Detecting Diabetic Retinopathy

Trisha Jani
Stanford University
trishaj@stanford.edu

Krishan Kumar
Stanford University
krishank@stanford.edu

Abstract

The project tackles classifying about 35,000 images of retinas to classify the likelihood of Diabetic Retinopathy being present. The images were classified into five categories, but the dataset is imbalanced and largely has images from one class. The images were preprocessed to remove the black margins present and the dataset was augmented with mirrored and rotated copies of the images. ResNets, SqueezeNets, and Inception v3 networks are used in both the original five class classification problem and a relaxed three class classification to address the class imbalance. These networks are measured on accuracy, precision, recall, and F1 scores and their performances are compared. The best performing model was the three-class Inception v3 with a 78% accuracy on the test set.

1. Introduction

Our project follows from a Kaggle challenge by the California Healthcare Foundation, which aims to find effective machine learning models that can aid in detecting Diabetic Retinopathy (DR). This disease is a leading cause of blindness and is an effect of long-term diabetes. One-third of patients with diabetes will be diagnosed with DR. Early detection of this condition is critical for good prognosis.

Diabetes is estimated to affect 29.1 million people in the United States and 347 million people worldwide. Moreover, around 40-45 percent of Americans with diabetes are at some stage of DR. Detecting DR can be difficult, because the disease has few symptoms. By the time the disease is detected, it can be too late to provide treatment. Highly trained clinicians are required to diagnose the disease, and the process takes several days and often ultimately leads to a lack of follow up from patients and thus delayed treatment. Furthermore, the equipment and expertise required for diagnosis is expensive and limited [1].

Comprehensive and automated DR detection would substantially alleviate this problem, and thus we aim to use convolutional neural networks trained on several thousands of images provided by Kaggle and classify them into five possible ratings of how likely the disease is present.

2. Dataset

2.1. Overview

The data is provided on Kaggle, and consists of 35,124 high-resolution images of retinas. There are images of both the left and right eye of patients, and each image has a rating 0-4 given to it by a clinician that describes how likely the clinician thinks that eye has DR.

Some of the images are inverted, and the California Healthcare Foundation indicates there is possible noise in both images and labels. Some of the images are also out of focus, underexposed, or overexposed. Since we do not know which images are misclassified, we hope to develop a model that will not overfit to the noise present in the dataset.

2.2. Preprocessing

Upon visually examining specific images in the dataset, we noticed that the images have very different lighting levels, focus, zoom, color contrast, and brightness. A lot of the images also have black borders of varying widths. The images were also of different dimensions and resolutions, with different alignments. To account for all this heterogeneity, we had to implement a decent amount of preprocessing in an effort to control for the variability among the images.

We first wanted to adjust for zoom and alignment variation among all the images. To do so, we used OpenCV to find the contours in a greyscale version of the image. We then found the maximum area contour in the image and generated a bounding box around it. We wanted the aspect ratio of this bounding box to be at least 16:9; if not, we increased the height of the image to meet the ratio. We then cropped the image according to this bounding box. These ideas and code were inspired by the preprocessing code done by Kuchibhotla [2].

Lastly, since all these images took up over 38GB of space, we scaled them to be 256x256 pixels. This resulted in similarly aligned images of the same resolution without unnecessary borders.

Below, we show an example of an image before and after

it has been processed using the procedure described above.

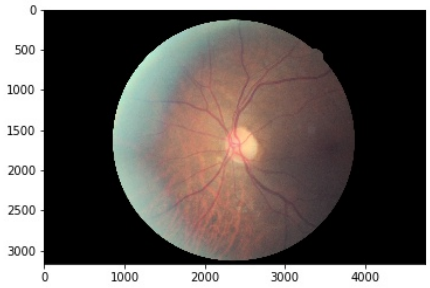

Figure 1. Original image of eye with border

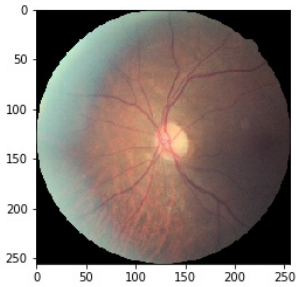

Figure 2. Processed image of eye, centered with border removed

2.3. Data Augmentation

We reserved 25% of our data set as our test set. This corresponds to 8781 images for test. Of the remaining data, we reserved another 25% for validation. So, our training set consisted of 19757 images, and our validation set consisted of 6586 images. We used stratified random sampling to ensure that the distribution of classes among the data subsets remained equal (further discussed in section 3.2.2).

