UE17EC327

ARTIFICIAL NEURAL NETWORKS

COURSE PROJECT

Image Classification Using Convolutional Neural Networks

Team:

Shreyas Kulkarni - PES1201700450

Rahul R.K – PES1201701552

Shobith Nandakumar – PES1201700507

Problem Statement:

Configure a convolutional neural network to classify high dimensional images.

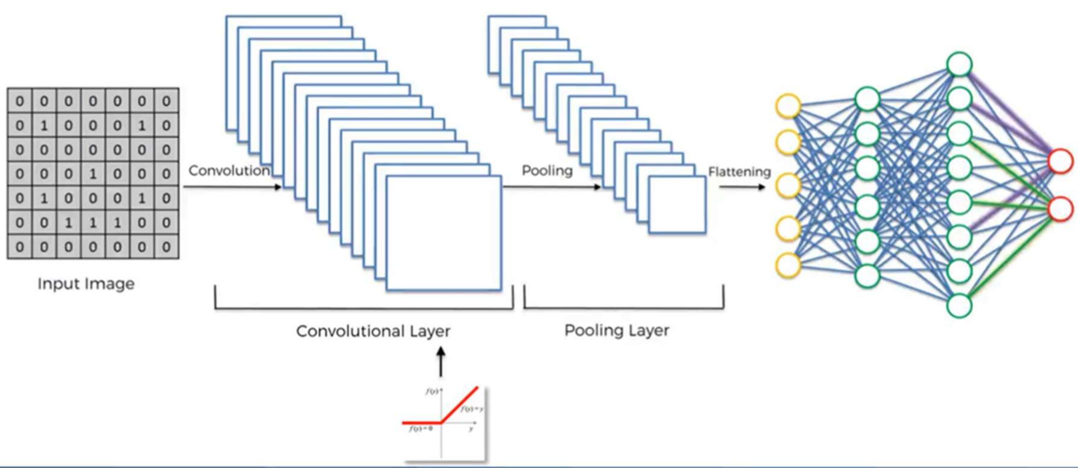
CNN:

CNN is a deep-learning algorithm that takes in input data, extracts important features based on some parameters and classifies using a fully connected neural network.

CNNs are more accurate over other networks because:

- They allow models to learn position and scale in variant structures in data, which is important when working with images.
- CNN retain spatial structure in images.
- They use fewer parameters in the learning procedure hence reducing unnecessary computation.
- Feature extraction helps compute only on relevant data which becomes very important when input dataset is of higher orders.

Structure of CNNs:



Consists of the following steps:

- Convolution layer followed by ReLu
- Pooling layer
- Fully-connected layer
- Soft-max layer

Convolution Layer:

- It extracts features from the input image.
- It preserves the relationship between pixels by learning image features using filters or masks.

Non-Linearity (ReLu):

- It non-linearizes the output of the convolution layer.
- $f(x) = \max(0,x)$.

Pooling Layer:

- Retains spatial integrity of image and reduces dimensionality further.

Fully Connected Layer:

- Does learning of the network using back-propagation algorithm.

Softmax Layer:

- It is an activation function that classifies the data.

Data-acquisition and pre-processing:

- Dataset chosen for this experiment is **blood-cell classes** taken from **Kaggle** an online community for data scientists and machine learning enthusiasts.
- Two classes of blood cells inside folder “Dataset”

