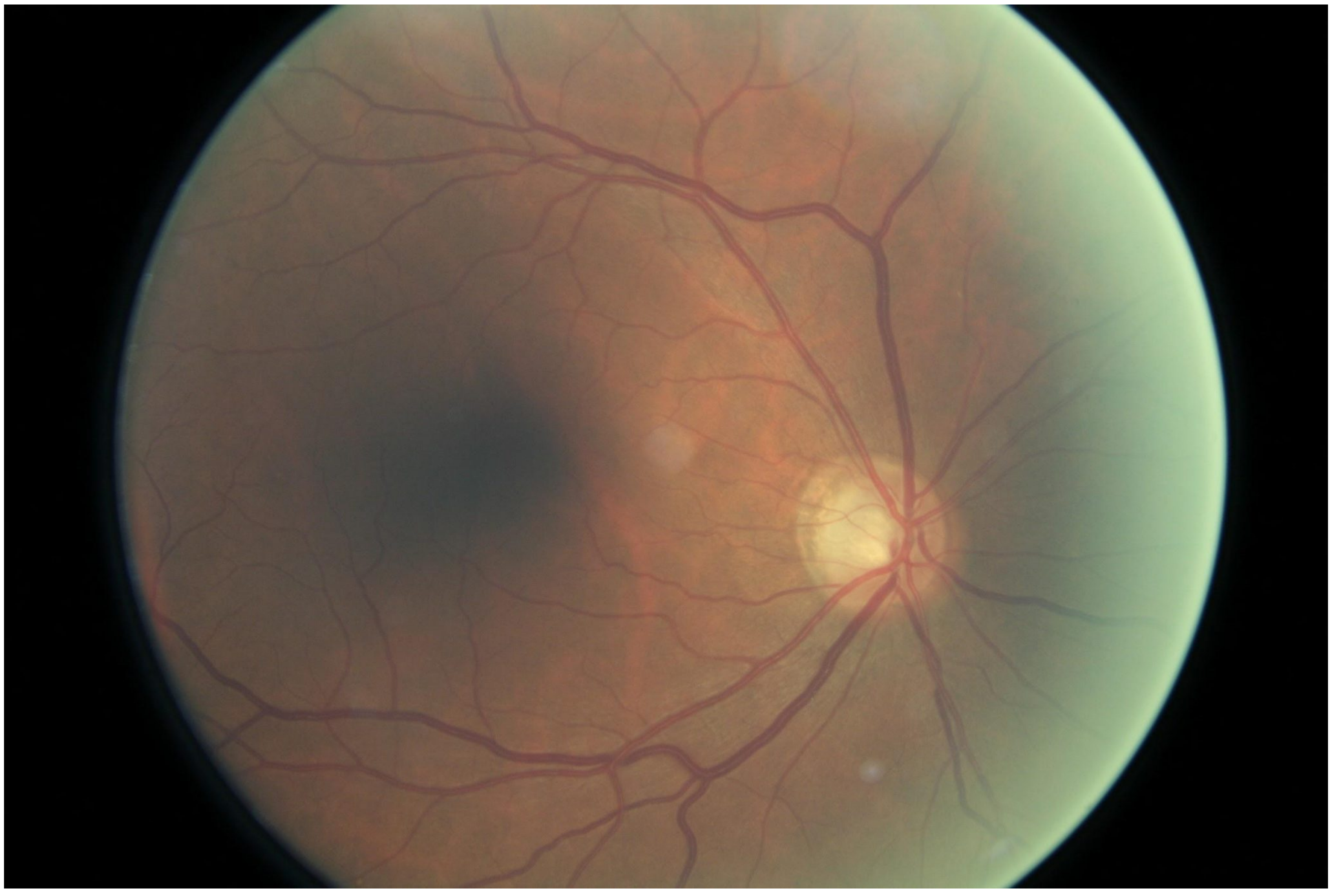
M E D I C I N E (H T T P S : / / Y S J O U R N A L . C O M / C A T E G O R Y / M E D I C I N E /)

Study of Convolutional Neural Networks for Early Detection of Diabetic Retinopathy

POSTED APRIL 9, 2020 (HTTPS://YSJOURNAL.COM/STUDY-OF-CONVOLUTIONAL-NEURAL-NETWORKS-FOR-EARLY-DETECTION-OF-DIABETIC-RETINOPATHY/) RACHEL CAI (HTTPS://YSJOURNAL.COM/AUTHOR/SHIVANSHITANDON/)



Abstract

Diabetic retinopathy (DR) is the leading cause of blindness in the working-age population of the developed world. Presently, detecting DR is a manual, time-consuming process that requires a trained ophthalmologist to examine and evaluate digital fundus photographs of the retina. Computer machine learning technologies such as Convolutional Neural Networks (CNNs) have emerged as an effective tool in medical image analysis for the detection and classification of DR in real-time.

In our study, I adapt several CNNs (Inception, VGG16, and Resnet) for the classification of DR stages, then compare the effectiveness of the above networks. We design experiments to evaluate the effect of different parameters (different training sample sizes and different image sizes) on training. In addition, we optimize the network with features such as attention maps and experiment with different image preprocessing.

Our study shows that all three CNNs have reasonable performances, with VGG16 slightly ahead. Optimizing techniques such as attention maps can also help. However, 5-stage DR classification is still a hard problem, especially when distinguishing between the middle stages of DR. We believe CNN will be adopted for automatic DR detection in the near future.