Upon examining the distribution of classes, we noticed we have a class imbalance. The table below shows the distribution of classes.

Class	Number	Percentage
0	25808	73.48%
1	2443	6.96%
2	5292	15.07%
3	873	2.49%
4	708	2.02%

We suspect that this class imbalance will be an issue. We address it by augmenting our training dataset with mirrored and rotated versions of images of eyes with DR, as well as by employing weighted sampling.

3. Methods

3.1. Background and Related Work

The National Eye Institute has categorized cases of DR by severity into four different classes. These classes relate to the categories we try to predict with our neural network. NPDR describes non-proliferative DR cases, and PDR describes proliferative DR cases. The classes are described as follows [3]:
- Mild NPDR - small balloon-like swelling in retinal blood vessels, micro-aneurysm lesions
- Mild NPDR - blood vessel distortion, greater swelling
- Severe NPDR - secretion from abnormal growths caused by blocked blood vessels
- PDR - new blood vessels that can leak emerge into the retinal surface, and retinal detachment can occur

For reference, human family doctors have achieved a 54% sensitivity and 87% specificity when detecting DR [4].

Previous work has been done in using machine learning techniques to detect diabetic retinopathy. A 2013 study used compared the results of using a Bayes Classifier, a Probabilistic Neural Network, and a SVM. However, this study only made use of 350 images and did not make use of Convolutional Neural Networks, which often work better with image data. Furthermore, while this study reported high training accuracies, these models would not perform well on new test data [5].

However, with the amount of data this challenge provides, people have been able to train CNNs to try and solve this problem. For example, Gulshan et al [6] used a CNN trained on a dataset of 120,000 labeled images to solve a binary classification problem of mild/severe DR. They were able to achieve sensitivities in the mid 90% range on various test sets.

Rakhlin [7] works with the publicly available Kaggle dataset, augmented with images from a private dataset, and tested on a private dataset. He used transfer learning and modifies the VGG network to train his network. This reaches a sensitivity of 99%, specificity of 71%, and AUC if 0.97.

Efforts to automate DR detection have not exclusively relied on utilizing neural networks. For instance, Quellec et. all attempt to detect microaneurysms in retina photographs, which are often the first lesions to appear with the onset of DR. They rely on a template-matching based algorithm which was able to detect 98% of the lesions in their learning set of 914 images [8]. Though this approach is successful for detecting microaneurysms, it does not necessarily detect DR.

Abramoff et. al evaluated the images of 16,670 diabetes patients who had not yet been diagnosed with DR. The team employed a kNN classifier to assign probabilities of image pixels being of red lesions and obtained an AUC of 0.84 [9].

Another study using an SVM was able to obtain sensitivities and specificities of approximately 90%. However, like the other studies, the dataset was limited to a few-thousand images [10].

This lack of data is seen in studies such as that done in study [11], which uses an SVM trained on 331 images to obtain a sensitivity of 82%, and that done by Massignan et al, who relied on a binary classifier to classify a dataset of only 55 patients [12].

3.2. ResNet-18

3.2.1 Baseline

We start by solving this 5-class classification problem. All code was developed in Python, and networks were developed with PyTorch [17]. We preprocess our data as described above, and as our baseline use weights from a pretrained ResNet-18 model. In general, as networks get deeper, they are harder to train due to the vanishing and exploding gradients problem. The ResNet architecture tackles this issue by making use of residual blocks, which allow the network to learn a residual mapping function, rather than the original mapping. This feature allows ResNets to be deeper and more powerful. In fact, ResNet won 1st place in the ILSVRC 2015 classification and COCO 2015 competition in ImageNet Detection [14]. This is a reason we were interested in starting with this network.

