Table of Contents

Machine Learning Engineer Nanodegree

Capstone Project

Vasudev Kartheek Akkur

April 11th, 2019

Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning - Retinal OCT (Optical Coherence Tomography)

I. Definition

Project Overview

In project we will develop an algorithm using convolution neural networks and transfer learning to process medical images to provide and accurate and timely diagnosis of key pathology in each image. The primary illustration of this technique involved optical coherence tomography (OCT) images of the retina but algorithm can be generalized to another image diagnosis.

Optical coherence tomography (OCT) is an imaging technique that uses coherent light to capture high resolution images of biological tissues. OCT is heavily used by ophthalmologists to obtain high resolution images of the eye retina. Retina of the eye functions much more like a film in a camera. OCT images can be used to diagnose many retina related eyes diseases.

With OCT an ophthalmologist can see each of the retina's distinctive layers. This allows your ophthalmologist to map and measure their thickness. These measurements help with diagnosis. They also provide treatment guidance for glaucoma and diseases of the retina. These retinal diseases include age-related macular degeneration (AMD) and diabetic eye disease.

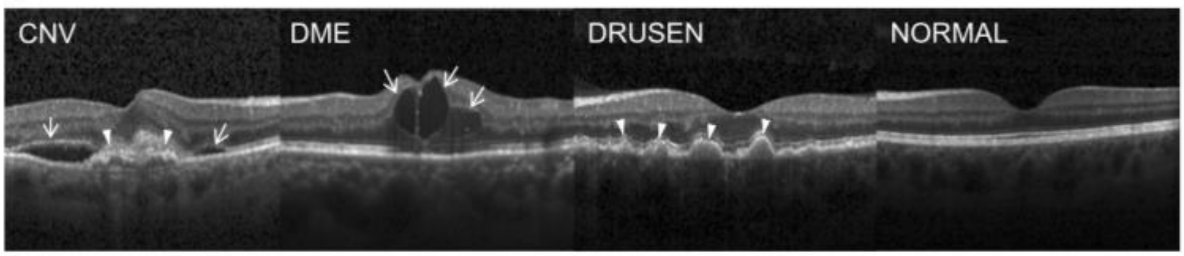
We will implement an algorithm that will classify a given image into four class labels

1. **Choroidal neovascularization (CNV):** Choroidal neovascularization is the creation of new blood vessels in the choroid layer of the eye. Choroidal neovascularization is a common cause of neovascular degenerative maculopathy commonly exacerbated by extreme myopia, malignant myopic degeneration, or age-related developments.
2. **Diabetic Macular Edema (DME):** DME is a complication of diabetes caused by fluid accumulation in the macula that can affect the fovea. The macula is the central portion in the retina which is in the back of the

eye and where vision is the sharpest. Vision loss from DME can progress over a period of months and make it impossible to focus clearly.

3. **Drusen:** Drusen are yellow deposits under the retina. Drusen are made up of lipids, a fatty protein. Drusen likely do not cause age-related macular degeneration (AMD). But having drusen increases a person's risk of developing AMD. Drusen are made up of protein and calcium salts and generally appear in both eyes.
4. **Normal**

A



(A) (Far left) choroidal neovascularization (CNV) with neovascular membrane (white arrowheads) and associated subretinal fluid (arrows). (Middle left) Diabetic macular edema (DME) with retinal-thickening-associated intraretinal fluid (arrows). (Middle right) Multiple drusen (arrowheads) present in early AMD. (Far right) Normal retina with preserved foveal contour and absence of any retinal fluid/edema.

Problem Statement

The implementation of clinical-decision support algorithms for medical imaging faces challenges with reliability and interpretability. Here, we establish a diagnostic tool based on a deep-learning framework for the screening of patients with common treatable blinding retinal diseases. Our framework utilizes transfer learning, which trains a neural network with a fraction of the data of conventional approaches. Applying this approach to a dataset of optical coherence tomography images, we demonstrate performance comparable to that of human experts in classifying age-related macular degeneration and diabetic macular edema.

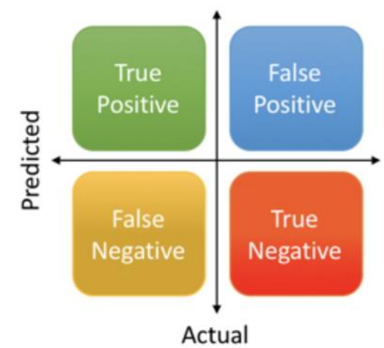
Metrics

We will compute the accuracy, confusion matrix, precision and recall for the model and compare the results with the benchmark model.

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$



The confusion matrix for the predicted model will give the True positives, True Negatives, False Positives and False Negatives. From this we can compute the precision, recall and accuracy of the model. Precision means the percentage of the results which are relevant. On the other hand, recall refers to the percentage of total relevant results correctly classified by the model. Accuracy of the model is total correct predictions to the total samples present in the model.

