Problem Statement

To develop a model that will classify severe levels of diabetic retinopathy based on eye images with both accuracy and recall of at least 80%.

Background

Diabetic retinopathy can lead to complications like loss of vision and even blindness. It would be useful to have an additional tool to identify this disease and to what stage it has progressed to be able to determine treatment and perhaps even reversal, when caught early enough.

Dataset

The full dataset used contains images from the 2015 Diabetic Retinopathy Detection and APTOS 2019 Blindness Detection competitions. Each image has been resized and cropped to have a maximum size of 1024px.

For the 2015 images, 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). The 2019 images do not specify a subject or eye.

A clinician has rated each image for the severity of diabetic retinopathy on a scale of 0 to 4: 0 – No DR, 1-Mild, 2-Moderate, 3-Severe, 4-Proliferative DR.

The images were gathered from multiple clinics using a variety of cameras over an extended period of time and are of varying quality.

Data Link

https://www.kaggle.com/benjaminwarner/resized-2015-2019-blindness-detection-images

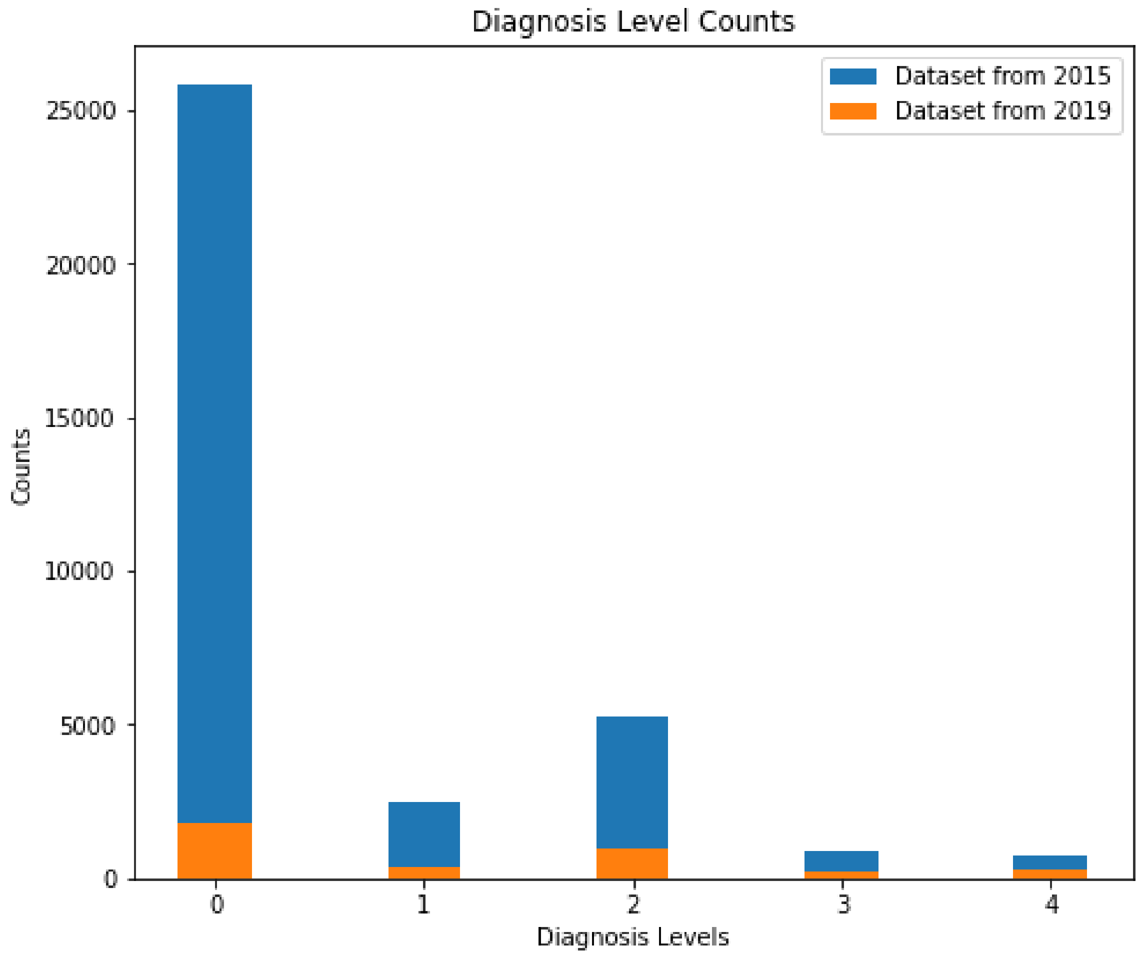
Dataset Counts

Data set from 2015 “train” folder diagnosis level counts

Diagnosis Level	Count
0	25810
1	2443
2	5292
3	873
4	708

Dataset from 2019 “train” folder

Diagnosis Level	Count
0	1805
1	370
2	999
