Experiment 10: CNN

Table of Contents

Aim: 1
Theory: 1
Convolutional neural network 1
Confusion Matrix 6
Testing image 1 9
Testing image 2 10
Conclusion: 11

Name: Ventrapragada Sai Shravani

PRN:17070123120

Batch:Entc(2017-21) G-3

Aim:

Implementation of Convolutional Neural Network (CNN) for any dataset

Theory:

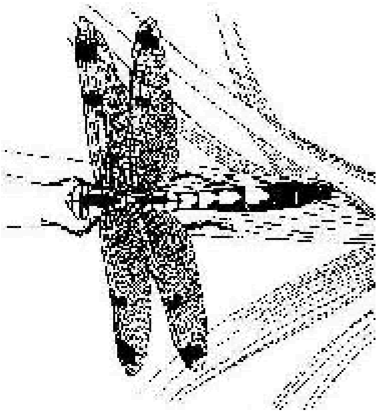
CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values. Stride is the number of pixels shifts over the input matrix. Pad the picture with zeros ReLu is used for activation function Pooling layers section would reduce the number of parameters when the images are too large The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network.

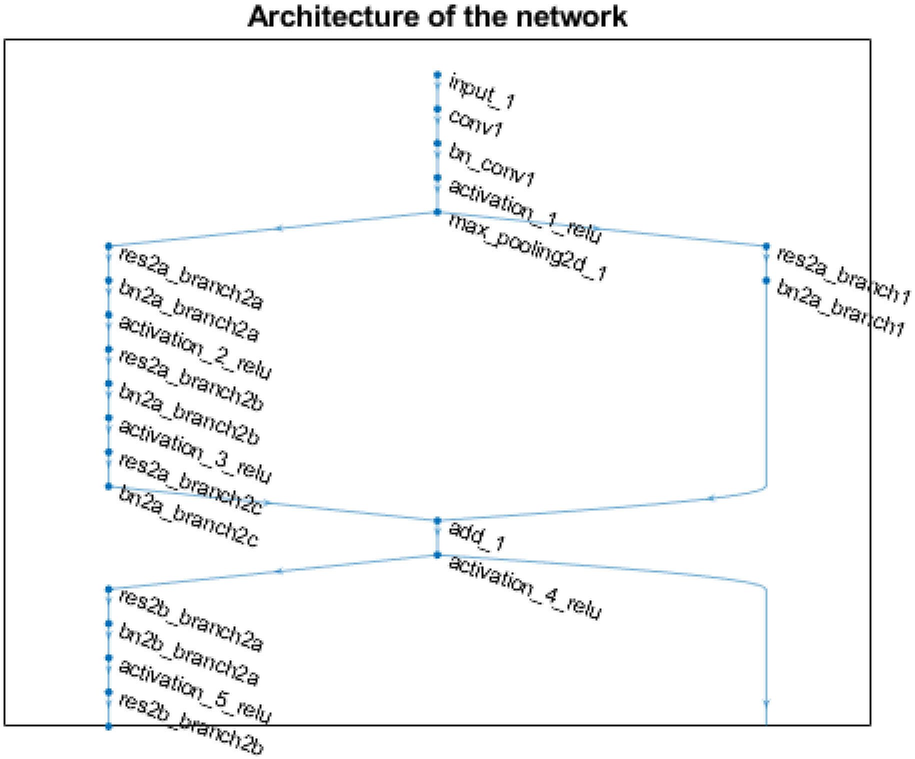
Convolutional neural network

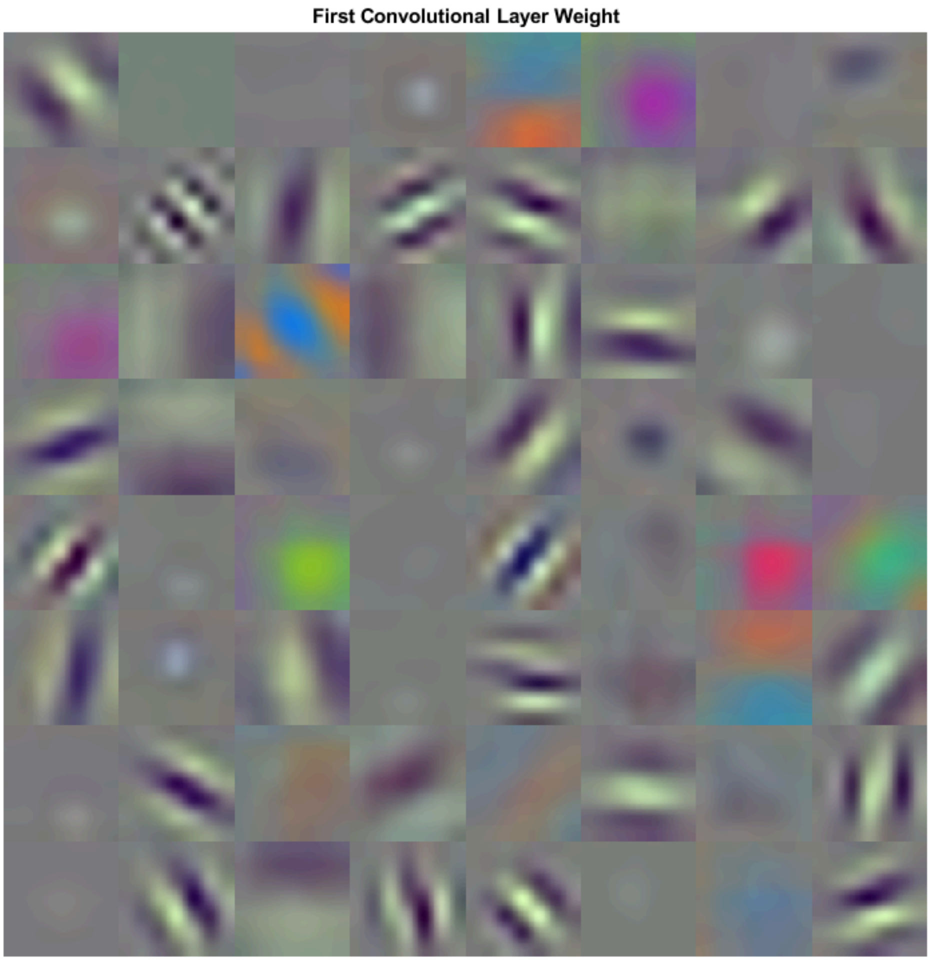
```
clc;
clear all;
close all;

outputFolder=fullfile('101_ObjectCategories');% to create file path
categories={'dragonfly','butterfly'};
imds =
 imageDatastore(fullfile(outputFolder,categories),'LabelSource','foldernames'); %i
 and categories stored
tbl = countEachLabel(imds)
imds = splitEachLabel(imds,68, 'randomize'); % taking equal number of
 images for both the categories randomly
tbl = countEachLabel(imds)
```

```
dragonFly = find(imds.Labels == 'dragonfly', 1);
butterFly = find(imds.Labels == 'butterfly', 1);
figure
subplot(121)
imshow(readimage(imds,dragonFly))
subplot(122)
imshow(readimage(imds,butterFly))


net = resnet50(); %Predefined function
figure();
plot(net) % Plotting the network
title('Architecture of the network');
set(gca, 'YLim', [150, 170]); % resizing the figure gca- It loads the
 current data, YLim- Y Limit, Region of limit

net.Layers(1) % Input layer 1 properties
net.Layers(end) % Input layer 1 properties
% There are 1000 classes in this neural network

[trainingSet , testSet]= splitEachLabel(imds, 0.4, 'randomize'); %
 Using 40 % of the data for training and rest for testing
imageSize= net.Layers(1).InputSize;
augmentedTrainingset=augmentedImageDatastore(imageSize,