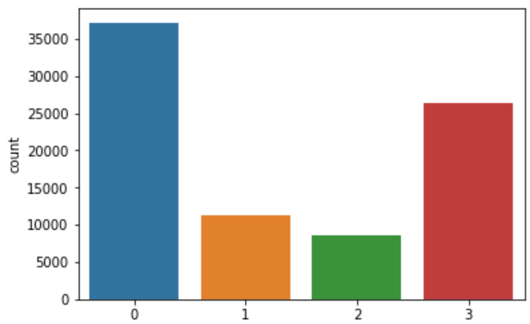
$$\textbf{F1 Score} = 2 * \frac{\text{Precision * Recall}}{\text{Precision + Recall}}$$

F1-score, which is simply the harmonic mean of precision and recall, the greater the F-1 score the better the model is performing.

II. Analysis

Data Exploration

The dataset is organized into 3 folders (train, test, Val) and contains subfolders for each image category (NORMAL, CNV, DME, DRUSEN). There are 84,495 X-Ray images (JPEG) and 4 categories (NORMAL, CNV, DME, DRUSEN).



It is an imbalanced dataset which we deal with using **Image Augmentation** and **Transfer learning**. Training dataset contains below number of images from the figure

0. DME = 37205
1. DRUSEN = 11348
2. NORMAL = 8616
3. CNV = 26315

Images are labeled as (disease)-(randomized patient ID) -(image number by this patient) and split into 4 directories: CNV, DME, DRUSEN, and NORMAL.

Optical coherence tomography (OCT) images (Sp-vizectralis OCT, Heidelberg Engineering, Germany) were selected from retrospective cohorts of adult patients from the Shiley Eye Institute of the University of California San Diego, the California Retinal Research Foundation, Medical Center Ophthalmology Associates, the Shanghai First People's Hospital, and Beijing Tongren Eye Center between July 1, 2013 and March 1, 2017.

Dataset:

https://www.kaggle.com/paultimothymooney/kermany2018

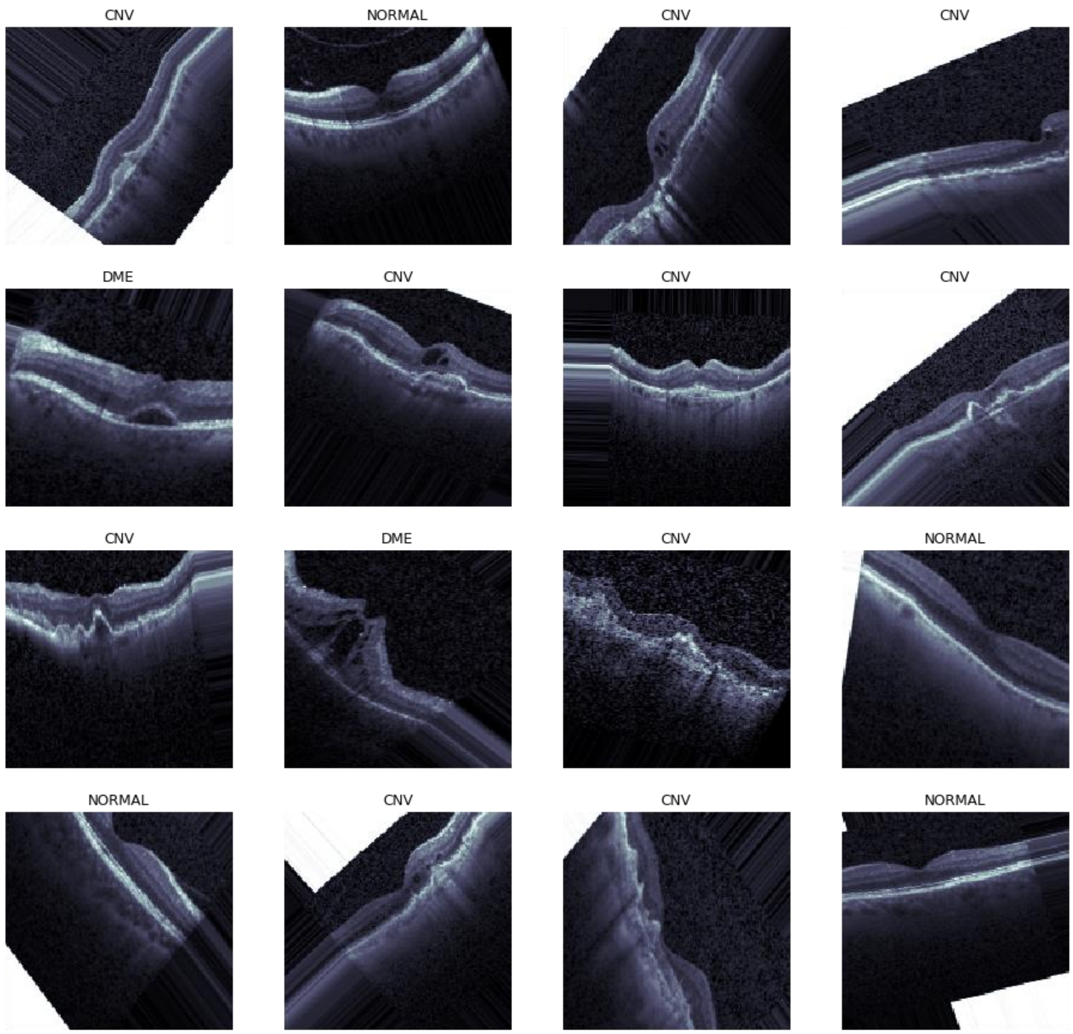
Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification http://dx.doi.org/10.17632/rscbjbr9sj.2#file-9e8f7acf-7d3a-487f-8eb5-0bd3255b9685

