Brunda Chouthoy
CSC 578 – Final Project
Diabetic Retinopathy Image classification

INTRODUCTION

This evaluation project focuses on applying neural network methods to images for classification. The main advantage of using neural network methods for image data is its adeptness at fitting nonlinear data and its ability to work as an unsupervised algorithm. Diabetic retinopathy also known as diabetic eye disease, is when damage occurs to the retina due to diabetes. In this project, convolutional neural network models are trained and analyzed for the classification of diabetic retinopathy eye images obtained from Kaggle.

The goal is to gain a better understanding of how multilayer convolutional neural networks processes images - the CNN models use a complex architecture composed of stacked layers in which is particularly well-adapted to classify the images. For multi-class classification, this architecture robust and sensitive to each feature present in the images. Tensorflow environment is used for training and evaluating the CNN models. Accuracy and loss statistics obtained for different parameter configurations are considered for performance assessment of models.

Research Question:

Performance assessment of two multi-layer deep convolutional neural network models using the Python Tensorflow environment for identifying Diabetic Retinopathy(DR) in eye images.

Dataset description:

The data is a large set of high-resolution retina images taken under a variety of imaging conditions. A left and right field is provided for every subject. Images are labeled with a subject id as well as either left or right (e.g. 1_left.jpeg is the left eye of patient id 1)

Each image is in the RGB color format and the presence of diabetic retinopathy is indicated on a scale of 0 to 4, according to the following scale:

0 - No Diabetic retinopathy
1 - Mild
2 - Moderate
3 - Severe
4 - Proliferative DR

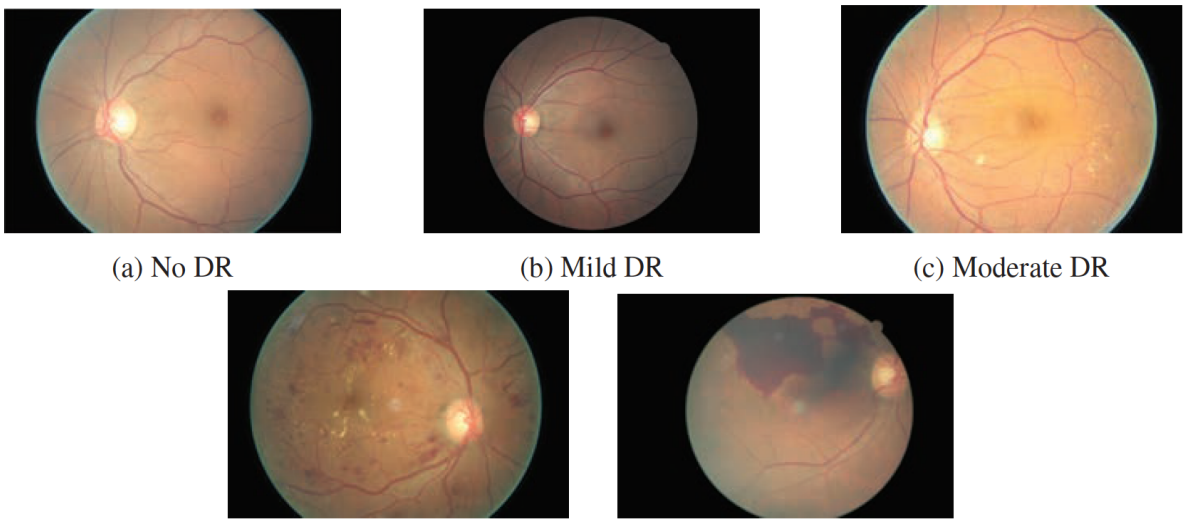
Dataset link: https://www.kaggle.com/c/diabetic-retinopathy-detection/data

Due to the extremely large size of this dataset (35126 train and 53,576 test high resolution images – around 64GB data) and limitations with my hardware a smaller subset of the original dataset is used for analysis.

Number of train images: 300

Number of validation images: 180

Stages of diabetic retinopathy: Sample images from the dataset



(a) No DR (b) Mild DR (c) Moderate DR

(d) Severe DR (e) Proliferative DR

Fig 1: Stages of diabetic retinopathy (DR) with increasing severity

EVALUATION METHODOLOGY

Pre-processing data:

A primary step involved for this analysis is pre-processing and transforming the input images. The images were retained in the RGB color format (3 channels – 3 values per pixel) and they were taken under a variety of imaging conditions. Most of them were also way too large to perform any non-trivial model fitting on them. A minimum preprocessing to make network training possible is to standardize the dimensions. Before feeding the images into the network for classification, the eye images were resized to the appropriate size with input dimensions specified for the model and image labels are transformed to one-hot encoded arrays.

