: Write a program to use your pretrained custom Convolution Neural Network.

Proposed Solution:

Build Convolution layers using the concepts below:



- Extract image patched.
- Vectorize image window and filter and perform dot product.
- Filter extends full depth of the image.
- Use multiple convolution filters per location.
- Use stride to move filter.
- Activation map may be smaller.

Use and manipulate parameters like Activation function, Loss function, strides, dropout parameters, weight initializer. The significance of these parameters are as follows:

Activation Function: In artificial neural networks, the activation function of a node defines the output of that node given an input or set of inputs.

Loss function: the perceptual loss function is used when comparing two different images that look similar, like the same photo but shifted by one pixel or same images across different resolutions.

Strides: Stride is a component of convolutional neural networks, or neural networks tuned for the compression of images and video data. Stride is a parameter of the neural network's filter that modifies the amount of movement over the image or video. For example, if a neural network's stride is set to 1, the filter will move one pixel, or unit at a time.

Dropout: Dropout is an effective way of regularizing neural networks to avoid the overfitting of ANN. During training, the dropout layer cripples the neural network by removing hidden units stochastically.

Weight Initializer: The aim of weight initialization is to prevent layer activation outputs from exploding or vanishing during a forward pass through a deep neural network.

Batch and layer normalization: Batch normalization normalizes the input features across the batch dimension. The key feature of layer normalization is that it normalizes the inputs across the features.

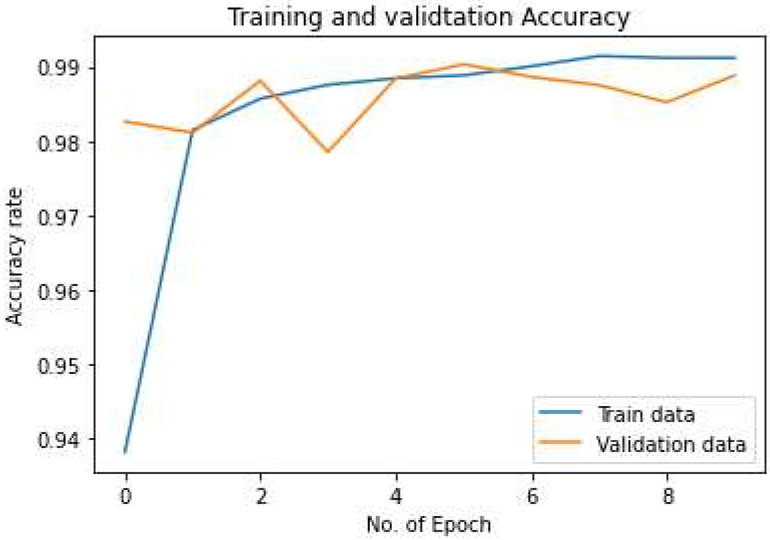
Implantation details:

1) Split data in test, train and validation dataset as mentioned in the coding question.
   No. of records Training data: 55000
   No. of records in validation data: 10000
   No. of records in testing data: 5000
2) Convert labels as odd and even odd = 0, even = 1.
3) Build convolution neural network with activation function: sigmoid because output has two levels.
4) Compile the model with Stochastic Gradient Descent optimizer and Binary Cross Entropy as the loss function. We use accuracy metric for model evaluation. Fit the training data to model.
5) Perform Hyperparameter tuning, first increase the number of filters and kernel size and observe the result. Keep the epoch size less to have a lesser training time. Test different parameters such as stride, dropout, optimizers, epoch size and kernel initializer.
6) Use the optimum value of above parameters and build an efficient model along with batch and layer normalization.

Results and discussions:

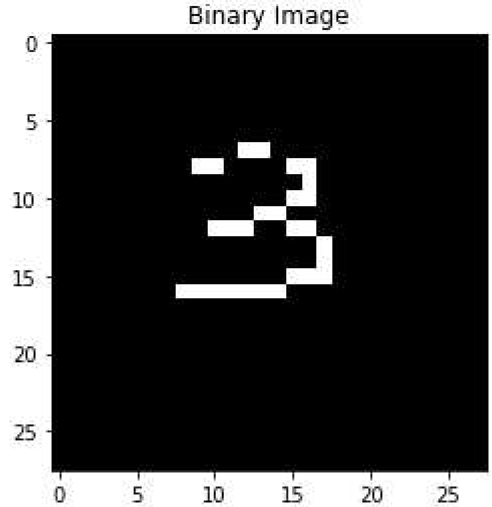
1) During hyper-parameter tuning choose optimum stride value as (2,2) because it gives good accuracy with optimum time consumption.
2) Dropout time 0.3 shows highest accuracy.
3) We use initializers like he_uniform, and glorot_uniform both give same performance, choose any of one, I am going with default value, glorot_uniform
4) I tunned for upto 12 epochs but after 10 epochs we see that the model does not improves in terms of accuracy. Hence, we select 10 epochs for training our model.
5) Hence, final parameters:
   Stride = (2,2)
   Dropout = 0.3
   Weight Initializer = Xaiver/"glorot_uniform"
   Epochs = 10
6) Add batch and layer normalization to the model.

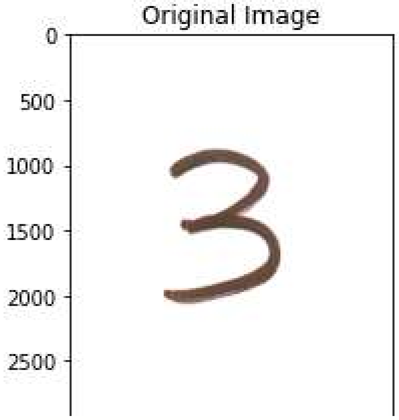
With these the accuracy of the model is:



Inference Results using the above model.

```
Image is odd numbered
```


Binary Image


Original Image