Exploratory Visualization

The below images show the different conditions that we will be training in the from the train dataset.

In case of CNV you look at the new blood vessels that are formed and compare it NORMAL, it difficult to distinguish between them but looking at the edges can spot deviations.

In case of DME you look at steep edges that are being formed.



Before training, each image went through a tiered grading system consisting of multiple layers of trained graders of increasing expertise for verification and correction of image labels. Each image imported into the database started with a label matching the most recent diagnosis of the patient. The first tier of graders consisted of undergraduate and medical students who had taken and passed an OCT interpretation course review. This first tier of graders conducted initial quality control and excluded OCT images containing severe artifacts or significant image resolution reductions. The second tier of graders consisted of four ophthalmologists who independently graded each image that had passed the first tier. The presence or absence of choroidal neovascularization (active or in the form of subretinal fibrosis), macular edema, drusen, and other pathologies visible on the OCT scan were recorded. Finally, a third tier of two senior independent retinal specialists, each with over 20 years of clinical retina experience, verified the true labels for each image. The dataset selection and stratification process are displayed in a CONSORT-style diagram in Figure 2B. To account for human error in grading, a validation subset of 993 scans was graded separately by two ophthalmologist graders, with disagreement in clinical labels arbitrated by a senior retinal specialist.

The major challenge is classifying the images and grouping them into the datasets which was already ready for us which reduced the amount of work required for preprocessing.

Algorithms and Techniques

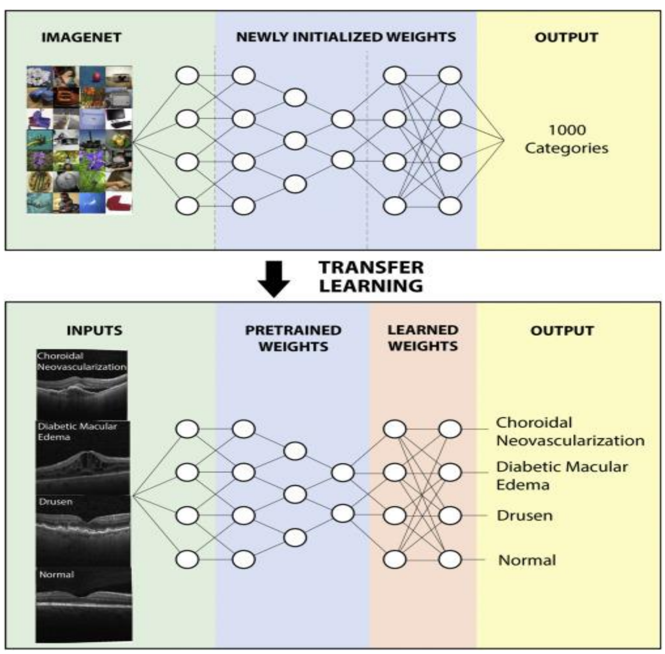
We initially build a Basic CNN and then switch to transfer learning with image augmentation and fine tuning.

A CNN can successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. This will be a very good fit for our problem. A typical CNN consists the following architecture.

Convolution Layer — The Kernel The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the **Kernel/Filter.** The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. CNN’s need not be limited to only one Convolutional Layer. Conventionally, the first Convolution Layer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, like how we would.

Pooling Layer- The Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model. There are two types of pooling which we will be using in architecture Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

Classification — Fully Connected Layer (FC Layer) This is where we convert our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model can distinguish between dominating and certain low-level features in images and classify them using the SoftMax Classification technique.