We replace the final layer to be a softmax layer, with loss function defined as

$$L_i = -\log(\frac{e^{f_{y_i}}}{\sum_j e^{s_j}}) = -f_{y_i} + \log(\sum_j e^{s_j})$$

We trained the network with a batch size of 128 using the Adam optimizer with the default parameter settings. We had also tried SGD, but Adam converged faster. We used a learning rate scheduler, reducing the learning rate by a factor of 10 if loss fails to decrease for 5 continuous epochs. After reaching convergence, we use the network weights of the model with the lowest loss on the validation set.

The table below summarizes the predictions on the test set. The model achieves a loss of 0.8588 and an accuracy of 73.807% on the test set.

Table 1. ResNet Results

Class	Precision	Recall	F1
0	0.75	0.98	0.85
1	0.13	0.00	0.01
2	0.41	0.05	0.08
3	0.31	0.14	0.19
4	0.48	0.18	0.26

Results by class are summarized in the confusion matrix below (true class on vertical axis, predicted class on horizontal axis). We see that the network often incorrectly predicts class 0 for most images. This is likely due to the fact that the majority of images belong to class 0. Since 73.48 % of the images belong to class 0, our method only does a little bit better than always predicting class 0. We next tackle the class imbalance problem to ameliorate this problem.

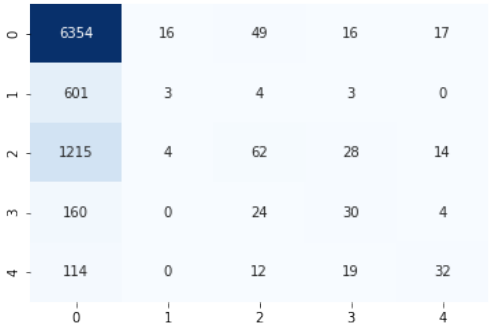

Figure 3. Confusion matrix for Baseline ResNet

3.2.2 Class imbalance problem

We notice that our baseline model has poor recall and F1 scores for Classes 1, 2, 3, 4. This is because there is a major class imbalance. We attempt to solve this problem in two ways. First, we oversample the underrepresented classes during the training phase. So, the probability of being sampled in the training phase is inversely proportional to class frequency. Next, we perform on-the-fly image augmentation during the training phase via random crops, angles, and center invariant affine transforms.

The table below summarizes the predictions on the test set. The model achieves a loss of 1.1201 and an accuracy of 51.999% on the test set. This is much better than random guessing, which would achieve an accuracy of 20%. This suggests that our model is indeed learning.

Table 2. Augmented ResNet Results

Class	Precision	Recall	F1
0	0.83	0.56	0.67
1	0.12	0.00	0.01
2	0.21	0.61	0.31
3	0.28	0.20	0.23
4	0.22	0.47	0.30

The confusion matrix below summarizes performance for each class.

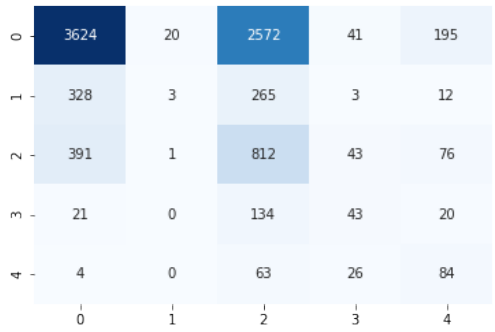

Figure 4. Confusion matrix for ResNet with data augmentation and class balancing

We note a particularly low precision, recall, and F1 score for Class 1. We found this interesting, since class 1 is the third most represented class, at nearly 7%. The confusion matrix shows that most class 1 images are actually predicted to belong to class 0 or class2, the two most populous classes. This result seems to suggest that the weight of class 1 could be adjusted, which is something we would have liked to try, more time permitting.