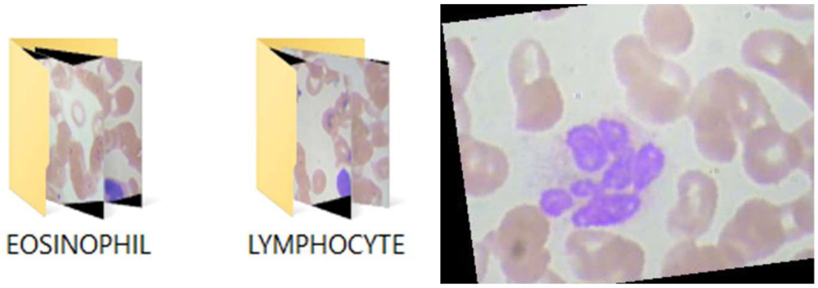


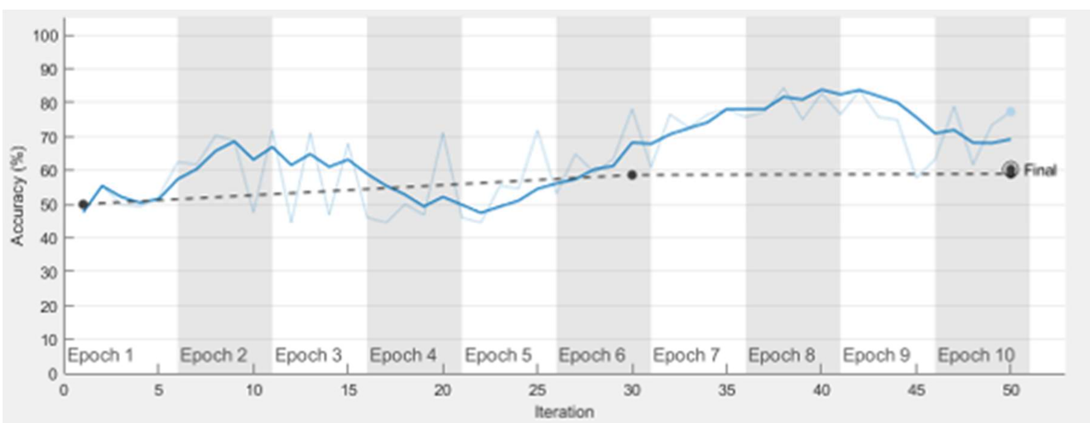
Image on the right is a blood cell shot under the microscope.

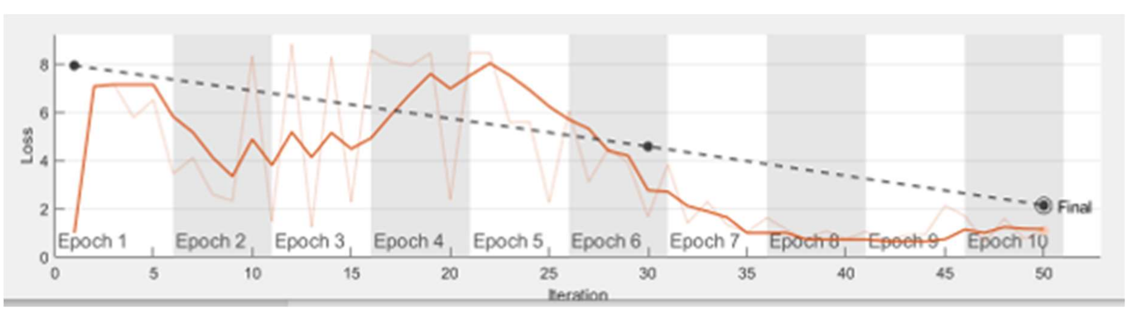
Code: [explanation commented]

```
a = imread('C:\Users\shobith\Documents\MATLAB\CNN
Proj\bloodcelltest.jpeg');

matlabroot = 'C:\Users\shobith\Documents\MATLAB'; %setting
root directory
DataSetPath = fullfile(matlabroot,'CNN Proj','Dataset');
%setting directory of dataset

Data =
imageDatastore(DataSetPath,'IncludeSubfolders',true,'LabelSour
ce','foldernames'); %get data as datastore

[dataTrain,dataValidation] =
splitEachLabel(Data,0.7,'randomize');
%split 70% of data as training and the rest as validation

inputSize = [240 320 3]; %image dimension
numClasses = 2;      %output classes

%define layers using deeplearning toolbox
Layers = [
 imageInputLayer(inputSize)

 convolution2dLayer(5,20)
 batchNormalizationLayer
 reluLayer

 maxPooling2dLayer(2,'stride',2)

 convolution2dLayer(5,20)
 batchNormalizationLayer
 reluLayer

 maxPooling2dLayer(2,'stride',2)

 dropoutLayer
```

```
 fullyConnectedLayer(numClasses)
 softmaxLayer
 classificationLayer

  ];

Options =
trainingOptions('sgdm','MaxEpochs',10,'initialLearnRate',0.001
','ValidationData',dataValidation,'ValidationFrequency',30,'Ve
rbose',false,'Plots','training-progress');
%give training options
%sgdm - stochastic gradient descent method
%no. of epochs given is 10
%initial learning rate is 0.001

Convnet = trainNetwork(dataTrain,Layers,Options);
%train the network

YPred = classify(Convnet,dataValidation);
%get predicted data for all validation datasets

YValidation = dataValidation.Labels;

accuracy = mean(YPred == YValidation) %print accuracy by
comparison
```