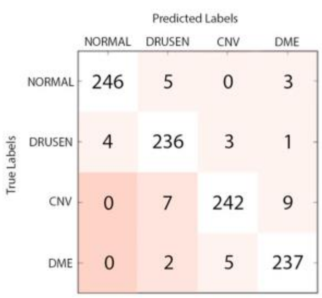
Schematic of a Convolutional Neural Network

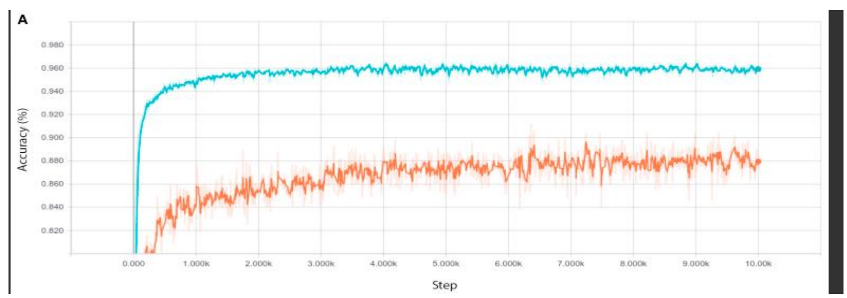
Now comes **Transfer learning** where pre-trained models such as VGG16, VGG19, Inception, ResNET-50 etc. are trained against ImageNet databases which consists of millions of images with optimized weights.

Rather than training a completely blank network, by using a feed-forward approach to fix the weights in the lower levels already optimized to recognize the structures found in images in general and retraining the weights of the upper levels with back propagation, the model can recognize the distinguishing features of a specific category of images, such as images of the eye, much faster and with significantly fewer training examples and less computational power.

Benchmark

We will compare our results against the Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning (https://www.cell.com/cell/fulltext/S0092-8674(18)30154-5) which has different metrics such as confusion matrix, accuracy, precision and recall. The below figures show metrics for the benchmark model





The accuracy for the model is 96.6%, from the confusion matrix we can compute the precession recall and f-1score for the Benchmark Model.

III. Methodology

Data Preprocessing

When using TensorFlow as backend, Keras CNNs require a 4D array (which we'll also refer to as a 4D tensor) as input, with shape (nb_samples, rows, columns, channels), where nb_samples correspond to the total number of images (or samples), and rows, columns, and channels correspond to the number of rows, columns, and channels for each image, respectively.

(nb_samples, rows, columns, channels),

The path_to_tensor function takes a string-valued file path to a color image as input and returns a 4D tensor suitable for supplying to a Keras CNN. The function first loads the image and resizes it to a square image that is pixels. Next, the image is converted to an array, which is then resized to a 4D tensor. In this case, since we are working with color images, each image has three channels. Likewise, since we are processing a single image (or sample), the returned tensor will always have shape

(1,150,150,3)

The paths_to_tensor function takes a numpy array of string-valued image paths as input and returns a 4D tensor with shape (nb_samples,150,150,3).

Here, nb_samples are the number of samples, or number of images, in the supplied array of image paths. It is best to think of nb_samples as the number of 3D tensors (where each 3D tensor corresponds to a different image) in your dataset!

We standardize the images by rescaling the images by dividing every pixel in every image by 255.

Next, we proceed with Image augmentation which artificially creates training images through different ways of processing or combination of multiple processing, such as random rotation, shifts, shear and flips, etc.

We used the following parameter for image augmentation, the train generators and val generators will be used during the fine-tuning our model.

```
from keras.preprocessing import image

train_datagen = image.ImageDataGenerator(zoom_range=0.3, rotation_range=50,
                                   width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2,
                                   horizontal_flip=True, fill_mode='nearest')

val_datagen = image.ImageDataGenerator()

train_generator = train_datagen.flow(train_tensors,train_targets, batch_size=30)

val_generator = val_datagen.flow(valid_tensors,valid_targets, batch_size=20)
```

Implementation

First, we implement a CNN with 3 convolution layers and 1 fully connected dense layer to predict the 4 different class labels. The below image shows the initial implementation