3	193
4	295

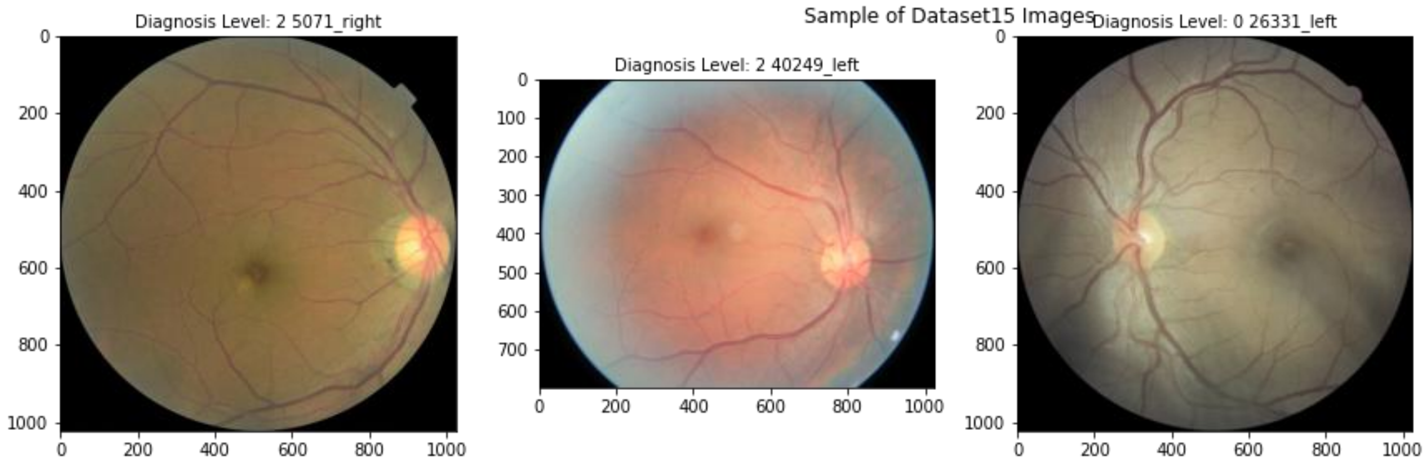


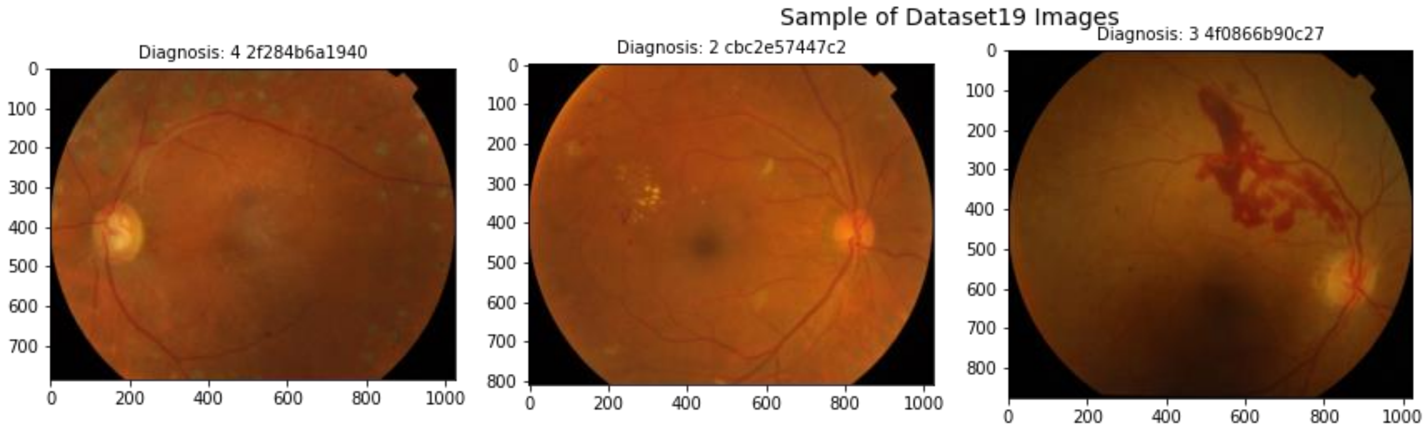
Dataset from 2015 “test” folder

Diagnosis Level	Count
0	39533
1	3762
2	7861
3	1214
4	1206

The dataset from the 2019 “test” folder had no labels and were not used.

Dataset Images


Sample of Dataset15 Images
Diagnosis Level: 2 5071_right
Diagnosis Level: 2 40249_left
Diagnosis Level: 0 26331_left


Sample of Dataset19 Images
Diagnosis: 4 2f284b6a1940
Diagnosis: 2 cbc2e57447c2
Diagnosis: 3 4f0866b90c27

Basis

Model Type

In looking into what models to work with when dealing with images, I decided to use Convolutional Neural Networks, or CNNs. This is because this model was designed to map image data to an output variable. They have proven so effective that they are the go-to method for any type of prediction problem involving image data as an input.

More specifically, I decided to work with a Residual Network - a type of CNN designed to work better for deeper learning than regular networks. Deeper learning (increased layers) tends to enhance model. In a regular network, the benefit of increasing layers may get degraded for various reasons and a residual network is designed to mitigate this degradation.