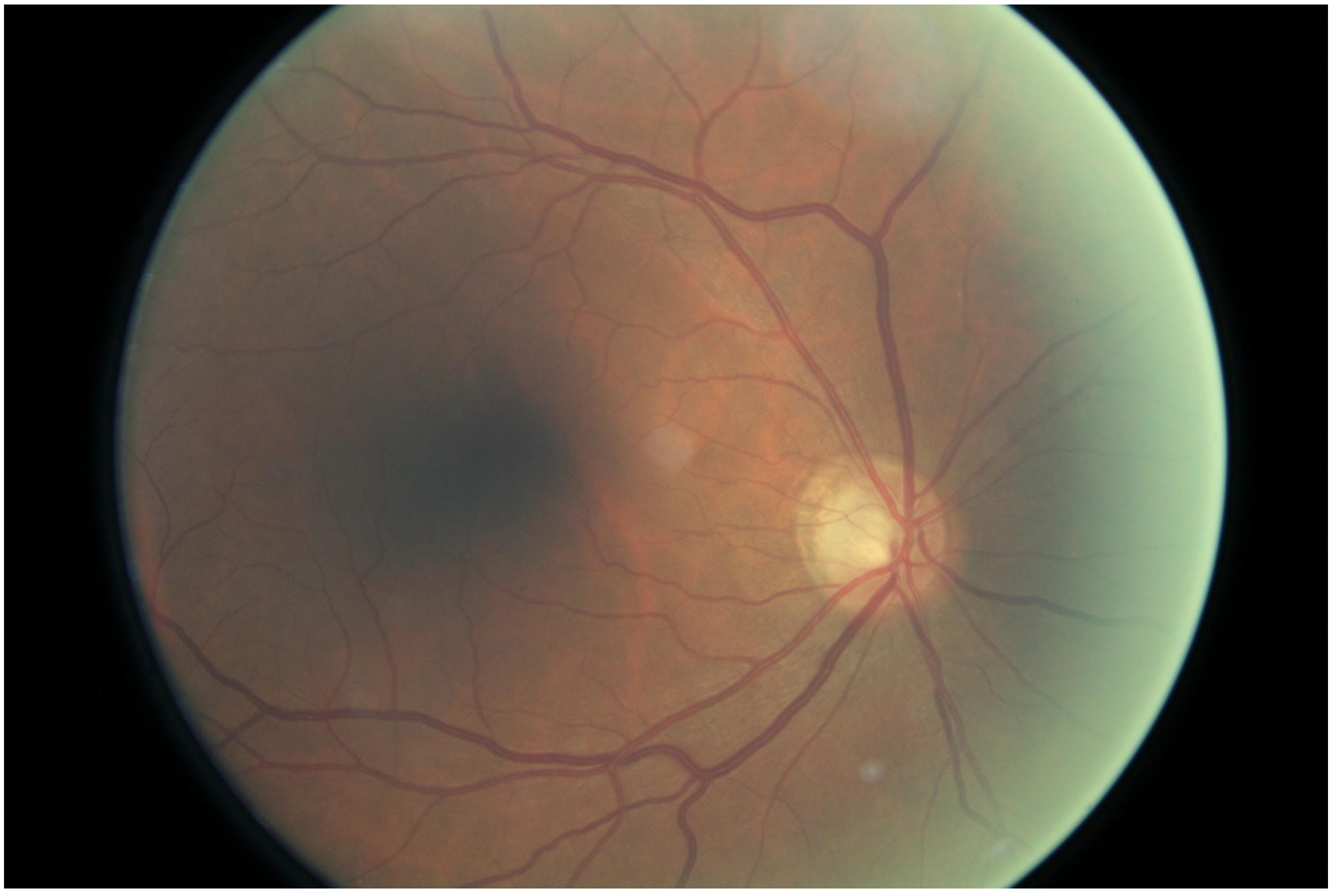
Keywords: Deep learning, Convolutional Neural Networks, Diabetic Retinopathy, Image Classification, Diabetes, Inception Network, VGG16 Network, ResNet

Introduction

During the summer of 2018 and the following school year, I had the opportunity to intern at the National Institutes of Health (NIH), National Library of Medicine (NLM), Lister Hill Center to study Convolutional Neural Networks for the early detection of diabetic retinopathy, an eye disease. In our study, I adapt several CNNs (Inception, VGG16, and Resnet) for the classification of DR stages. I compare the effectiveness of the above networks. We also design experiments to evaluate the effect of different parameters (different training sample sizes and different image sizes) on training. I also optimize the network with features such as attention maps and experiment with different image preprocessing.

Diabetic Retinopathy (DR)

Diabetic retinopathy is the leading cause of blindness in the working-age population of the developed world. It is estimated to affect over 93 million people.



Around 40-45% of Americans with diabetes have some stage of the disease [16]. Progression to vision impairment can be slowed or averted if DR is detected in time, which can be difficult as the disease often shows few symptoms until it is too late to provide effective treatment.

Presently, detecting DR is a manual, time-consuming process that requires a trained clinician to examine and evaluate digital fundus photographs of the retina. By the time human readers submit their reviews, sometimes several days later, the delayed results can lead to lost communication, mix-ups, little follow-up, and delayed treatment.

Computer Vision through Convolutional Neural Network

Convolutional Neural Networks (CNNs), a branch of deep learning, have a great record for application in image analysis and interpretation, including medical imaging. However, it wasn’t until several breakthroughs in neural networks, such as the implementation of dropout, rectified linear units and the accompanying increase in computing power through graphical processing units (GPUs) that CNNs became viable for more complex image recognition problems. Presently, large CNNs are used to successfully tackle highly complex computer vision tasks with many object classes to an impressive standard. CNNs are used in many current state-of-the-art image classification tasks such as the annual ImageNet challenges.

CNN for Diabetic Retinopathy detection

Convolutional Neural Network is a feed-forward neural network. It mainly consists of an input layer, many hidden layers (such as convolutional relu, pooling, flatten, fully connected and softmax layers) and a final multi-label classification layer. CNN methodology involves two stages of processing: a time-consuming training stage where millions of images go through many iterations of CNN architecture to finalize the model parameters of each layer and a second real-time prediction stage where each image in test dataset is fed into the trained model to score and validate the model.

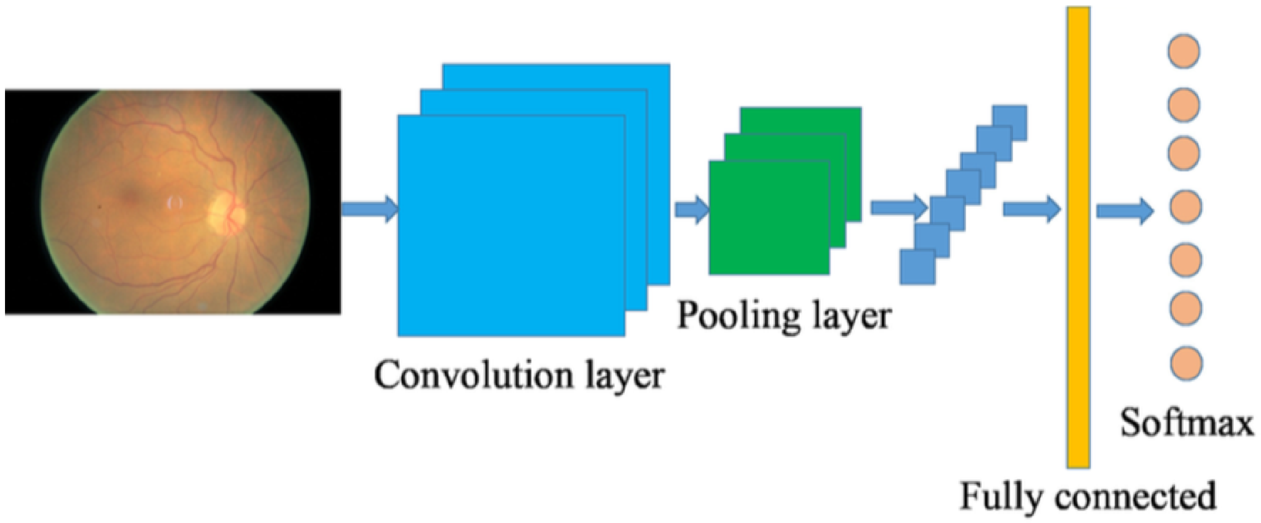


Figure 1: CNN for DR Detection

The output of the above framework as shown in Figure-1 will emit a multi-class prediction for the likelihood that the image is in that class, with confidence scores on each category such as:

- 52% Category-0 No DR (Normal)
- 17% Category-1 DR
- 27% Category-2 DR
- 2.5% Category-3 DR
- 0.8% Category-4 DR

However, there are two issues with CNN methods for DR detection. One is achieving a desirable offset in sensitivity (patients correctly identified as having DR) and specificity (patients correctly identified as not having DR). This is significantly harder for a five-class problem containing normal, mild, moderate, severe, and proliferative DR classes. The second problem is overfitting. Skewed datasets cause the network to over-fit to the class most prominent in the dataset. Large datasets are often massively skewed.

