Digit classification and recognition using convolution neural network using MNISt dataset , We have trained the data using CNN with manipulation of training options to get desired training option. NOn-maximum suppression and thresholding is implementation to obtain digit predictions, MNIST dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.

**Method:-**

Convolution neural network could considered as gradable NN which generally works in extraction of features of images by choosing options of convolving input with the help gaggle of kernel filters. After that the process of pooling of obtained feature is executed and filter resolution of next layer following, Now implementing convolution neural network algorithm.

Above given condition tells us about the lth layer of ith map, piece channel of lth layer related with the ith maps in theLayer and index map

Set In the layer.

 convolution operation is represented by above equation . activation function is represented by f. which is known ReLU non-linear f (z)= max(0,z ), bias is represented as

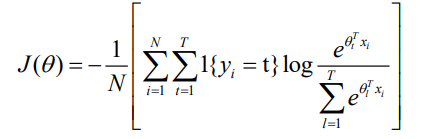
 equation of pooling is represented as -

Where sum-sampling function is down(.).

 softmax regrssion is used for multiclass classifiction and proved to be an effective method , assume for a given information you have T classes ,preparing the preparation information for the every classification are indicated as where i={1,……N} .

 are feature vector arelable apart

function known as Cross entropy loss given as-

Here θ is model parameters, Normalization factor is

Training options =

TrainingOptionsSGDM with properties:

Momentum: 0.9000

InitialLearnRate: 0.0100

LearnRateScheduleSettings: [1×1 struct]

L2Regularization: 1.0000e-04

GradientThresholdMethod: 'l2norm'

GradientThreshold: Inf

MaxEpochs: 100

MiniBatchSize: 128

Verbose: 1

VerboseFrequency: 30

ValidationData: {[28×28×1×10000 double] [10000×1 categorical]}

ValidationFrequency: 30

ValidationPatience: Inf

Shuffle: 'once'

CheckpointPath: ''

ExecutionEnvironment: 'auto'

WorkerLoad: []

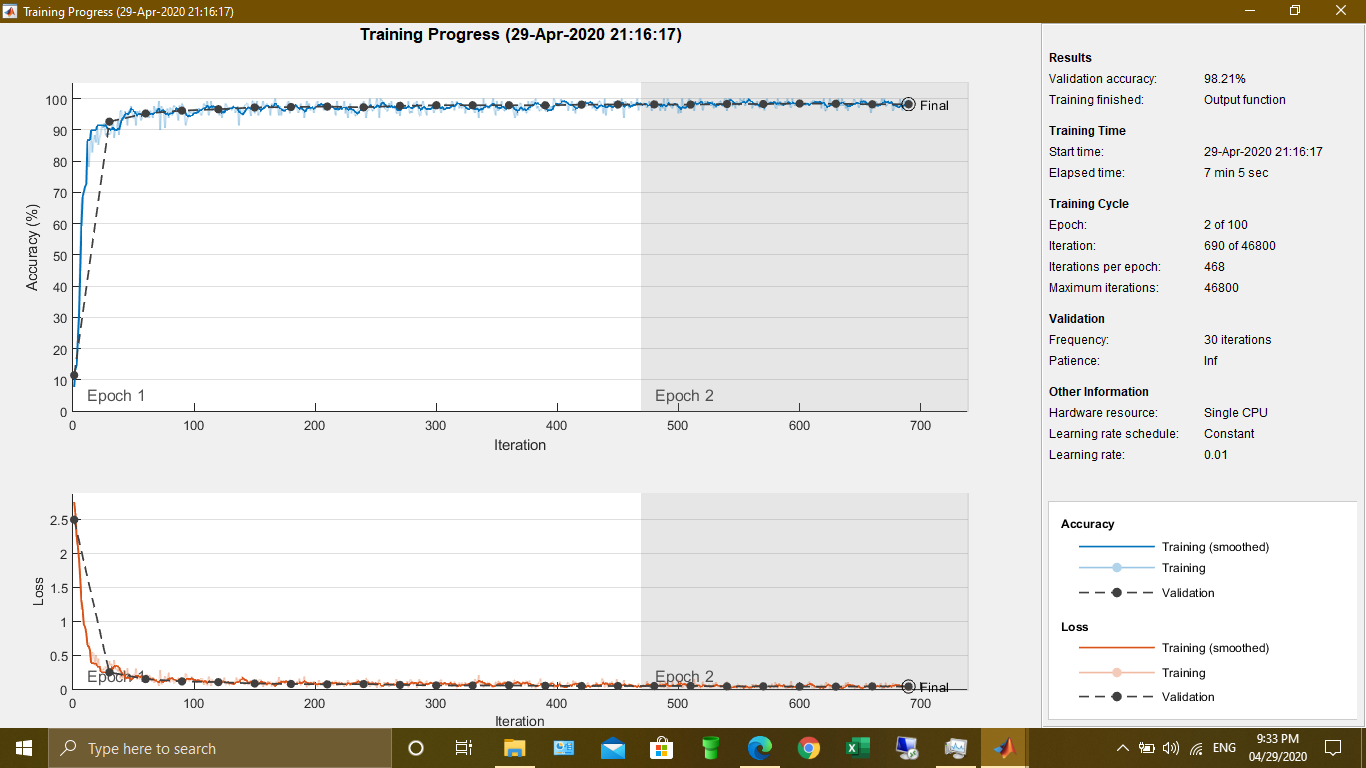
OutputFcn: @(info)stopIfAccuracyNotImproving(info,3)