Furthermore, I decided to use the fastai library and the "resnet family". Resnet-18, Resnet-34, Resnet-50, Resnet-101 and Resnet-152 are all residual networks pretrained on the ImageNet dataset, which contains over 14 million images. The number indicates number of layers. For this project, Resnet-34 and Resnet-152 were experimented with a little, Resnet-18 and Resnet-101 were not used at all. Resnet-50 was used for the bulk of the experiments.

Model testing

The datasets were consisted of four directories worth of training and testing image data from each of 2015 and 2019. All the data had labels indicated in a separate .csv file except for the testing data from 2019.

All the models were trained with a 80/20 training/validation split.

Images sizes (1024 to match size of images vs 128 to allow processing to experimenting with 256 until eventually setting on 224 when discovering that that size matched with the images the resnet was pretrained on.

Approximately 30 tests were conducted, varying:

Datasets of various sizes and configurations with a sampling of:

2015 data, 2019 data or a combination

Using all 5 diagnosis levels vs trinary (two different ways) vs binary

Resnet-34 vs Resnet-50 vs Resnet-152

Batch size 10 or 16 or 32 or 64

Number of epochs (5,7,8,10,15) with and without refining epochs run with learning rate based on the learn rate finder

Transforms: generated or

generated with addition of image size 128 vs 224 vs 256 vs 1024

Basic Testing Progression

1) Small subsets of 2015 data, trying to be able to work to complete on Colab platform, using 1024 image size, with models preforming very badly.