Our Work

We explore the use of Deep Convolutional Neural Network methodology for the automatic classification of diabetic retinopathy using color fundus images. I evaluate several CNN architectures, design various experiments, and study several tuning techniques for model training.

Methodology and Training Platform

In this project, I adapt several CNN architectures (Inception Network, VGG16, and Microsoft ResNet) for DR and evaluate their performance on DR image classification. I apply various techniques to cleanse and augment the data, and also optimize the CNN to accommodate the skewed data sets.

Our experiment is conducted on the hosted Linux platform with NVidia Tesla K80 GPU. The environment is hosted by Google Colab and Kaggle Kernel. The implementation is based on Keras/TensorFlow framework. For most of the trainings, we ran 25 epochs with a short circuit abort if there is no improvement after 6 epochs.

Terminology

Most of the tests measure two key aspects of the training model for statistical evaluation:

- **Accuracy** measures true positive and true negative rates.
  - **Binary Accuracy** Whether the model can accurately predict an image is DR
  - **Categorical Accuracy** Which category/stage of DR does the image belong to.
  - In our study, we will be using categorical accuracy most of the time since this is a more meaningful validation. Categorical accuracy is lower than Binary accuracy.
- **AUC** is an abbreviation for the area under the curve. AUC is a more holistic measurement considering both true negative and true positive. The closer AUC is to 1,

the better the model performance.

Other terminologies used in the report are from the standard statistics domain:

- **true positive**: images with DR classified as with DR
- **true negative**: images without DR classified as without DR
- **false positive**: images without DR classified as with DR
- **false negative**: images with DR classified as without DR
- **precision**: positive predictive value, the number of true positives divided by the sum of total true positives and false positives
- **recall** or **sensitivity**: true positive rate, the number of true positives divided by the sum of total true positives and false negatives
- **specificity**: true negative rate, the number of true negatives divided by the sum of total true negatives and false positives
- The heatmap measures the distribution of precision/recall on each of the DR category
- **micro average**: takes the precision or recall for the results of every class
- **macro average**: takes the precision or recall for each class, then averages it
- **weighted average**: takes the sum of the precision or recall for each class, but weighted with the number of samples in each class

Note: if the sample size is not explicitly mentioned, it consists of 1,000 samples, 750 images for training and 250 for testing/validation. The samples are always split 3 to 1, training set to testing set. For example, in a 1,000 image data set, 750 images are used for training and 250 images for testing. The test set and the validation set are the same in our experiment.

Datasets

Kaggle DR competition dataset

Our main dataset is based on the [Kaggle Diatebic Retinopathy Detection competition](https://www.kaggle.com/c/diabetic-retinopathy-detection) which was carried out in 2016. The main dataset contains 35,000 eye images with 5 stages of DR disease.

Messidor dataset

We also look at the Messidor dataset which contains 1,200 images with 4 stages of DR progression. Although the Messidor dataset is smaller, there are fewer labeling errors.

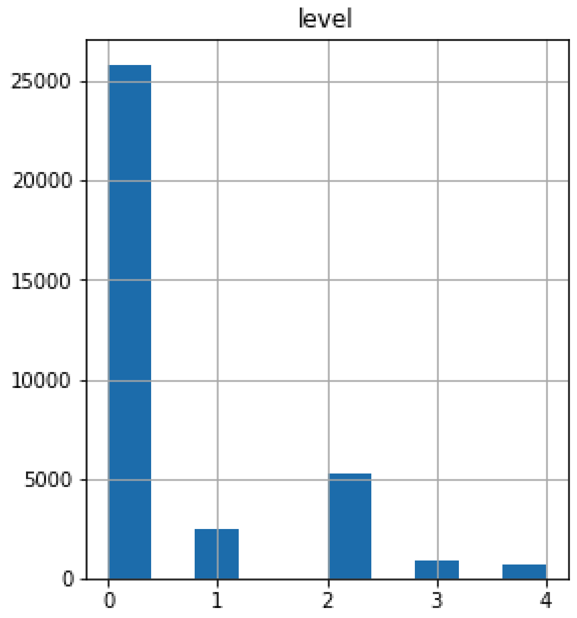
Stages of diabetic retinopathy (DR) with increasing severity

Figure 2 shows the 5 class DR classification, ranging from 0 (No DR) to 5 (Proliferative DR).

Figure 2: Different Stages of Diabetic Retinopathy

Unbalanced training data set

Figure 3: Training Data Distribution between 5 stages of DR



X-axis: category of DR, Y-axis: number of training samples

Skewed datasets cause the network to over-fit to the class most prominent in the dataset. Large datasets are often massively skewed. Figure 3 shows how the training data are distributed in our DR datasets.

In the Kaggle dataset with 35,000 images, less than three percent of images came from the 4th and 5th class. This means changes had to be made in our network to ensure it could still learn the features of these images, without enough training. To overcome the difference in data points distribution, I use sampling with replacement statistics technique to boost the data samples in category 2, 4 and 5:

Figure 4 shows a balanced distribution after using replacing techniques. The graph on the left displays evenly distributed samples for the left eye and the right eye. Both have over 12,000 images to test and train. The graph on the right displays evenly distributed samples from each DR level. Every level has 5,000 images to test and train, which is much less skewed to favor training and testing for category 0.