3.2.3 Weighted loss function

As another way to address the class imbalance problem, we adjusted our loss function. In particular we used weighted cross entropy loss, which appropriately scales the cross entropy loss for an example based on its class. We chose weighting to highly penalize errors on the less frequent classes. Our revised loss function was

$$L_i = weight[y_i] * (-f_{y_i} + \log(\sum_j e^{s_j}))$$

Interestingly, we found that this strategy was less effective that the weighted sampling strategy described above. When we used weights inversely proportional to the frequency of the class, we achieved an accuracy of 38.378 % on the test set. This is much lower than the accuracy achieved above via weighted sampling, which was over 10% higher. Perhaps further tuning the weights would have improved the performance, but we instead decided to stick with the weighted sampling approach to tackle the class imbalance problem moving forwards.

3.2.4 Problem Relaxation

Next, we attempted to solve a relaxation of the original problem. Instead of solving a 5-class problem, we solved a 3-class problem where we classify an image as not having DR (class 0), having early stage DR (class 1 or 2), or having advanced stage DR (class 3 or 4).

We used weighted sampling to account for the class imbalance, and we also augmented our training set with random crops, flips, and rotations. We were able to achieve a loss of 0.7158 and an accuracy of 71.495% on the test set. This is better than random guessing, which would give an accuracy of 33%.

Table 3. ResNet Results for 3-class problem

Class	Precision	Recall	F1
0	0.79	0.90	0.84
1	0.38	0.12	0.18
2	0.30	0.61	0.40

The confusion matrix below summarizes performance for each class.

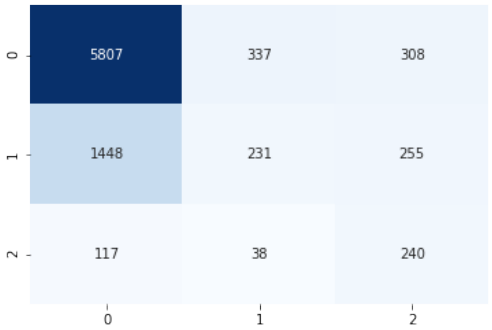

Figure 5. Confusion matrix for ResNet with data augmentation and class balancing

We see above that Class 1 has a low recall and F1 score. This makes sense, since of all the Class 1 images, the majority are incorrectly classified as class 0. A small number are classified as class 2. This suggests that the network has a hard time differentiating between no DR and early stage DR. However, the confusion matrix also shows that fewer class 2 images are classified as class 0, which suggests the model is able to detect to some extent the severity of DR.

3.3. Inception v3

The Inception Net overcomes the issues of choosing the right kernel size for convolution operations and over-fitting present in many deep neural networks by using filters of multiple sizes that operate at the same level. Therefore a network gets wider than it does deeper. A naive inception module uses 3 different filters of sizes 1x1, 3x3, and 5x5 on the input, along with max-pooling. These outputs are concatenated and input to the next module. The Inception v3 specifically improves on this concept by including an RMSProp optimizer, factorized 7x7 convolutions, BatchNorm in the auxiliary classifiers, and a regularizing component added to the loss formula that prevents over-fitting.

When using the Inceptionv3 for transfer learning, we use the pretrained weights and modify the final layers. Incep-

tion is a unique network in that it contains two output layers in the training phase. The primary output layer is a linear layer at the end of the network, whereas the secondary output layer is called an auxiliary output and is present in the AuxLogits part of the network. During testing, we only use the primary output. During finetuning, both layers are reshaped. [16]

3.3.1 5-class Classification

We solve the 5-class classification problem with class balancing and data augmentation by training the network with a batch size of 64 using the Adam optimizer with the default parameter settings. Once again, this converged faster than using SGD.

The table below summarizes the predictions on the test set. The model achieves a loss of 0.9521 and an accuracy of 66.120% on the test set.

Table 4. Inception v3 Results

Class	Precision	Recall	F1
0	0.86	0.76	0.81
1	0.00	0.00	0.00
2	0.30	0.53	0.39
3	0.22	0.65	0.33
4	0.38	0.32	0.35

The confusion matrix below summarizes performance for each class.