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 150, 150, 16)      208
_________________________________________________________________
batch_normalization_9 (Batch (None, 150, 150, 16)      64
_________________________________________________________________
activation_9 (Activation)    (None, 150, 150, 16)      0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 75, 75, 16)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 75, 75, 32)        2080
_________________________________________________________________
batch_normalization_10 (Batc (None, 75, 75, 32)        128
_________________________________________________________________
activation_10 (Activation)   (None, 75, 75, 32)        0
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 37, 37, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 37, 37, 64)        8256
_________________________________________________________________
batch_normalization_11 (Batc (None, 37, 37, 64)        256
_________________________________________________________________
activation_11 (Activation)   (None, 37, 37, 64)        0
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 18, 18, 64)        0
_________________________________________________________________
global_average_pooling2d_2 ( (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 260
_________________________________________________________________
batch_normalization_12 (Batc (None, 4)                 16
_________________________________________________________________
activation_12 (Activation)   (None, 4)                 0
=================================================================
Total params: 11,268
Trainable params: 11,036
Non-trainable params: 232
_________________________________________________________________
```

The model is then complied using rms prop as the optimizer, categorical cross entropy as loss function and accuracy as metrics.

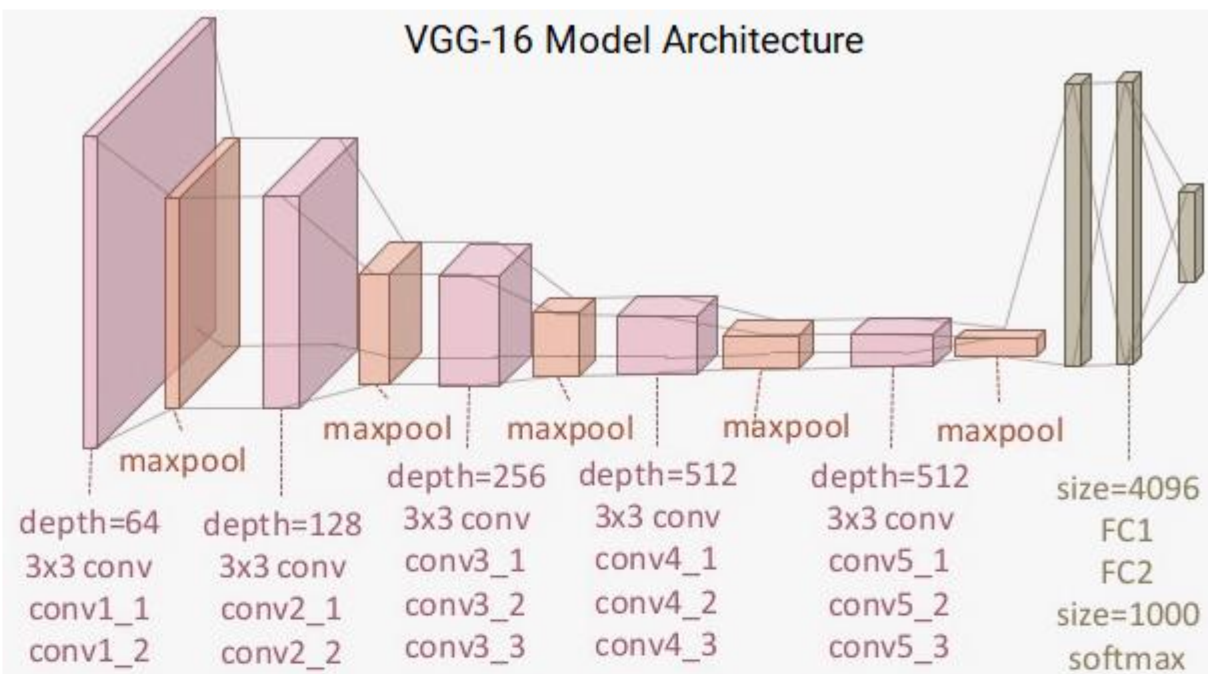
The model is then fit to the train sensor and validation tensor and ran for 4 epochs. We are using a check pointer which will the save the (weightsweights.best.from_scratch.hdf5) best weights that are produced by model in the back propagation method. The best weights that are captured in the previous step is used by the model.

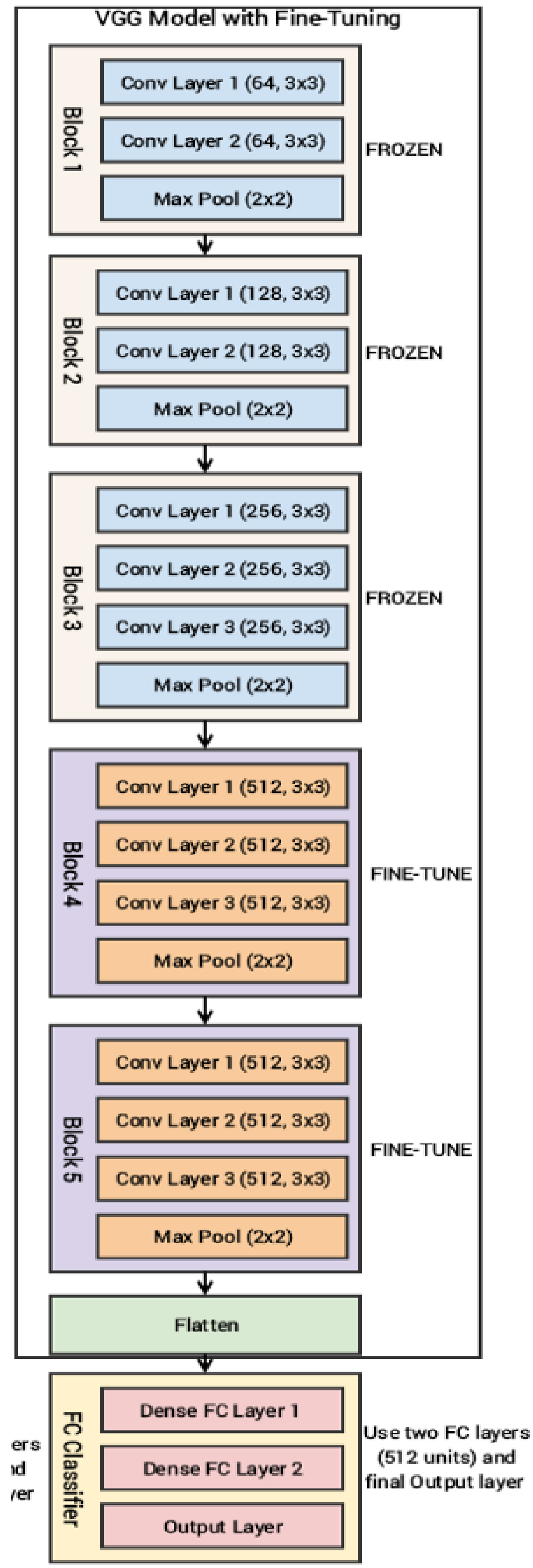
This model yielded an accuracy of 64.7727 % when tested against the test dataset, which is pretty good for initial configuration without any fine-tuning and image augmentation. Key Observations are shown below