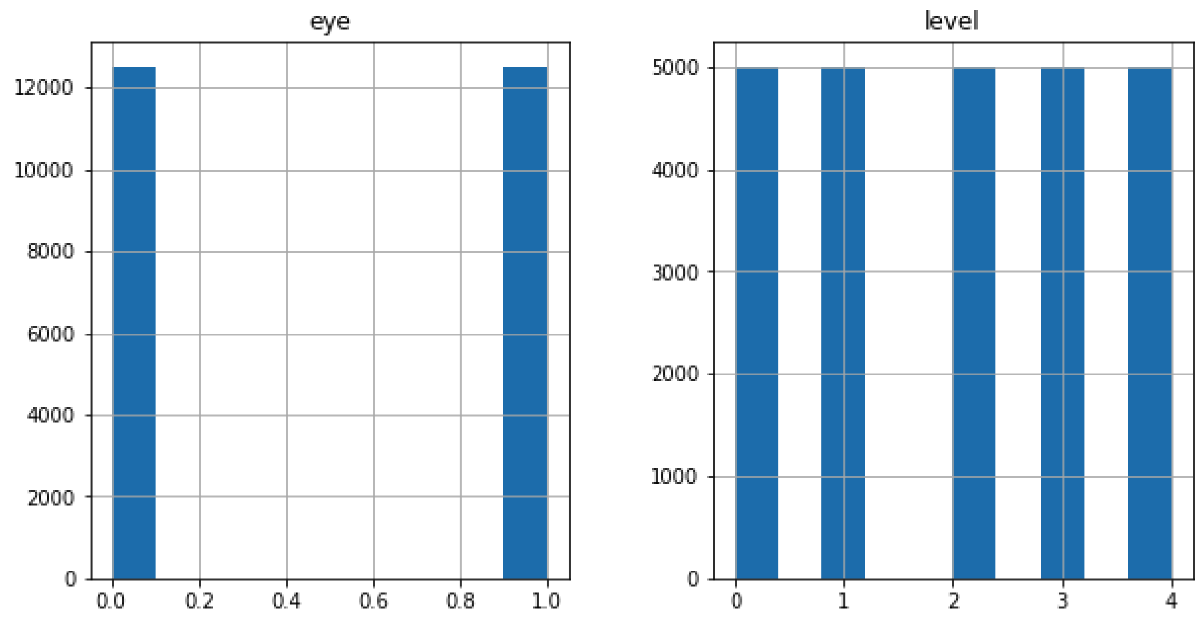


Figure 4: Rebalancing training data between eyes and DR stages

CNN Architectures

There are various CNN architectures proposed in academia: VGG16, Inception, ResNet, GoogLeNet. This pyImageSearch article has a good introduction to many popular CNN architectures. In our study, we evaluated the performance of InceptionV3 vs. VGG16 vs. ResNet

InceptionV3

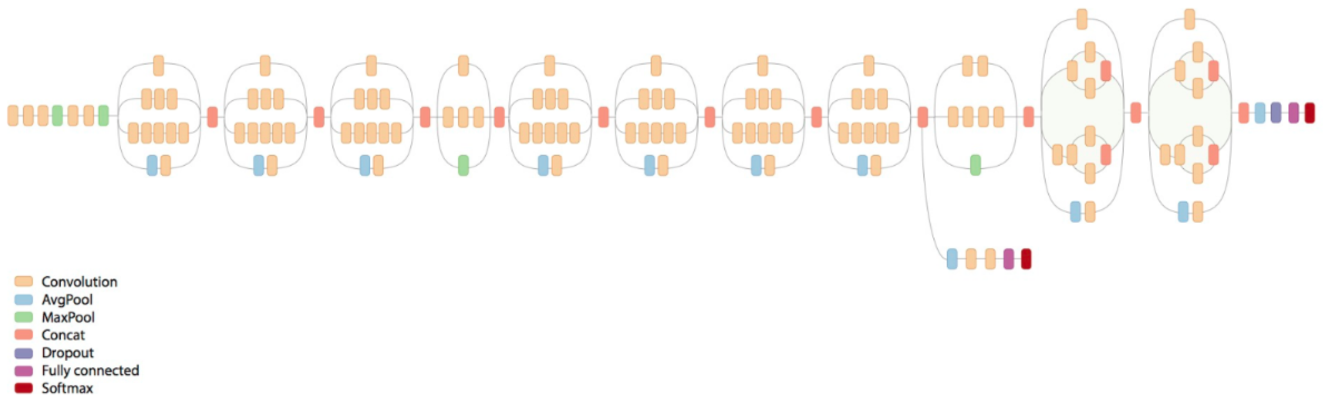


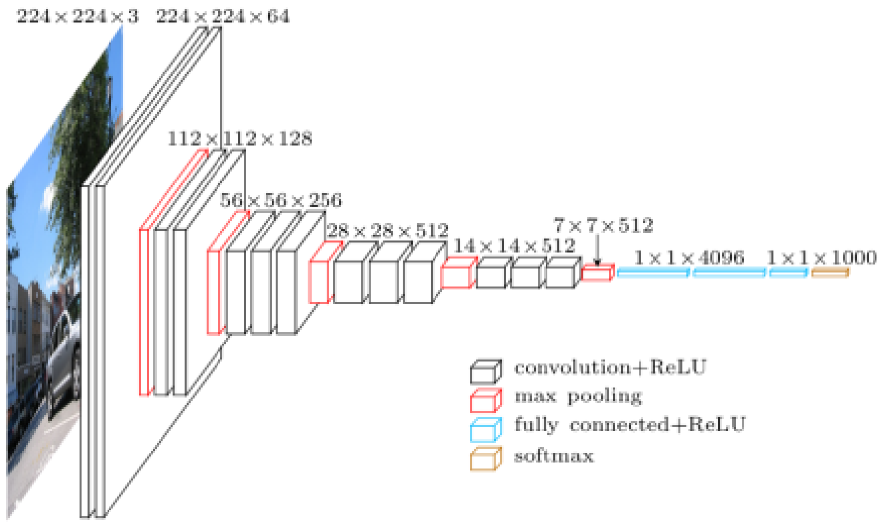
Figure 5: Inception Network Architecture

The Inception microarchitecture was first introduced by Szegedy et al. in their 2014 paper Going Deeper with Convolutions. The goal of the inception module is to act as a “multi-level feature extractor” by computing 1×1, 3×3 and 5×5 convolutions within the same module of the network. The original incarnation of this architecture was called GoogLeNet, but subsequent manifestations have simply been called Inception vN, where N is the version number put out by Google.

Our implementation of InceptionV3 was mostly inspired by the work by Kevin Mader on Kaggle

VGG16 and VGG19

Figure 6: VGG Network Architecture



The VGG network architecture was introduced by Simonyan and Zisserman in their 2014 paper: Very Deep Convolutional Networks for Large Scale Image Recognition. The VGG network is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4096 nodes are then followed by a softmax classifier (above). The “16” and “19” stands for the number of weight layers in the network.