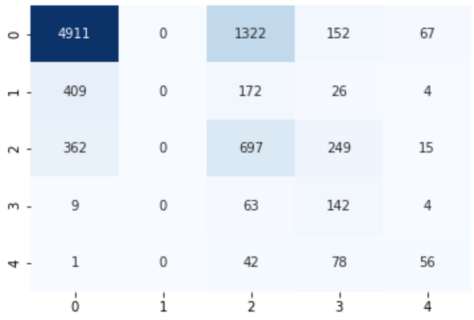

Figure 6. Confusion matrix for Inception v3 5-class classification with data augmentation and class balancing

We find low precision, recall, and F1 score for notably classes 1 and 3. The confusion matrix shows that most class 1 images are predicted to be class 0, and most class 3 images are predicted to be class 4, even with the class balancing. A decent number of class 2 images are incorrectly classified as class 3. This shows that the model has a difficult time differentiating between classes 2 and 3, and overestimates severity.

3.3.2 Problem Relaxation

We try to address the class confusion from the model above by solving a 3-class problem with the Inception Net instead of the 5-class problem, where we classify an image as not having DR (class 0), having early stage DR (class 1 or 2), or having advanced stage DR (class 3 or 4).

We were able to achieve a loss of 0.5530 and an accuracy of 78.362% on the test set.

Table 5. Inception v3 Results for 3-class problem

Class	Precision	Recall	F1
0	0.84	0.92	0.88
1	0.54	0.38	0.45
2	0.59	0.52	0.55

The confusion matrix below summarizes performance for each class.

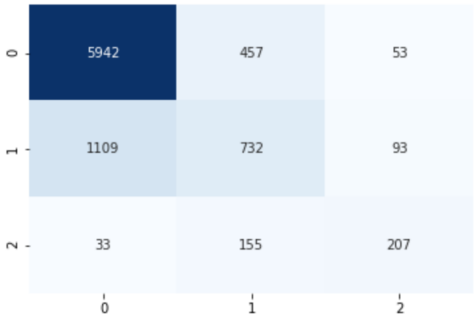

Figure 7. Confusion matrix for Inception v3 3-class classification

Here we note that the precision, recall, and F1 scores are improved from the 5-class classification. Notably, the precision is more balanced out among the classes. Looking at the confusion matrix, for all 3 classes, the plurality of predictions are made in the correct class.

3.4. SqueezeNet

Finally, we decided to solve this classification problem using a SqueezeNet network trained on ImageNet. SqueezeNet claims to achieves AlexNet-level accuracy on ImageNet with 50x fewer parameters. The main idea behind a SqueezeNet is the replacement of 3×3 convolutional filters with 1×1 filters. A combination of "squeeze" and "expand" blocks allow for a big feature map while keeping computation down [15]. We thought it would be interesting to compare the performance of this smaller network with that of larger networks like ResNet and Inception v3.

When using a pretrained SqueezeNet for transfer learning, the output classifier consists of a dropout layer, a 1x1 convolutional layer, a ReLU, and an AvgPool2D layer.

3.4.1 5-class problem

The output of the SqueezeNet comes from a convolutional layer, and we reinitialize this Conv2d layer to have an output feature map of depth 5 to solve our 5-class classification problem. We run our training with class balancing and image augmentation, and achieve an accuracy of 51.236% and loss of 1.1636 on the test set. In comparison, Alban and Gilligan's AlexNet classifier achieved a test accuracy of 40.73 % [13]. However, they do have higher recall and precision scores for the underrepresented classes. The table below details SqueezeNet performance for each class.

Table 6. SqueezeNet Results

Class	Precision	Recall	F1
0	0.82	0.55	0.66
1	0.14	0.02	0.04
2	0.21	0.60	0.31
3	0.30	0.24	0.27
4	0.23	0.51	0.32

The confusion matrix below summarizes performance for each class.

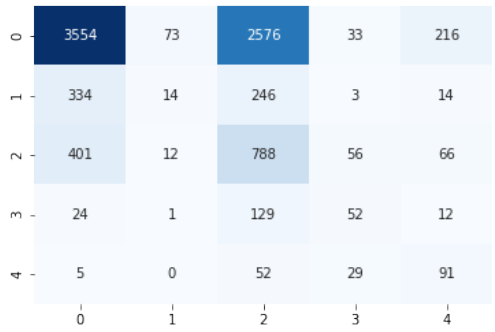

Figure 8. Confusion matrix for SqueezeNet with data augmentation and class balancing

Like the networks above, we note that class 1 has a particularly low recall and F1 score. Interestingly, a large number of class 0 images are misclassified as class 2, rather than class 1. This could be because class 2 is the next most populous class. Perhaps adjusting weights of the random sampler could help mitigate this problem.