Initialization of networks – CNN Image classification:

Two multi-layer CNN models are trained for evaluation and analysis of diabetic eye images.

Weights initialization: The weights for the CNN model are initialized with random Gaussian distribution of 0 mean and a standard deviation of 0.1.

Activation/Transfer function: ReLu (Rectified Linear Unit) is used as the activation function for CNNs.

Softmax Layer: Softmax layer being a standard classification layer for CNNs is applied to the fully connected layer in the network to classify diabetic eye images to the respective classes.

Cost function: Cross entropy with softmax with logits is used as the cost function

Dropout layer: Convolutional layers have inbuilt resistance to overfitting, so dropouts are applied to the fully connected layers to reduce overfitting of the model.

Model A – 2 convolution layers:

1 Input layer: Image reshaped to a 4D tensor 128*128 pixels, RGB format

2 Convolutional layers - Convolutions uses a stride of one and are zero padded so that the output is the same size as the input.

Convolution layer 1 – 5*5 convolution, 3 input channels (RGB format), 128 outputs

Convolution layer 2 – 5*5 convolution, 128 inputs and outputs

Pooling layer – Max pooling is used for down sampling over 2*2 blocks

Activation/transfer function 'Relu' is applied for the convolution layers

1 Fully/Densely connected layer - Flatten and reshape the tensor from the pooling layer into the batch of vectors and apple 'Relu' activation function and max pooling.

Fully connected 1 – 32*32*128 inputs, 128 outputs

1 Dropout layer – keep rate probability 0.8

1 Classification/output layer – 128 inputs, 5 outputs (class prediction)

Model B – 3 convolution layers:

1 Input layer: Image reshaped to a 4D tensor 512*512 pixels, RGB format

3 Convolutional layers - Convolutions uses a stride of one and are zero padded so that the output is the same size as the input.

Convolution layer 1 - 5*5 convolution, 3 input channels (RGB format), 64 outputs

Convolution layer 2 - 5*5 convolution, 64 inputs and 128 output channels

Convolution layer 3 - 5*5 convolution, 128 inputs and 256 output channels

Pooling layer – Max pooling is used for down sampling over 2*2 blocks

Activation/transfer function 'Relu' is applied for the convolution layers

1 Fully/Densely connected layer - Flatten and reshape the tensor from the pooling layer into the batch of vectors and apple 'Relu' activation function and max pooling.

Fully connected 1 – 64*64*256 inputs, 1024 outputs

1 Dropout layer – keep rate probability 0.8

1 Classification/output layer – 1024 inputs, 5 outputs (class prediction)

Training the network:

Batch generation: Custom batches are generated for the input images data with the specified batch size before feeding to the multi-layer CNN models.

ADAM optimizer with a learning rate of 0.001 is used as the optimizer to minimize loss. keep_prob is added in feed_dict to control the dropout rate. Accuracy and loss statistics are computed and displayed for each epoch step.

RESULTS

Model A – Image size: 128*128 pixels
Trial 1: Displaying Train and Validation accuracy for each epoch step
Epochs: 10, Batch Size: 100, Learning Rate: 0.001

```
Epoch: 001/010 cost: 1596.791992188
Training accuracy: 0.67667
Validation accuracy: 0.65556
Epoch: 002/010 cost: 718.808715820
Training accuracy: 0.64667
Validation accuracy: 0.58333
Epoch: 003/010 cost: 289.322919210
Training accuracy: 0.72333
Validation accuracy: 0.65556
Epoch: 004/010 cost: 55.413264434
Training accuracy: 0.74667
Validation accuracy: 0.67778
Epoch: 005/010 cost: 12.275577267
Training accuracy: 0.76000
Validation accuracy: 0.67222
Epoch: 006/010 cost: 1.604072809
Training accuracy: 0.75667
Validation accuracy: 0.67778
Epoch: 007/010 cost: 12.567846775
Training accuracy: 0.75667
Validation accuracy: 0.67778
Epoch: 008/010 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy: 0.67778
Epoch: 009/010 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy: 0.67778
Epoch: 010/010 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy: 0.67778
```

Best values for 10 epochs:
Train accuracy 0.76 or 76%
Validation accuracy 0.678 or 67.8%

Brunda Chouthoy
CSC 578 – Final Project
Diabetic Retinopathy Image classification

Trial 2: Displaying Train and Validation accuracy for every 10 epochs
Epochs: 100, Batch Size: 100, Learning Rate: 0.0001

```
Epoch: 001/100 cost: 1301.521036784
Training accuracy: 0.71000
Validation accuracy: 0.63889
Epoch: 010/100 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy: 0.67778
Epoch: 020/100 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy: 0.67778
Epoch: 030/100 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy: 0.67778
Epoch: 040/100 cost: 1.609437585
Training accuracy: 0.76333
Validation accuracy: 0.67778
Epoch: 050/100 cost: 1.609437585
Training accuracy:  0.76000
Validation accuracy: 0.67222
Epoch: 060/100 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy: 0.67778
Epoch: 070/100 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy: 0.67778
Epoch: 080/100 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy: 0.67778
Epoch: 090/100 cost: 1.609437585
Training accuracy: 0.76333
Validation accuracy: 0.67778
Epoch: 100/100 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy: 0.67778
```