ResNet50

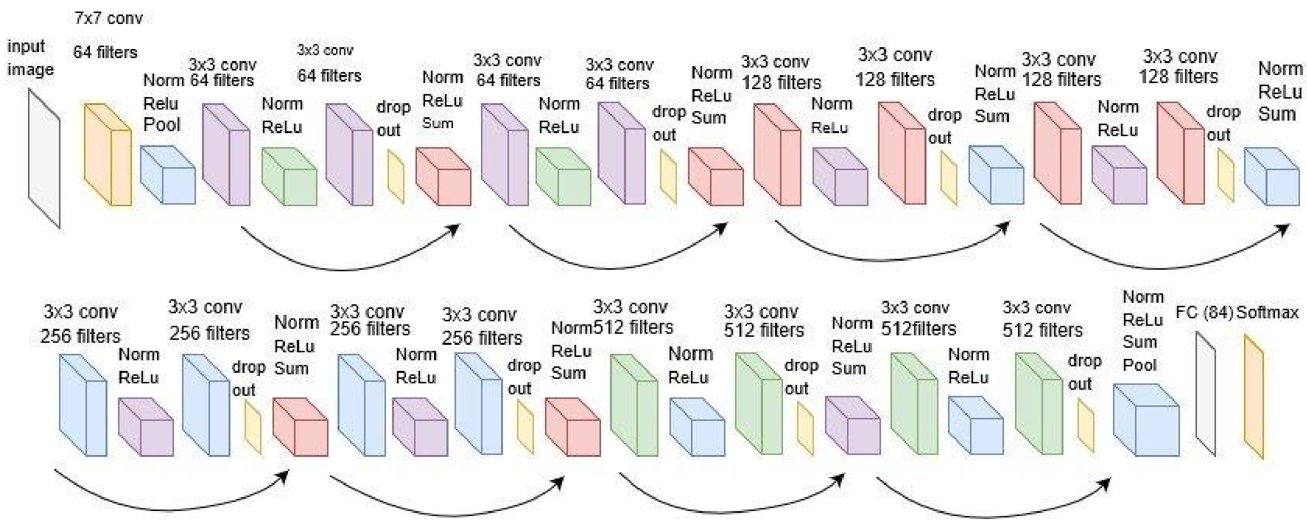


Figure 7: ResNet Architecture

ResNet is short for Residual Network. Many visual recognition tasks have greatly benefited from deep Convolutional Neural Network Models. Over the years there has been a trend to go deeper for more complex tasks. But as we go deeper, the training of neural networks becomes difficult.

In general, in a deep convolutional neural network, several layers are stacked and trained to the task at hand. The network learns several low/mid/high-level features at the end of its layers. In residual learning, instead of trying to learn some features, we try to learn the residuals. Residual can be simply understood as the subtraction of features learned from the input of that layer. ResNet does this using shortcut connections (directly connecting the input of the nth layer to some (n+x)th layer). It has been proven that training this form of networks is easier than training deep convolutional neural networks and also the problem of degrading accuracy is resolved.

ResNet50 is a 50 layer Residual Network. For the details of the ResNet architecture, please refer to this Microsoft research paper: Deep Residual Learning for Image Recognition

Comparison between InceptionV3, VGG16, and ResNet

On the high level the difference among the three networks are:

1. The weights for Inception V3 are smaller than both VGG and ResNet, coming in at the size of 96MB, so it is faster to download and train Inception network;
2. For VGG Network: it is generally slower to train and the network architecture weights themselves are quite large.

CNN	**Accuracy**	**AUC**
InceptionV3	55%	0.61
VGG16	59%	0.67
ResNet50	56%	0.50

Table 1: Comparison between CNNs

Table-1 shows the initial test result on 1,000 image sampling (750 for training and 250 for testing) from the Kaggle dataset. VGG16 gives slightly better performance on both Accuracy and AUC result. ResNet50 has a worse result which we think might be due to the number of layers (50) we are using not being sufficient.

Optimizing CNN

Data Augmentation

Researchers usually use some image transformation techniques to boost the training of the images. Four different transformation types are used here, including flipping, rotation, re-scaling and translation. See Table-2 for explanations:

Transformation	**Description**
Rotation	Rotate 0-360 degrees
Flipping	Flip horizontally or vertically
Rescaling	Randomly scale with scaling factor between 1/1.6 and 1.6

Translation	Randomly shift between -10 and 10 pixels

Table 2: Image Transformation Techniques

Study of different image sizes

The image size from the Kaggle competition dataset is 3072×2048, but this image size is too large to feed into the training pipelines. Researchers usually scale down the image size before training. Various neural networks usually prefer different image input size: VGG prefers 224×224 while InceptionV3 prefers 299×299.

I try various image sizes for training through the Inception architecture, with the results shown below in Table-3. From our experiment, the 512×512 image size still gives the best result. However, 512×512 can only go through the Inception network; it will run out of memory in VGG and ResNet.

Image Size	Accuracy
224×224	53%
299×299	59%
512×512	62%

Table 3: Experiment results on different image sizes (Inception)

Study on different image preprocessing techniques

As described above, there are various image preprocessing techniques we can apply before the training. Table-4 shows the experiment results with preprocessing through the Inception architecture, and the vertical flipping has the best result.

Transformation	Accuracy

90-degree Rotation	60%
Horizontal Flipping	63%
Vertical Flipping	72%
Re-scaling	59%

Table 4: Experiment Results on different image transformation techniques (Inception)

Study on different sampling techniques and different data sample size

Because of the skewed data distribution among 5 DR categories, we only have a few hundred data samples in category 4 and category 5. One technique to compensate for the low data volume in these categories is to replace and reuse the sample data. I use this technique and I also show the impact on different data sampling with the Inception architecture on the final outcome in Table-5:

Sampling Technique	Accuracy
Reuse Data Sample	60%
No data reuse	35%

Table 5: Experiment results on reusing data samples (Inception)

Data sample size	Accuracy	AUC	Training Time
1000	59%	0.67	12 min
1500	62%	0.66	12 min

3500	57%	0.63	12 min
10000	64%	0.66	54 min
20000	64%	0.69	76 min
35000	61%	0.66	144 min

Table 6: Experiment results on different data sample size (VGG)

Table-6 shows the result for different sampling sizes, but the variance on the result is not significant. The accuracy result peaks for the 10,000 and 20,000 data samples.

Note that the training time doesn’t go linearly with the training sample size. This is because the more data samples, the less the number of epochs the CNN could complete. Once Tensorflow detects that there is no improvement after 6 epochs, it aborts the training.

Training with a pretrained model

Building and training a deep neural network (e.g. VGG16) from scratch takes a lot of time. Transferred learning is a technique to transfer a model parameter pretrained in a common public dataset (e.g. ImageNet) to the new image domains. Since the final categories are different between ImageNet (categorization for common objects) and DR (categorization for eye disease). We usually use the no-top-layer pretrained model and then add several layers of normalization and fully connected layer on the top. We will then retrain the final classification layer.