 trainingSet, 'ColorPreprocessing', 'gray2rgb'); %resizing the images
 to the required image size
% grayscale images to rgb
augmentedTestset=augmentedImageDatastore(imageSize,
 testSet, 'ColorPreprocessing', 'gray2rgb'); %resizing the images to
 the required image size

w1 = net.Layers(2).Weights; %inputing previous output to input
w1= mat2gray(w1); %Matrix to gray scale W1

% Plotting w1
figure();
montage(w1)
title('First Convolutional Layer Weight')

featureLayer ='fc1000';
trainingFeatures= activations(net, augmentedTrainingset,
 featureLayer, 'MiniBatchSize',32 , 'OutputAs', 'columns'); %Minibatchsize
 is set to 32 for fitting the GPU memory the activations output is
 arranged in columns

% levels of the training set
trainingLabels=trainingSet.Labels;
classifier= fitcecoc(trainingFeatures,
 trainingLabels, 'Learner', 'Linear', 'Coding', 'onevsall', 'ObservationsIn', 'col
 uses binary support vector models for error free neural network

testFeatures= activations(net, augmentedTestset,
 featureLayer, 'MiniBatchSize',32 , 'OutputAs', 'columns'); %Minibatchsize
 is set to 32 for fitting the GPU memory the activations output is
 arranged in columns

predictLabels= predict(classifier,
 testFeatures, 'ObservationsIn', 'columns'); %Predicted class levels
testLabels=testSet.Labels;
```

```
tbl =

  2×2 table

      Label      Count
    _________    _____

    butterfly     91
    dragonfly     68


tbl =

  2×2 table

      Label      Count
    _________    _____

    butterfly     68
    dragonfly     68


ans =

  ImageInputLayer with properties:

                     Name: 'input_1'
                InputSize: [224 224 3]

   Hyperparameters
         DataAugmentation: 'none'
            Normalization: 'zerocenter'
    NormalizationDimension: 'auto'
                     Mean: [224×224×3 single]


ans =

  ClassificationOutputLayer with properties:

            Name: 'ClassificationLayer_fc1000'
         Classes: [1000×1 categorical]
      OutputSize: 1000

   Hyperparameters
    LossFunction: 'crossentropyex'
```