Matlab simulation Results:





```
Layers = 

  13x1 Layer array with layers:

     1   ''   Image Input             240x320x3 images with 'zerocenter' normalization
     2   ''   Convolution             20 5x5 convolutions with stride [1  1] and padding [0  0  0  0]
     3   ''   Batch Normalization     Batch normalization
     4   ''   ReLU                    ReLU
     5   ''   Max Pooling             2x2 max pooling with stride [2  2] and padding [0  0  0  0]
     6   ''   Convolution             20 5x5 convolutions with stride [1  1] and padding [0  0  0  0]
     7   ''   Batch Normalization     Batch normalization
     8   ''   ReLU                    ReLU
     9   ''   Max Pooling             2x2 max pooling with stride [2  2] and padding [0  0  0  0]
    10   ''   Dropout                 50% dropout
    11   ''   Fully Connected         2 fully connected layer
    12   ''   Softmax                 softmax
    13   ''   Classification Output   crossentropyex
```

```
 TrainingOptionsSGDM with properties:

                     Momentum: 0.9000
             InitialLearnRate: 1.0000e-03
    LearnRateScheduleSettings: [1×1 struct]
             L2Regularization: 1.0000e-04
      GradientThresholdMethod: 'l2norm'
            GradientThreshold: Inf
                    MaxEpochs: 10
                MiniBatchSize: 128
                      Verbose: 0
             VerboseFrequency: 50
               ValidationData: [1×1 matlab.io.datastore.ImageDatastore]
          ValidationFrequency: 30
           ValidationPatience: Inf
                      Shuffle: 'once'
               CheckpointPath: ''
         ExecutionEnvironment: 'auto'
                   WorkerLoad: []
                    OutputFcn: []
                        Plots: 'training-progress'
               SequenceLength: 'longest'
         SequencePaddingValue: 0
     SequencePaddingDirection: 'right'
         DispatchInBackground: 0
      ResetInputNormalization: 1
```

Accuracy: 70%. Can be improved with more number of training dataset inputs.

Test Results:

```
YPred =

  420×1 categorical array

     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
```

```
     2
     2
     1
     2
     2
     2
     1
     2
     1
     1
     1
     1
     2
     2
     2
     1
     2
```

- Above images represent first few results and the last few results.
- 1 – Eosinophile and 2- Lymphocyte
- Network classifies 210 Eosinophile followed by 210 Lymphocyte images.
- Therefore 420 YPredicted outputs.

Some other important results:

- If input was just fed into neural network, then number of input nodes would have been **240*320*3 = 2,30,400**
- After doing feature-extraction, the number of inputs to the fully-connected layer for learning is:

```
>> Convnet.Layers(11).InputSize

ans =

       87780
```

- Clearly by extracting important features of the images, we have reduce the computation required by a great deal and also are able to use high dimensional images for classification which wouldn’t have been possible if feature-extraction wasn’t done.

Applications and future scope:

- CNNs are already the most widely used neural network for large number of image classification problems.
- Used by large product-based industries such as amazon, Facebook etc for recommendation engines and image tag.
- For facial recognition and other security purposes.
- For predictive analytics in the medical field. This is a field that is researched heavily using CNNs. With enough training and resources, CNNs should be able to predict cancerous cells growth before they are fully formed.

References:

- Mathworks documentation.
- Towardsdatascience.com
- www.superdatascience.com