Table-7 shows how we add custom layers on top of the existing pretrained model (VGG16):

Layer (type)	**Output Shape**	**Param Number**
input_1 (InputLayer)	(None, 224, 224, 3)	0
vgg16 (Model)	(None, 7, 7, 512)	14714688

batch_normalization_1	(Batch (None, 7, 7, 512)	2048
flatten (Flatten)	(None, 25088)	0
fcDense1 (Dense)	(None, 4096)	102764544
fcDense2 (Dense)	(None, 4096)	16781312
fcOutput (Dense)	(None, 5)	20485

Table 7: Custom Built VGG16 Network using fully connected layers

Table-8 shows our results on training time and accuracy achieved between the pre-trained model and trained from scratch model for a training set of 1,000 images:

Training Method	**Accuracy**	**AUC**	**Training Time**
VGG Pretrained	59%	0.67	12 min
VGG From scratch	52%	0.46	12 min

Table 8: Experiment results comparing pre-trained and scratch-trained VGG network

Training the VGG model from scratch didn't give us a better result than the pretrained model. We think the image quality of the DR images is not necessarily better than the images from Imagenet. Also, the running time when training from scratch didn't go very long before Tensorflow aborted the training after no improvement. In most of our study in this project, we use pretrained models.

Optimization, Attention Map

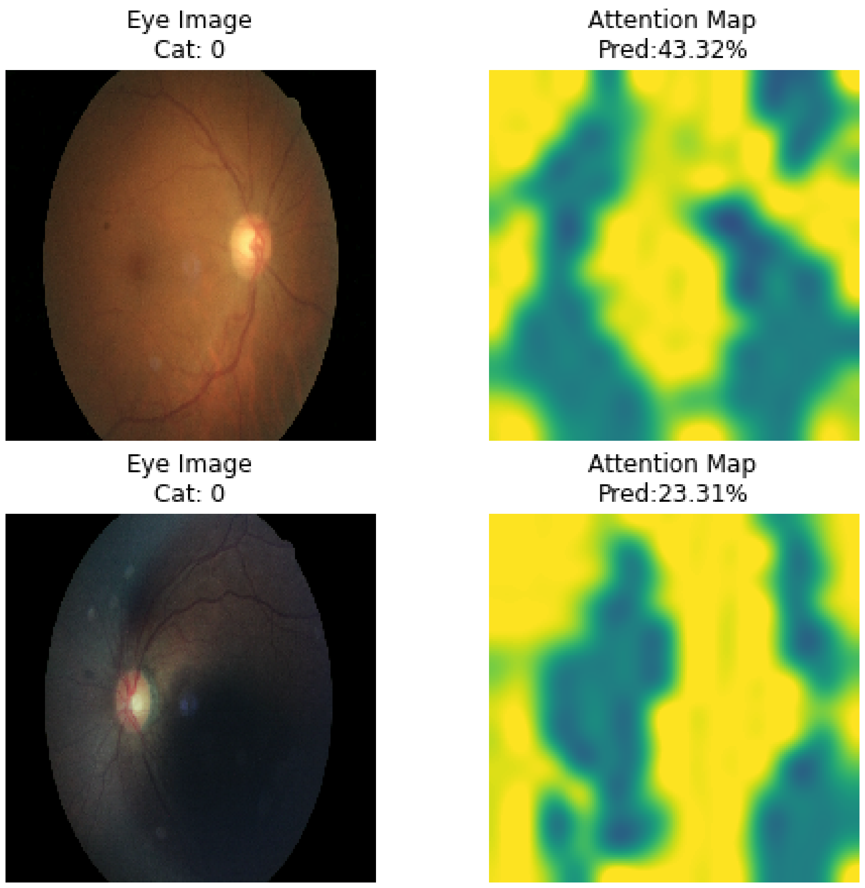


Figure 8: Attention Map Feature

Attention Map, as shown in Figure-8, is a mechanism to expand the capabilities of neural networks. They enable focusing on specific parts of the input, and it has been shown they can improve the performance results of neural processing. I have shown in Figure-8 some fundus images and the attention maps exerted on the region to help processing.

Table-9 shows the layers we are adding on top of pretrained model (vgg16) to build attention map:

Layer (type)	Output Shape	Param Number	Connected to	Notes
input_1 (InputLayer)	(None, 224, 224, 3)	0		
vgg16 (Model)	(None, 7, 7, 512)	14714688	input_1[0][0]	
batch_normalization_1	(BatchNor (None, 7, 7, 512)	2048	vgg16[1][0]	for attention map
dropout_1 (Dropout)	(None, 7, 7, 512)	0	batch_normalization_1[0][0]	for attention map
conv2d_1 (Conv2D)	(None, 7, 7, 64)	32832	dropout_1[0][0]	for attention map
conv2d_2 (Conv2D)	(None, 7, 7, 16)	1040	conv2d_1[0][0]	for attention map
conv2d_3 (Conv2D)	(None, 7, 7, 8)	136	conv2d_2[0][0]	for attention map
conv2d_4 (Conv2D)	(None, 7, 7, 1)	9	conv2d_3[0][0]	for attention map

conv2d_5 (Conv2D)	(None, 7, 7, 512)	512	conv2d_4[0][0]	for attention map
multiply_1 (Multiply)	(None, 7, 7, 512)	0	conv2d_5[0][0]	for attention map
global_average_pooling2d_1 (Glo	(None, 512)	0	multiply_1[0][0]	for attention map
global_average_pooling2d_2 (Glo	(None, 512)	0	conv2d_5[0][0]	for attention map

Table 9: Custom built VGG Network using attention map layers

Experimentation Results for Attention Map Feature

Table-10 shows the result of training with and without attention map optimization:

Test Name	Accuracy	AUC
With Attention Map	59%	0.67
Without Attention Map	58%	0.64

Table 10: Experiment results comparing attention map feature (Inception)

The attention map optimization brings a slight edge in performance. Most of the tests in our study use the attention map feature.

Model Prediction with AUC scores:

AUC is an abbreviation for the area under the curve, as shown in Figure-9. It is used in classification analysis to determine which model predicts the classes best. Figure-9 shows the AUC score for the VGG16 network for 1,000 training samples.