Best values for 100 epochs:
Train accuracy: 0.76 or 76%
Validation accuracy: 0.678 or 67.8%

Model B – Image size: 512*512 pixels
Trial 1: Epochs: 10, Batch Size: 100, Learning Rate: 0.0001

```
Epoch: 001/010 cost: 241564.15
Training accuracy: 0.80000
Validation accuracy:  0.73333
Epoch: 002/010 cost: 243029.87
Training accuracy: 0.75999
Validation accuracy:  0.67333
Epoch: 003/010 cost: 76110.70
Training accuracy: 0.68999
Validation accuracy:  0.65556
Epoch: 004/100 cost: 16627.54
Training accuracy: 0.68000
Validation accuracy:  0.67778
Epoch: 005/100 cost: 1.609437585
Training accuracy: 0.76000
```

```
Validation accuracy:  0.67778
Epoch: 006/100 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy:  0.67778
Epoch: 007/100 cost: 25.932778438
Training accuracy: 0.75667
Validation accuracy:  0.67778
Epoch: 008/100 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy:  0.67778
Epoch: 009/100 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy:  0.67778
Epoch: 009/100 cost: 1.609437585
Training accuracy: 0.76000
Validation accuracy:  0.67778
```

Best values for 10 epochs:
Train accuracy: 0.800 or 80%
Validation accuracy: 0.733 or 73.3%

DISCUSSION OF RESULTS

For my analysis, I expected to test and evaluate two multi-layer convolutional neural networks and optimize hyper parameters to derive a relatively better model for classifying diabetic retinopathy eye images. Though I considered a smaller subset of the original dataset for analysis – the processing times were a big hurdle. Due to limitations in hardware and tensorflow environment setup, my analysis was restricted to a smaller dataset and lesser number of iterations/epochs.

The best accuracy values for the 3-convolutional layer model (model B) with 512*512 pixel images appears to perform slightly better than the model with 2 convolutional layers (model A). However, it is not possible to make any conclusions about the better performing model considering the number of input images and lesser number of iterations for each model. I considered using different image sizes (128*128 for model A and 512*512 pixels for model B) for each of the models to see if more data would make any additional impact to the accuracy of the model. But, there is no significant difference in the accuracy values between the two models and only resulted in large processing times.

The best accuracy values achieved for the diabetic retinopathy image data is for model B:
Train Accuracy: 0.80000 or 80.00%
Validation Accuracy: 0.73333 or 73.33%
Average Cost/loss: 241564.15 (Epoch 1 of 10)

CONCLUSION

Through this project, I understood the basics of image recognition using deep Convolution Neural Networks (CNNs). I gained a better understanding of some of the aspects of tuning a neural network such as activation functions, weights initialization and image preprocessing. Furthermore, I got some intuition into why deep CNNs should work better than traditional approaches and understood different elements present in a general deep CNN. This project has also allowed me to explore and gain familiarity with the Tensor flow libraries with much more depth.

Future work / Improvements:

To assess the performance with a larger dataset: By using Tensorflow library compiled for native CPU architecture i.e. by installing and building the library from sources on local machine OR by installing GPU version of Tensorflow. (I only had the CPU version of Tensorflow installed in my system)

Experiment with different multi-layer CNN models – by increasing the number of hidden/ convolution layers in the network

Tune the different adjustable hyper parameters for the network to optimize the results and find the best performing model for diabetic retinopathy classification.

REFERENCES

Tensor flow tutorials:

https://www.tensorflow.org/tutorials/layers

https://www.tensorflow.org/tutorials/deep_cnn

Convolutional Neural Networks for Diabetic Retinopathy - http://www.sciencedirect.com/science/article/pii/S1877050916311929

Detection and classification of diabetic retinopathy using retinal images - http://ieeexplore.ieee.org/document/6139346/

Convolutional Neural Networks with TensorFlow - Deep Learning with Neural Networks tutorial- https://pythonprogramming.net/cnn-tensorflow-convolutional-nerual-network-machine-learning-tutorial/

Stanford's CS231 course by Andrej Karpathy et Al., an interesting course to learn about CNNs for visual recognition - http://cs231n.stanford.edu/slides/2016/winter1516_lecture7.pdf

Brunda Chouthoy
CSC 578 – Final Project
Diabetic Retinopathy Image classification