```
Test accuracy: 64.7727%
Confusion Matrix
[[177  59   6   0]
 [  9 223   8   2]
 [ 14  53 173   2]
 [  0 123  65  54]]
Classification Report
              precision    recall  f1-score   support

         CNV       0.89      0.73      0.80       242
         DME       0.49      0.92      0.64       242
      DRUSEN       0.69      0.71      0.70       242
      NORMAL       0.93      0.22      0.36       242

   micro avg       0.65      0.65      0.65       968
   macro avg       0.75      0.65      0.62       968
weighted avg       0.75      0.65      0.62       968
```

Refinement

Since our initial model accuracy was not on par with the benchmark model. I implement ed Transfer learning algorithm with fine tuning and image augmentation. I used VGG16 Model and imported the model excluding the Top layer and freezing the first 3 convolution layers.




VGG Model with Fine-Tuning
Block 1
Conv Layer 1 (64, 3x3)
Conv Layer 2 (64, 3x3)
Max Pool (2x2)
FROZEN
Block 2
Conv Layer 1 (128, 3x3)
Conv Layer 2 (128, 3x3)
Max Pool (2x2)
FROZEN
Block 3
Conv Layer 1 (256, 3x3)
Conv Layer 2 (256, 3x3)
Conv Layer 3 (256, 3x3)
Max Pool (2x2)
FROZEN
Block 4
Conv Layer 1 (512, 3x3)
Conv Layer 2 (512, 3x3)
Conv Layer 3 (512, 3x3)
Max Pool (2x2)
FINE-TUNE
Block 5
Conv Layer 1 (512, 3x3)
Conv Layer 2 (512, 3x3)
Conv Layer 3 (512, 3x3)
Max Pool (2x2)
FINE-TUNE
Flatten
FC Classifier
Dense FC Layer 1
Dense FC Layer 2
Output Layer
Use two FC layers
(512 units) and
final Output layer

	Layer Type	Layer Name	Layer Trainable
0	<keras.engine.input_layer.InputLayer object at...	input_1	False
1	<keras.layers.convolutional.Conv2D object at 0...	block1_conv1	False
2	<keras.layers.convolutional.Conv2D object at 0...	block1_conv2	False
3	<keras.layers.pooling.MaxPooling2D object at 0...	block1_pool	False
4	<keras.layers.convolutional.Conv2D object at 0...	block2_conv1	False
5	<keras.layers.convolutional.Conv2D object at 0...	block2_conv2	False
6	<keras.layers.pooling.MaxPooling2D object at 0...	block2_pool	False
7	<keras.layers.convolutional.Conv2D object at 0...	block3_conv1	False
8	<keras.layers.convolutional.Conv2D object at 0...	block3_conv2	False
9	<keras.layers.convolutional.Conv2D object at 0...	block3_conv3	False
10	<keras.layers.pooling.MaxPooling2D object at 0...	block3_pool	False
11	<keras.layers.convolutional.Conv2D object at 0...	block4_conv1	True
12	<keras.layers.convolutional.Conv2D object at 0...	block4_conv2	True
13	<keras.layers.convolutional.Conv2D object at 0...	block4_conv3	True
14	<keras.layers.pooling.MaxPooling2D object at 0...	block4_pool	True
15	<keras.layers.convolutional.Conv2D object at 0...	block5_conv1	True
16	<keras.layers.convolutional.Conv2D object at 0...	block5_conv2	True
17	<keras.layers.convolutional.Conv2D object at 0...	block5_conv3	True
18	<keras.layers.pooling.MaxPooling2D object at 0...	block5_pool	True
19	<keras.layers.core.Flatten object at 0x0000016...	flatten_1	True

Now we will use this VGG_model as the input for our model and which is first passed thorough a fully connected layer of 512 modes and finally a fully connected layer of 4 nodes which will classify our class labels.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
model_1 (Model)              (None, 8192)              14714688
_________________________________________________________________
dense_1 (Dense)              (None, 512)               4194816
_________________________________________________________________
batch_normalization_1 (Batch (None, 512)               2048
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 2052
_________________________________________________________________
batch_normalization_2 (Batch (None, 4)                 16
_________________________________________________________________
activation_2 (Activation)    (None, 4)                 0
=================================================================
Total params: 18,913,620
Trainable params: 17,177,100
Non-trainable params: 1,736,520
_________________________________________________________________
```

Next, we compile the model with binary cross entropy, rms prop with a low learning rate of (1e-5) and accuracy as the metric. The model is fit and trained for 15 epochs where we will get best weights weights.best.from_transfer.hdf5 for the trained model. This model yielded an accuracy of 97.84% when tested against the test dataset, this is better than the benchmark model.