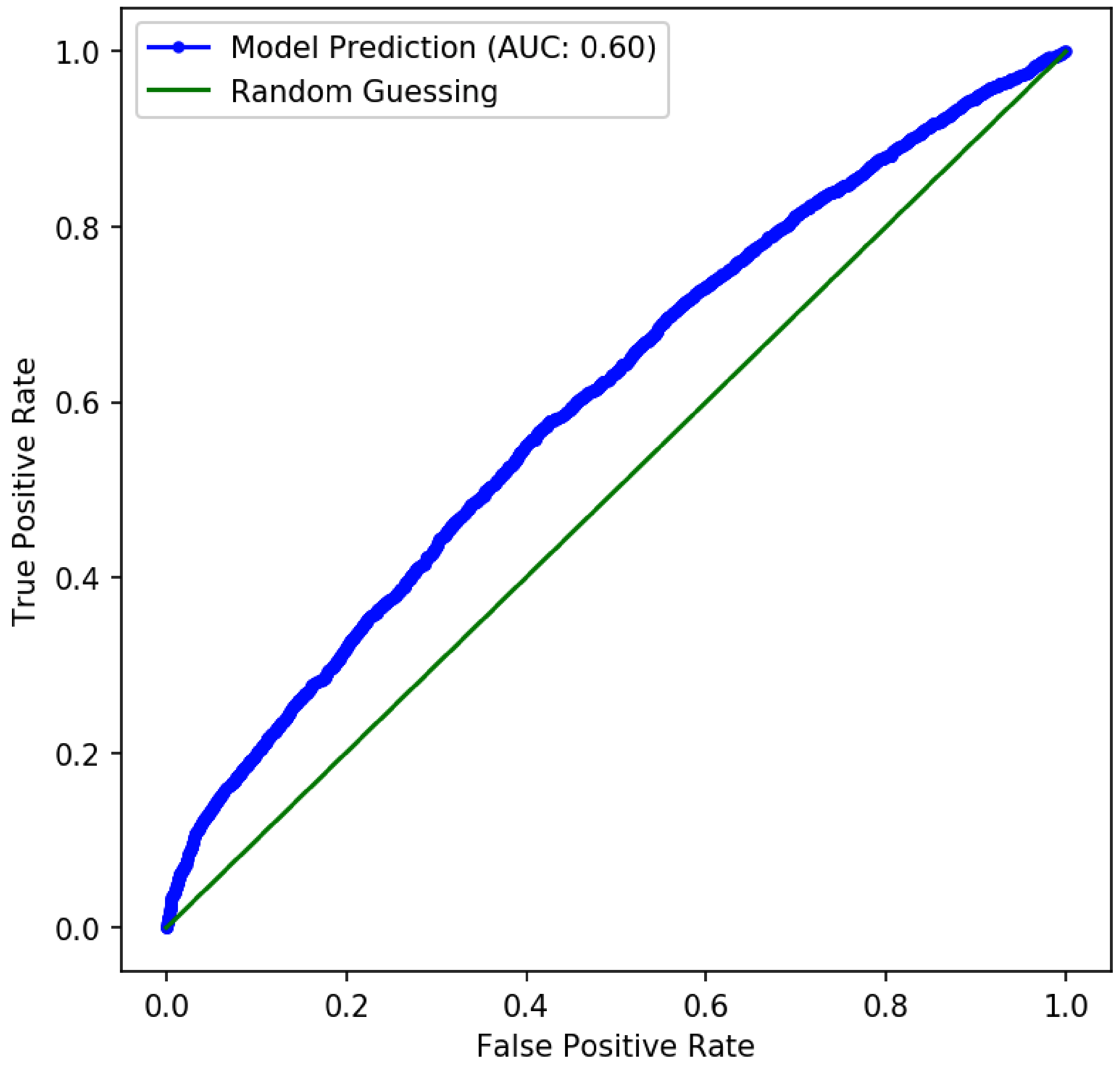


Figure 9: AUC Result

Further Analysis using precision/category matrix and heatmap

	Precision	**Recall**	**f1-Score**	**Support**

0	0.72	0.75	0.74	6704
1	0.12	0.25	0.16	826
2	0.26	0.12	0.16	1561
3	0.13	0.01	0.02	261
4	0.41	0.07	0.12	200
Micro Average	0.57	0.57	0.57	9552
Macro Average	0.33	0.24	0.24	9552
Weighted Average	0.57	0.57	0.56	9552

Figure 10: Precision Per DR Category (0-5)

As shown in Figure-10, the precision/recall on each of the DR categories is different. The top (Category 0) and bottom (Category 4) have the highest precision scores. This is in a way expected, since it is easy to tell whether the patient has no DR or severe DR. It would be much more difficult to tell whether the patient has medium DR or mild DR. Also note we usually use the weighted average (last row) as our reporting metric since the number of data samples in each category is not balanced.

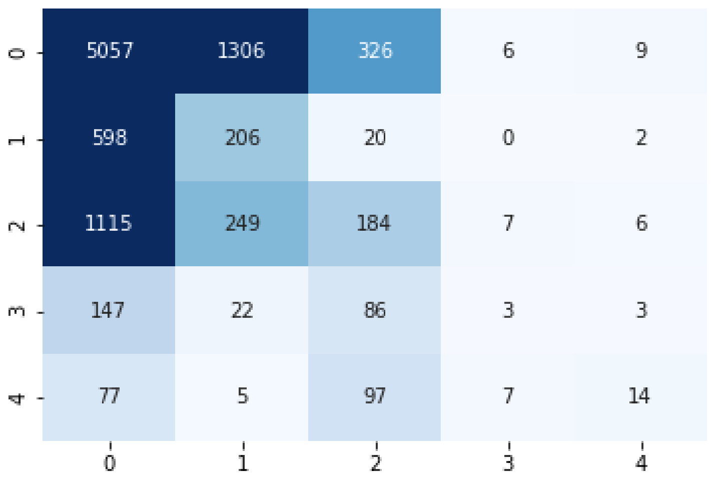


Figure-11: heatmap on actual/predicted result distribution

X-axis: actual DR category; Y-axis: predicted DR category; Value in the matrix: number of data samples

And similarly, I show the heatmap in Figure-11 to see the breakdown of each category on prediction vs. actual classes. As expected, most of the data points are concentrated on the top left corner for category 0.

As shown in these two figures, the prediction accuracy is heavily biased in category 0. Improving the accuracy in other DR categories will be the key to improve the overall accuracy of the training framework.

Conclusion and Future Work

In our study, we adapt several CNNs (Inception, VGG16, and Resnet) for DR stage classification. I compare the effectiveness of the above networks. I also design experiments to evaluate the effect of different parameters (different training sample sizes and different image sizes) on training. In addition, I optimize the network with features such as attention maps and experiment with different image preprocessing.

Our study shows that all three CNNs have reasonable performance, with VGG16 slightly ahead. Optimizing techniques such as attention map or image preprocessing will also help. However, 5-stage DR classification is still a hard problem, especially when differentiating between the middle stages of DR.

Our work in DR classification with CNN neural networks is still very primitive. Given more time and resources, we would like to extend the study to cover:

1. Comparison studies on more neural network architectures such as GoogleNet
2. Training on high graded datasets with high quality images
3. Training on more balanced datasets on each subcategory of DR
4. Better data sampling techniques; reusing data samples or enabling Tensorflow sampling function might cause data bias
5. Introduce more target attributes (e.g. age)

Related Work

There is extensive research being carried out on methods of DR with encouraging results, especially in the area of binary classification.