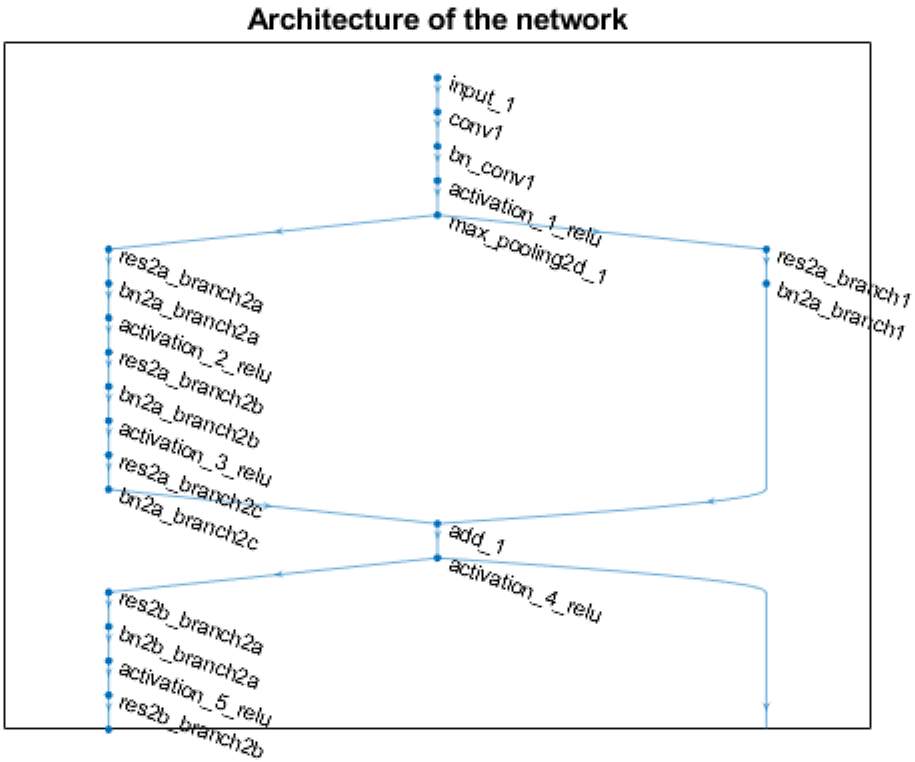


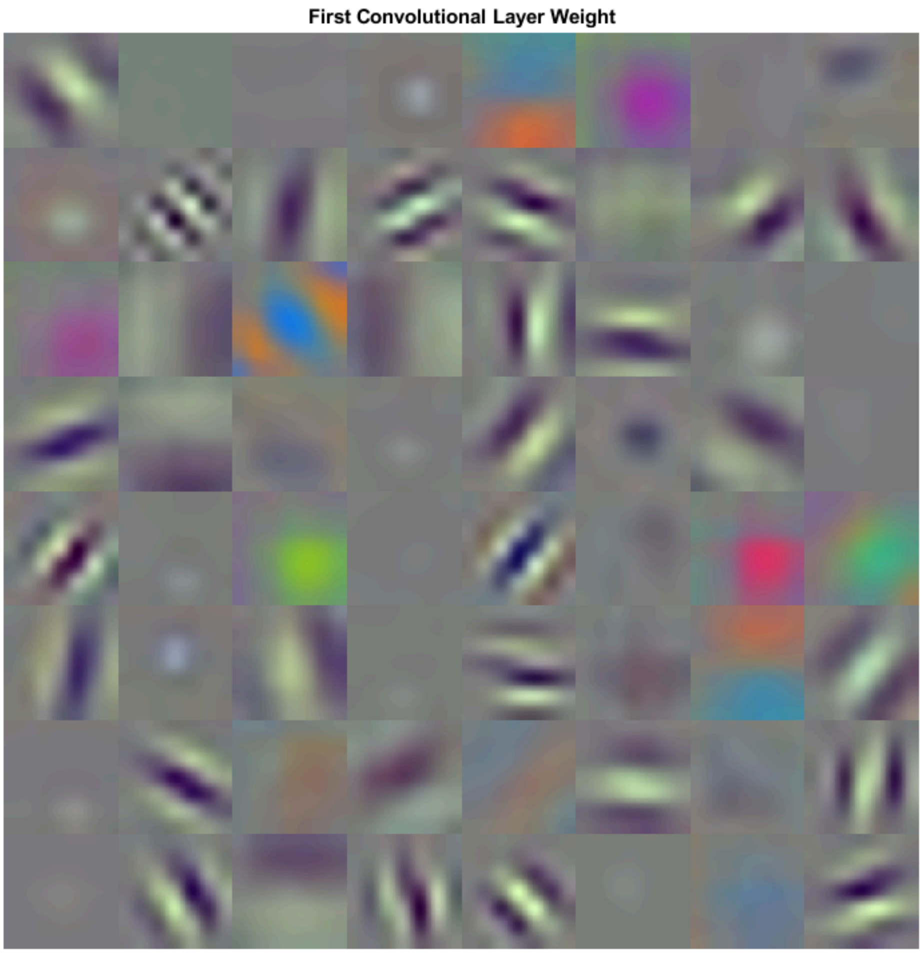

Architecture of the network
input_1
conv1
bn_conv1
activation_1_relu
max_pooling2d_1
res2a_branch2a
bn2a_branch2a
activation_2_relu
res2a_branch2b
bn2a_branch2b
activation_3_relu
res2a_branch2c
bn2a_branch2c
res2a_branch1
bn2a_branch1
add_1
activation_4_relu
res2b_branch2a
bn2b_branch2a
activation_5_relu
res2b_branch2b