Plots: 'training-progress'

SequenceLength: 'longest'

SequencePaddingValue: 0

Training accuracy:-



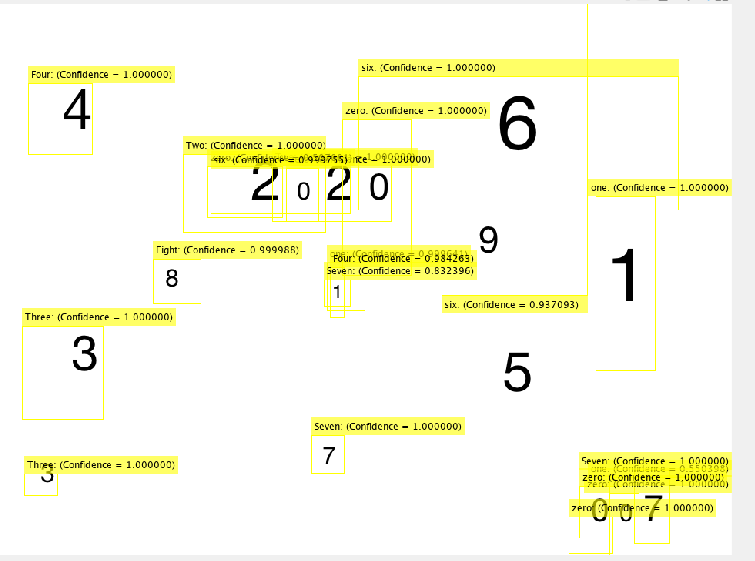
Train mean accuracy: 0.983667

Train mean cross entropy: 0.0528463

Test mean accuracy: 0.9821

Test mean cross entropy: 0.0520467

Recognition of digit on trained network



MATLAB CODE

addpath('datasets');

[XTrain, YTrain, XTest, YTest] = load\_dataset('mnist');

%%

% nb\_features = 256;

% nb\_classes = 10;

layers = [

imageInputLayer([28 28 1])

convolution2dLayer(3,8,'Padding','same')

batchNormalizationLayer

reluLayer

maxPooling2dLayer(2,'Stride',2)

convolution2dLayer(3,16,'Padding','same')

batchNormalizationLayer

reluLayer

maxPooling2dLayer(2,'Stride',2)

convolution2dLayer(3,32,'Padding','same')

batchNormalizationLayer

reluLayer

fullyConnectedLayer(10)

softmaxLayer

classificationLayer];

options = trainingOptions('sgdm', ...

'InitialLearnRate',0.01, ...

'MaxEpochs', 100, ...

'MiniBatchSize', 128, ...

'VerboseFrequency',30, ...

'ValidationData',{XTest, YTest}, ...

'ValidationFrequency', 30, ...

'Plots','training-progress','OutputFcn',@(info)stopIfAccuracyNotImproving(info,3))

%%

load net\_mnist.mat

%net = trainNetwork(XTrain, YTrain,layers,options);

%%

YPred = predict(net,XTrain);

acc = mean\_accuracy( YTrain, YPred );

ce = mean\_cross\_entropy( YTrain, YPred );

fprintf( 'Train mean accuracy: %g\n', acc );

fprintf( 'Train mean cross entropy: %g\n\n', ce );

YPred = predict(net,XTest);

acc = mean\_accuracy( YTest, YPred );

ce = mean\_cross\_entropy( YTest, YPred );

fprintf( 'Test mean accuracy: %g\n', acc );

fprintf( 'Test mean cross entropy: %g\n\n', ce );

save( 'net\_mnist.mat', 'net' );

%%

load network.mat

% Extract the first convolutional layer weights

w = cifar10Net.Layers(2).Weights;

% rescale the weights to the range [0, 1] for better visualization

w = rescale(w);

%%

% Load the ground truth data

load maindata.mat

%%

T = T(:, {'imageFilename','six','one','zero','Two','Three','Seven','Eight','Four','Nine','Five'});

%%

% Display one training image and the ground truth bounding boxes

I = imread(T.imageFilename{3});

I = insertObjectAnnotation(I,'Rectangle',T.six{1},'six','LineWidth',8);

I = insertObjectAnnotation(I,'Rectangle',T.one{1},'one','LineWidth',8);

I = insertObjectAnnotation(I,'Rectangle',T.zero{1},'zero','LineWidth',8);

figure

imshow(I)

%%

load ntworkk.mat

%%

% Train an R-CNN object detector. This will take several minutes.

% rcnn = trainRCNNObjectDetector(T, cifar10Net, options, ...

% 'NegativeOverlapRange', [0 0.3], 'PositiveOverlapRange',[0.5 1])

%%

% Read test image

testImage = imread('computer\_generated.png');

imshow(testImage)

% Detect stop signs

[bboxes,score,label] = detect(rcnn,testImage,'MiniBatchSize',128)

%%

% Display the detection results

% [score, idx] =score

% bbox = bboxes(idx, :);

for idx=1:22

annotation = [sprintf('%s: (Confidence = %f)', label(idx), score(idx));];

le{idx}=annotation;

end

outputImage = insertObjectAnnotation(testImage, 'rectangle', bboxes, le);

imshow(outputImage)

pause(1)

for i=1:22

w=label(i,:);

tts(char(w));

end

% hold on

%%