2) Trying smaller image sizes, settling on 128, trying various data distributions.
3) Getting more success when reducing number of 0 level diagnosis and upping level 4 diagnosis levels as much as possible (which could be done because they were proportionally lower).
4) Models finally start to reach past 80% accuracy on test7 using all the level 4s possible within the 2015 dataset, and using about the same number of 3s, along with lower numbers of 1s and 2s and a number of 0s that were about the same as all the 1s, 2s, 3s and 4s put together.
5) Model was tried with trinary classification and got less accuracy, but improved into high 80 accuracy with binary classification.
6) Started to use 2019 data along with the 2015 data and got mid-80s accuracy.
7) Then noticed 2019 data by itself had similar proportions of diagnosis levels as did well in models and also was on the order of the total amount of data being used in other models that did well, so developed model on all of 2019 training data. That model (from test 13) got accuracy and recall in the high 90s.
8) The excitement about that did not last as tried that 2019-based model on 2015 test data (only those that had a level of 1 or more) and it did horribly. However, among those horrible numbers, diagnosis level 4 had the clear best recall.
9) Since that model appeared to be overfitted, the test 11 model was brought back in to try as it had been trained on both 2015 and 2019 data. It was also run on non-zero level 2019 test data. The level 1 and level 2 recall percentages were not good. However, the level 3 and level 4 recall percentages were 90% and 97%, respectively. There was some further experimentation with using a trinary pattern that put 0s alone, 1s and 2s together and 3s and 4s together, among other things, but no model did better. Thus, this was chosen as the final model.

Final Model

Dataset: each value below is "diagnosis level:count" (e.g 0:1500 indicates there was a random sample count of 1500 of diagnosis level 0 images chosen)

from 2015 dataset [0:1500, 1:300, 2:300, 3:700, 4:700]
from 2019 dataset [0:340, 1:80, 2:80, 3:190, 4:190]
Converted to binary diagnosis levels thus levels 1,2,3 set to 4
Final dataset [0:1840, 4:2540]

Transform: Generated and resizing (224) to fit with pretrained data
Batch size: 16
Model: Resnet-50
Number of epochs: 10, with refinement on additional two based with assigned learning rate

Validation Metrics

Accuracy 0.86
Recall 0.82

Testing on 2000 Test Images from 2015 Dataset
(Diagnosis Levels 1-4 only)

Diagnosis Level	Recall
1	0.20
2	0.60
3	0.90
4	0.97

Discoveries

On the test data, the chosen model detected (showed as positive) level 3 and 4 diabetic retinopathy very well, although it did not do well detecting levels 1 and 2. Thus, even its current form, the model could possibly be useful with an understanding that results would be very dependable for detecting the most severe cases, but not for the low severity cases. Thus, negative results would mean another kind of test would be warranted but the good news is that any false negative would probably be a low-level case, so the most severe cases would likely be caught. In other words, it could be used as tool to find severe cases of diabetic retinopathy very accurately without a technician from an eye image that is produced with a methodology consistent with the 2015 or 2019 datasets.

With more detail on how the images were produced, perhaps the model could be restricted to only be used on certain input (maybe images produced by certain machines or prepared in whatever way the datasets were prepared from the source). With those kind of parameters, the model that did well on only 2019 images could perhaps even come back to play with the understanding that it could only be used for restricted types of eye images.

Ideas to Possibly Improve Model

1) Experiment with more data
2) Experiment with black-and-white images
3) Experiment more with parameters discussed, as well as other parameters available
4) Experiment with more varied data – unless have process whereby can guarantee a certain image quality and then train intensely on that quality of image
5) If the specific machine/image quality standard could be found, this could be a tool used for that specific setup and further trained for it specifically

Ideas for Future

1) Set up process, as automated as possible for monitoring accuracy, deciding on images to submit for retraining and to do the retraining.
2) Create an additional independent model that concentrates on detecting less severe cases to run as a follow up to run on image that return come back negative on first model.

Conclusion

This model could be tried out, with monitoring, as a first-pass detector for severe diabetic retinopathy. A negative test would mean that there would need to be additional testing, especially to possibly detect an earlier stage of the disease. However, a positive test would be a red flag that there is a likely a severe issue and it could be handled as such.

In addition, the model should be monitored based on cross-checking results and using that information to retrain and improve the model, possibly along with the other improvements and future considerations as stated earlier.