In [1], Gulshan et al. from Google Labs have trained a custom neural network using 128000 expert graded images. They achieved a sensitivity of 97% on the EysPACS-1 dataset. It doesn't give a breakdown result of each sub-category of DR. Based on the authors' discussions, the results are benefited from the large training dataset, the training images are graded multiple times by experts, but the results are still limited to a specific DR, not for the combination of DR and other eye diseases. In [2], Pratt et al. have trained a custom neural network with about a dozen layers. The training dataset was based on 35000 Kaggle competition data sets, they were using preprocessing techniques such as color normalization and augmentation. They achieved an accuracy of about 75% overall with the accuracy of middle classes around 50%. In [3], Lam et. al in Stanford group has trained the neural network model based on GoogleNet and AlexNet. They have used two datasets with images coming from Kaggle and Messidor-1 datasets. They have tuned the network performance using techniques such as batch normalization, L2 regularization, gradient descent update rules. They were able to achieve an accuracy of 85% and 75% on no DR and severe DR cases respectively but only 29% on mild DR. They also compared the network performance on a pretrained network with the pretrained model achieving 74% accuracy.

Prior to the rise of neural networks, most of the research on DR classification was based on traditional machine learning methods such as support vector machines, random forest and XGBoost. The work usually involved heavy feature extraction and hence can only be done on a small dataset. In [10], Xu et al compared the implementations with XGBoost (accuracy 89%) and CNN (accuracy 95%), the training was based on MxNet. In [11], Jaafar et al. based their study on hand-made top-down image segmentation and feature extraction tools to extract features such as blood vessels, foves, exudates and optic discs. The achieved accuracy of 93% on a very small 147 image dataset. In [4] Ramon et al. based their study on a traditional random forest learning model on a small 100 image dataset with about 90% accuracy. They also compared their results with the traditional logistic regression which is about 70% accuracy.

There are many CNN architectures proposed in the academia, in our research we studied [12] for Inception architecture, [13] for VGG16 design and [14] for Resnet. For the optimization of the network, I studied Attention Map feature [15].

References

1. Varun Gulshan, Lily Peng, Marc Coram. *Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs* (JAMA Network, December 1, 2016.) https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45732.pdf (https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45732.pdf)
2. Harry Pratt, Frans Coenen, Deborah Broadbent, Simon Harding, Yalin Zheng. *Convolutional Neural Networks for Diabetic Retinopathy* (20th Conference on Medical Image Understanding and Analysis (MIUA 2016) July 25, 2016). https://www.sciencedirect.com/science/article/pii/S1877050916311929 (https://www.sciencedirect.com/science/article/pii/S1877050916311929)
3. Carson Lam, Darvin Yi, Margaret Guo, Tony Lindsey. *Automated Detection of Diabetic Retinopathy using Deep Learning* (Proceedings – AMIA Joint Summits on Translational Science. May 18, 2018). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5961805/ (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5961805/)
4. Casanova, Ramon, Santiago Saldana, Emily Y. Chew, Ronald P. Danis, Craig M. Greven, and Walter T. Ambrosius. *Application of Random Forests Methods to Diabetic Retinopathy Classification Analyses*. (PLOS ONE 9, no. 6 (2014): 1-7. Accessed December 26, 2014). www.plosone.org (http://www.plosone.org/).
5. Sinthanayothin, C., J.F. Boyce, T.H. Williamson, H.L. Cook, E. Mensah, S. Lal, and D. Usher. *Automated Detection of Diabetic Retinopathy on Digital Fundus Images.* (Diabetic Medicine 19 (2002)): 105-12.
6. Usher, D., M. Dumskyjs, M. Himaga, T.H. Williamson, S. Nussey, and J. Boyce. *Automated Detection of Diabetic Retinopathy in Digital Retinal Images: A Tool for Diabetic Retinopathy Screening.* (Diabetic Medicine 21 (2003)): 84-90.
7. Jaafar, Hussain F., Asoke K. Nandi, and Waleed Al-Nuaimy. *Automated Detection And Grading Of Hard Exudates From Retinal Fundus Images.* (19th European Signal Processing Conference (EUSIPCO 2011), 2011), 66-70.
8. CDC*. National Diabetes Statistics Report, 2014.* (Centers for Disease Control and Prevention. January 1, 2014). Accessed December 26, 2014.
9. The World Health Organization. *Diabetes.* (The World Health Organization. November 1, 2014. Accessed December 26, 2014). http://www.who.int/mediacentre/factsheets/fs312/en/ (http://www.who.int/mediacentre/factsheets/fs312/en/).
10. Xu Kele, Feng Dawei, Mi Haibo, *Deep Convolutional Neural Network-Based Early Automated Detection of Diabetic Retinopathy Using Fundus Image*, (Second CCF

Bioinformatics Conference, 23 November 2017). https://www.mdpi.com/1420-3049/22/12/2054 (https://www.mdpi.com/1420-3049/22/12/2054)

11. Hussain F. Jaafar, Asoke K. Nandi, Waleed Al-Nuaimy. *AUTOMATED DETECTION AND GRADING OF HARD EXUDATES FROM RETINAL FUNDUS IMAGES*, (19th European Signal Processing Conference, September, 2011) .https://www.eurasip.org/Proceedings/Eusipco/Eusipco2011/papers/1569416955.pdf (https://www.eurasip.org/Proceedings/Eusipco/Eusipco2011/papers/1569416955.pdf)
12. Christian Szegedy, Wei Liu, Yangqing Jia. *Going Deeper with Convolutions*, (IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015), https://arxiv.org/abs/1409.4842 (https://arxiv.org/abs/1409.4842)
13. Karen Simonyan, Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*, (ICLR 2015), https://arxiv.org/abs/1409.1556 (https://arxiv.org/abs/1409.1556)
14. Kaimine He, Xiangyu Zhang, Shaoqing Ren. *Deep Residual Learning for Image Recognition*. (IEEE Xplore Dec 2015), https://arxiv.org/pdf/1512.03385.pdf (https://arxiv.org/pdf/1512.03385.pdf)
15. Adam R. Kosiorek, Alex Bewley, Ingmar Posner. *Hierarchical Attentive Recurrent Tracking*. (Published as a conference paper at NIPS 2017. June 2017), https://arxiv.org/abs/1706.09262 (https://arxiv.org/abs/1706.09262)
16. National Eye Institute *Facts About Diabetic Eye Disease.* (*National Eye Institute*, U.S. Department of Health and Human Services, 1 Sept. 2015), nei.nih.gov/health/diabetic/retinopathy. https://nei.nih.gov/health/diabetic/retinopathy (https://nei.nih.gov/health/diabetic/retinopathy)

About the Author

Rachel Cai is a senior student in California, USA. She has been involved in many STEM related activities: (1) Research Intern at National Institutes Of Health (NIH) in Maryland (2) Gold level contestant in US Computing Olympiad (3) Participant in UC COSMOS. Her main interest is biomedical engineering and computer science. Her past projects/research can be found at: https://github.com/cairachel9
Arti

⟵ Does CRISPR Cas9 have the potential to solve all of society’s problems? (https://ysjournal.com/does-crispr-cas9-have-the-potential-to-solve-all-of-societys-problems/)

Investigations On Isotopic Elements In Terms Of Quarks (https://ysjournal.com/investigations-on-isotopic-elements-in-terms-of-quarks/) ⟶