3.4.2 Problem Relaxation

We now solve the relaxed 3 class problem with SqueezeNet. Again, in the training stage, we freeze all layers but the last and use image augmentation. We achieve a loss of 0.8093 and an accuracy of 64.856% on the test set. In comparison, Alban and Gilligan's AlexNet classifier achieved a test accuracy of 57.05 % [13]. However, they do have higher recall and precision scores for the underrepresented classes. The table below summarizes SqueezeNet results.

Table 7. SqueezeNet Results for 3-class problem

Class	Precision	Recall	F1
0	0.80	0.76	0.78
1	0.32	0.28	0.30
2	0.24	0.60	0.34

A confusion matrix shows the model performance for each class.

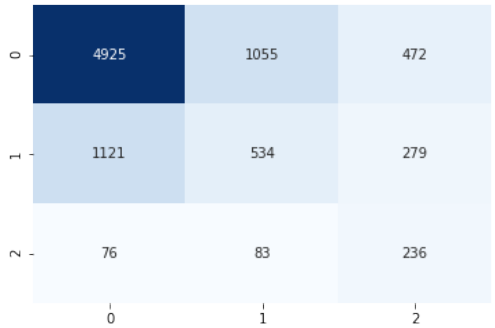

Figure 9. Confusion matrix for SqueezeNet 3-class classification

While not as strong as the scores from Inception network, the recall and F1 scores of the SqueezeNet for class 1 are higher than those of the ResNet. This suggests that the SqueezeNet is overfitting to the data less than the ResNet, which seems reasonable given its smaller size.

4. Discussion

The table below compares the accuracy of the different networks for the 5-class problem and 3-class problem.

Table 8. Model comparison for 5-class problem

Network	5 class Accuracy	3 class Accuracy
ResNet	51.999%	71.495%
Inception v3	66.120%	78.362%
SqueezeNet	51.236%	64.856%

We see that the Inception v3 network has the strongest performance, while the ResNet and SqueezeNet have a similar, weaker performance for the 5 class problem. For the 3 class problem, we again see that the Inception v3 network has the strongest performance out of the three models. The ResNet is the second strongest, followed by the Squeezenet.

We see that in all cases, the performance on the 3-class problem is better than the 5-class, because the class imbalance is less pronounced in the former. The Inception v3's performance might be explained by the fact that it a larger network than the SqueezeNet and thus has potentially greater learning capacity, yet is wider than the ResNet

which prevents the possible overfitting that comes from deeper networks. We were rather impressed by the performance of the SqueezeNet, and surmise that the smaller number of parameters helps combat overfitting.

5. Further Work

Given more computing power and memory, we would run the model for a larger number of epochs. We would try furthering our image preprocessing by filtering and denoising the images. We noticed that the quality of the images varies a lot. Some images have bright spots, while others are very dark. While this variety is good during the training stages and helps build a robust classifier, we did notice that the classifier fails on these types of images.

Below, we show an example of some of the images in the training set. The second image is very dull, while the third image is very bright. During preprocessing, it might be helpful to perform median filtering, brighten dull images, and lower the exposure of the brighter images.

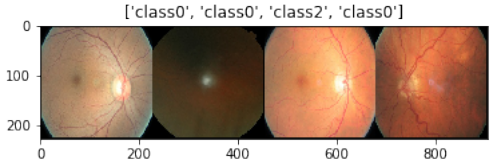

Figure 10. The images have varying brightness and contrast.

In addition to the transfer learning explored, we would compare the performance of these models to other pretrained models such as AlexNet or GoogLeNet. It would also be interesting to starting with pretrained weights and adjust all the weights of the network, instead of freezing all but the last layer and only training the final layer. If we had more time, we could have also played with architectures and added more convolution or fully connected layers after the frozen layers and before the final softmax layer.