Key Observations are shown below

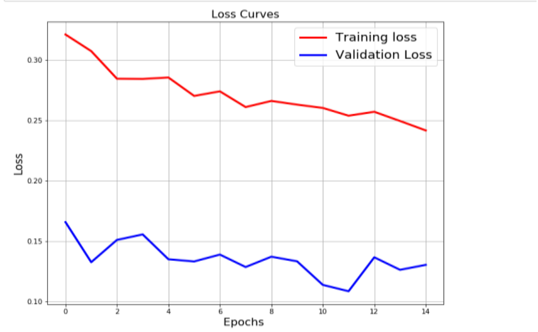
```
Test accuraccuracy_score for Transfer Learning Model: 97.83057851 23967
Confusion Matrix
[[241   1   0   0]
 [  1 235   0   6]
 [  9   2 229   2]
 [  0   0   0 242]]
Classification Report
              precision    recall  f1-score   support

         CNV       0.96      1.00      0.98       242
         DME       0.99      0.97      0.98       242
      DRUSEN       1.00      0.95      0.97       242
      NORMAL       0.97      1.00      0.98       242

   micro avg       0.98      0.98      0.98       968
   macro avg       0.98      0.98      0.98       968
weighted avg       0.98      0.98      0.98       968
```

IV. Results

Model Evaluation and Validation

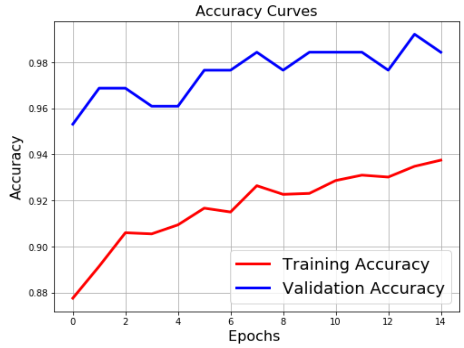
Loss Curves

The below graph shows the training loss and validation loss for 15 epochs where they are closing in, if we increase the number of epochs we can still decrease the validation loss.



Accuracy Curves

The model is performing well on the validation set and training set where the accuracy is close, we can conclude the model is not overfitting.



Test Data

I tested the model with test data which consists of 968 images, it yielded an **accuracy of 97.84%.**

Reading the Confusion matrix

CNV: True Positives (TP) (correct predictions) = 241, False Positives (FP) =10 (1 is DME and 9 are Drusen which are predicted as CNV) and False Negatives (FN) = 1 (it needs to predict CNV, but it predicted as DME) which yields precession of 0.96 and recall of 1.0

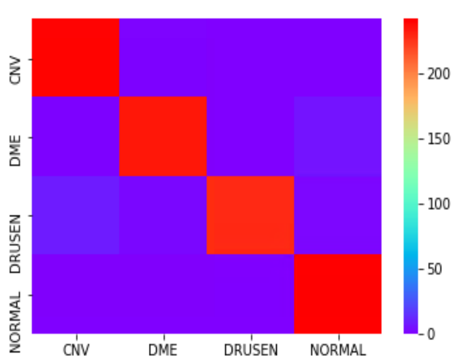
DME: TP = 235, FP=3 (1 is CNV and 2 are Drusen which are predicted as DME) and FN = 7 (1 as CNV, 6 as Normal) which yields precession of 0.99 and recall of 0.97

DRUSEN: TP= 229, FP=0 and FN=13 (9 as CNV, 2 as DME and 2 as NORMAL) which yields precession of 1.0 and recall of 0.95

NORMAL: TP =242, FP= 8 (2 as DRUSEN and 6 as DME) and FN=0 which yields precession of 0.97 and recall of 1.0

```
Test accuraccuracy_score for Transfer Learning Model: 97.8305785123967
Confusion Matrix
[[241   1   0   0]
 [  1 235   0   6]
 [  9   2 229   2]
 [  0   0   0 242]]
Classification Report
              precision    recall  f1-score   support

         CNV       0.96      1.00      0.98       242
         DME       0.99      0.97      0.98       242
      DRUSEN       1.00      0.95      0.97       242
      NORMAL       0.97      1.00      0.98       242

   micro avg       0.98      0.98      0.98       968
   macro avg       0.98      0.98      0.98       968
weighted avg       0.98      0.98      0.98       968
```

Heatmap for the Confusion Matrix



Justification

The predicted model is performing better than the benchmark model. Comparing the accuracy as metric