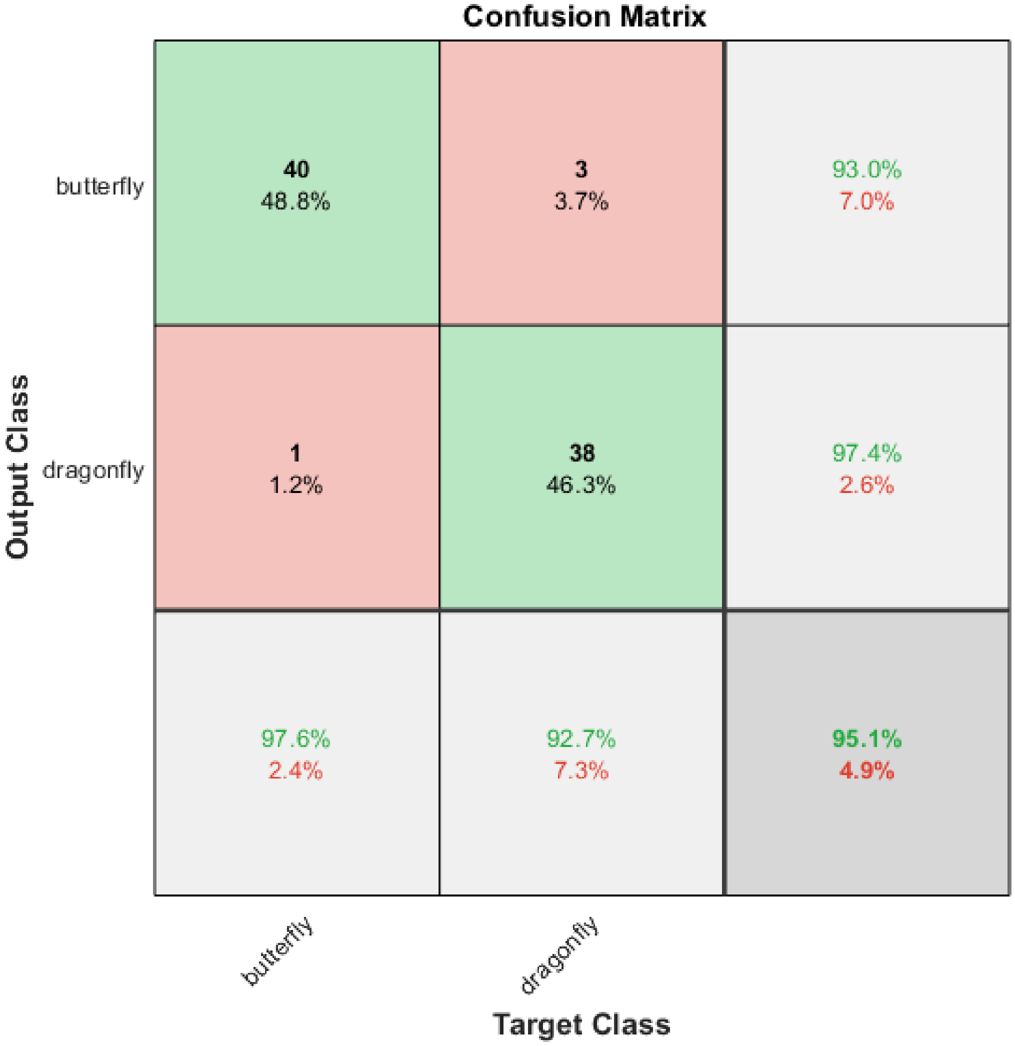


Confusion Matrix

```
figure()
plotconfusion(testLabels,predictLabels) %plot the network training
 data
```


Architecture of the network
input_1
conv1
bn_conv1
activation_1_relu
max_pooling2d_1
res2a_branch2a
bn2a_branch2a
activation_2_relu
res2a_branch2b
bn2a_branch2b
activation_3_relu
res2a_branch2c
bn2a_branch2c
res2a_branch1
bn2a_branch1
add_1
activation_4_relu
res2b_branch2a
bn2b_branch2a
activation_5_relu
res2b_branch2b

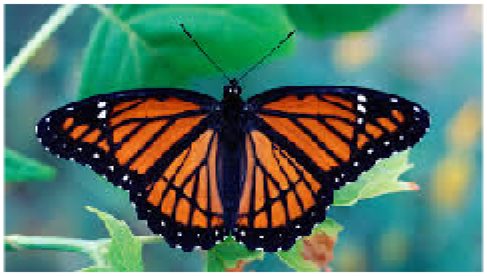

First Convolutional Layer Weight



Testing image 1

```
newImage= imread(fullfile('butterfly_test.jpg'));
figure();
imshow(newImage);
ds= augmentedImageDatastore(imageSize,
 newImage, 'ColorPreprocessing', 'gray2rgb');
imageFeatures= activations(net,ds,
 featureLayer, 'MiniBatchSize',32 , 'OutputAs', 'columns');
imageLabels= predict(classifier,
 imageFeatures, 'ObservationsIn', 'columns');
sprintf('The loaded image belongs to %s class', imageLabels)
```

```
ans =

    'The loaded image belongs to butterfly class'
```



Testing image 2

```
newPicture= imread(fullfile('dragonfly_test.jpg'));
figure();
imshow(newPicture);
Ds= augmentedImageDatastore(imageSize,
 newPicture, 'ColorPreprocessing', 'gray2rgb');
pictureFeatures= activations(net,Ds,
 featureLayer, 'MiniBatchSize',32 , 'OutputAs', 'columns');
pictureLabels= predict(classifier,
 pictureFeatures, 'ObservationsIn', 'columns');
sprintf('The loaded image belongs to %s class', pictureLabels)
```

```
ans =

    'The loaded image belongs to dragonfly class'
```

Conclusion:

In the above program I classified images of butterfly and dragonfly. I used the inbuilt fuction resnet50 which is an inbuilt fuction, we plotted the 1 convolution layer, saw the layer 1 and layer 1000 properties

then using fitcecoc to get the levels of the training set plotted the confusion matrix to see the data distribution I also used activations for image extracting features and then put them into labels

I also tested two images of butterfly and dragonfly to show that the neural network works

Published with MATLAB® R2020a