It would also be interesting to leverage the fact that we have images of both eyes for a single individual. We built our model assuming each image was independent. However, we could perhaps combine information from both eyes (especially in the 3-class scenario) to determine the stage of DR.

6. Contributions and Acknowledgements

We referred to a public GitHub repo to get a high level understanding of how to load the raw data: `https://www.kaggle.com/meenavyas/diabetic-retinopathy-detection`. For the image preprocessing, we made use of code from the public GitHub of Jayaram Kuchibhotla, found at `https://github.com/jayaram1125/Diabetic-Retinopathy-Detection-using-CNN`. For information on how to set up transfer learning and training/test loop, we made use of the following PyTorch tutorial: `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html`. We also referred to this PyTorch tutorial for more details on fine-tuning networks: `https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html`. Trisha did the data setup and preprocessing. Both Trisha and Krishan trained the models included and wrote the paper. We want to thank Tien-Ning Hsu and Owen Wang for helping resolve data download issues in the early stages of the project.

7. References

References

[1] Diabetic Retinopathy Detection. https://www.kaggle.com/c/diabetic-retinopathy-detection/overview.

[2] Jayaram Kuchibhotla. Preprocessing Code. https://github.com/jayaram1125/Diabetic-Retinopathy-Detection-using-CNN

[3] Vinod Patel Eva M Kohner and Salwan M B Rassam. Role of blood flow and impaired autoregulation in the pathogenesis of diabetic retinopathy. American Diabetes Association.

[4] Gill, James M., et al. "Accuracy of screening for diabetic retinopathy by family physicians." The Annals of Family Medicine 2.3 (2004): 218-220.

[5] R. Priya and P. Aruna. Diagnosis of diabetic retinopathy using machine learning techniques. Journal on Soft Computing, 3(4): 563575.

[6] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA, 316(22):24022410, 2016

[7] A. Rakhlin. Diabetic Retinopathy detection through integration of Deep Learning classification framework. bioRxiv. June 2018.

[8] G. Quellec, M. Lamard, P. M. Josselin, G. Cazuguel, B. Cochener, and C. Roux. Optimal wavelet transform for the detection of microaneurysms in retina photographs. IEEE Transactions on Medical Imaging, 27(9):12301241, 2008.

[9] M. D. Abramoff, J. M. Reinhardt, S. R. Russell, J. C. Folk, ' V. B. Mahajan, M. Niemeijer, and G. Quellec. Automated early detection of diabetic retinopathy. Ophthalmology, 117(6):11471154, 2010.

[10] A. Ur. Decision support system for diabetic retinopathy using discrete wavelet transform. Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine, 227(3):251261, 2013.

[11] Ng EY Chee C Tamura T. Acharya U, Lim CM. Computer-based detection of diabetes retinopathy stages using digital fundus images. Proceedings of the Institute of Mechanical Engineers, 545-553, 2009.

[12] Benetti E. Massignan F. Pilotto E. Varano M. Cavarzeran F. Avogaro A. Vujosevic, S. and E. Midena. Screening for diabetic retinopathy: 1 and 3 nonmydriatic 45-degree digital.

[13] Marco Alban and Tanner Gilligan. Automated Detection of Diabetic Retinopathy using Fluorescein Angiography Photographs. CS 231n Final Project, 2016.

[14] K He, X Zhang, S Ren, J Sun. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2015.

[15] F N Iandola, S Han, M W Moskewicz, K Ashraf, W J Dally, K Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size. arXiv:1602.07360, 2016.

[16] C Szegedy, V Vanhoucke, S Ioffe, J Shlens, Z Wojna. Rethinking the Inception Architecture for Computer Vision. arXiv:1512.00567, 2015.

[17] Paszke, Adam and Gross, Sam and Chintala, Soumith and Chanan, Gregory and Yang, Edward and DeVito, Zachary and Lin, Zeming and Desmaison, Alban and Antiga, Luca and Lerer, Adam. Automatic differentiation in PyTorch. NIPS, 2017.