Accuracy for Benchmark: 93.4%

Accuracy for Predicted Model: 97.84%

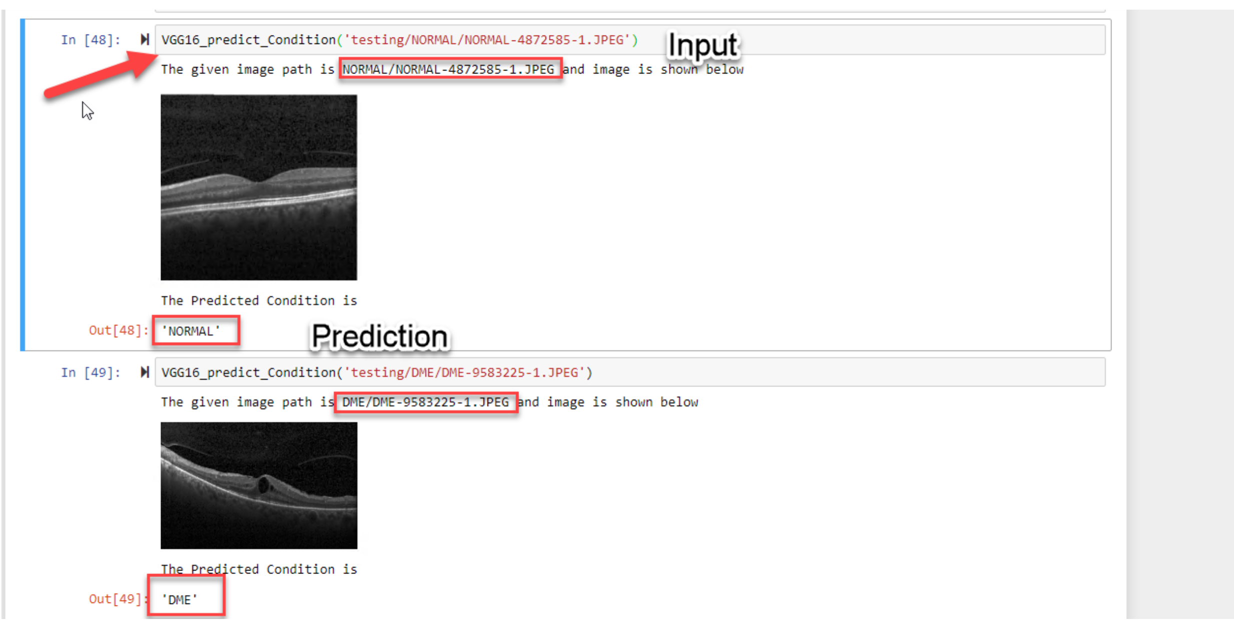
We improved the model with fine-tuning and image augmentation.

V. Conclusion

Free-Form Visualization

I used the model to define a function that will output the class when we input the image path. Here

VGG16_predict_Condition is function that takes the Image path as input and then predicts the class label



Reflection

The implementation of clinical-decision support algorithms for medical imaging faces challenges with reliability and interpretability. Here, we establish a diagnostic tool based on a deep-learning framework for the screening of patients with common treatable blinding retinal diseases. Our framework utilizes transfer learning, which trains a neural network with a fraction of the data of conventional approaches using Image Net database. Applying this approach to a dataset of optical coherence tomography images, we demonstrate performance comparable to that of human experts in classifying age-related macular degeneration and diabetic macular edema.

Here is the list of steps:

1. Import the data consisting of training, validation and test sets.
2. Pre-Process and normalize the data to be fed as input to keras.
3. Data Augmentation.
4. Creating a Model using Convolution Neural Network (Basic CNN)
5. Implementing Transfer Learning with VGG16 Model and augmented data.
6. Evaluating the metrics against the test data set.

The most interesting part was the way transfer learning boosted the accuracy to 97.84% which can do wonders to predict in the clinical decisions.

The preprocessing the data and image augmentation were challenging but running the model multiple times which took long hours to complete was one of the major challenges as the data set was large.

After testing it on different images, I was really satisfied with the outcome of the model.

Improvement

I would like to use different transfer learning model such as ResNet50, Inception and compare the accuracies of the predicted model. Similarly, I would like to try different optimization parameters to compile the model, increase the number of epochs and test the prediction of the model.

Initially I considered using Inception-ResNet Model but thought it might be resource intensive to perform transfer learning on such a huge dataset.

Yes, there might be model that can be achieved with optimizing different parameters and transfer learning algorithms, but it might be resource intensive and increase the overall operating cost.

Citations

1404033991. “A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning.” *Towards Data Science*, Towards Data Science, 14 Nov. 2018, towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a.

Mooney, Paul. “Retinal OCT Images (Optical Coherence Tomography).” *Kaggle*, 25 Mar. 2018, www.kaggle.com/paultimothymooney/kermany2018.

“Optical Coherence Tomography.” *EyeWiki*, 20 Jan. 2015, eyewiki.aao.org/Optical_Coherence_Tomography.

“The Keras Blog.” *The Keras Blog ATOM*, blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html.