Blindness Detection

CS 584 – Machine Learning

Spring 2023

Dr. Yan Yan

Girish Rajani-Bathija (grajanibathija@hawk.iit.edu) – A20503736

Shriya Prasanna (sprasanna@hawk.iit.edu) – A20521733

Bhavesh Rajesh Talreja (btalreja@hawk.iit.edu) – A20516822

Intermediate Project Report

1. Introduction

Diabetic retinopathy is one of the most common causes of blindness in people who have diabetes. Having high blood sugar from diabetes for a prolonged period of time can lead to diabetic retinopathy and if not treated early, can lead to vision loss. For early detection of DR, there needs to be a manually screening exam that can identify the risk and level of someone developing this disease. However, in some rural areas, this exam can be difficult to conduct due to lack to access to infrastructure and skilled professionals. Additionally, this manual process of screening can be very time consuming and labor intensive. Due to the ever increasing demand for diabetic retinopathy screening, there arises a need to automate such a process.

The goal of this project is to use algorithms in Machine Learning and Deep Learning such as Convolutional Neural Network to extract features from retina images to automate this process. This will be done by performing multi-class image classification using the APTOS 2019 Blindness Detection dataset. Machine learning models allow us to speed up the process of detecting DR early and making this service available to those who may not have access to the infrastructure.

2. Dataset

The APTOS 2019 Blindness Detection dataset available on kaggle will be used for this project. This real-world dataset consists of retina images from multiple clinics (3,662 training images and 1,928 testing images) in rural areas of India. Each training image has been carefully examined and labeled as belonging to 1 of 5 classes - No DR, Mild DR, Moderate DR, Severe DR, and Proliferative DR.

3. Background and Related Work (to be completed)

[1] Lam C, Yi D, Guo M, Lindsey T. Automated Detection of Diabetic Retinopathy using Deep Learning. AMIA Jt Summits Transl Sci Proc. 2018 May 18;2017:147-155. PMID: 29888061; PMCID: PMC5961805.

[2] Sreng, S.; Maneerat, N.; Hamamoto, K.; Panjaphongse, R. Automated Diabetic Retinopathy Screening System Using Hybrid Simulated Annealing and Ensemble Bagging Classifier. Appl. Sci. 2018, 8, 1198. https://doi.org/10.3390/app8071198.

[3] Ting DSW, Cheung CY, Lim G, Tan GSW, Quang ND, Gan A, Hamzah H, Garcia-Franco R, San Yeo IY, Lee SY, Wong EYM, Sabanayagam C, Baskaran M, Ibrahim F, Tan NC, Finkelstein EA, Lamoureux EL, Wong IY, Bressler NM, Sivaprasad S, Varma R, Jonas JB, He MG, Cheng CY, Cheung GCM, Aung T, Hsu W, Lee ML, Wong TY. Development and Validation of a Deep Learning System for Diabetic Retinopathy and Related Eye Diseases Using Retinal Images From Multiethnic Populations With Diabetes. JAMA. 2017 Dec 12;318(22):2211-2223. doi: 10.1001/jama.2017.18152. PMID: 29234807; PMCID: PMC5820739.

[4] Ramachandran, N., Hong, S.C., Sime, M.J. and Wilson, G.A. (2018), Diabetic retinopathy screening using deep neural network. Clin. Experiment. Ophthalmol., 46: 412-416. https://doi.org/10.1111/ceo.13056.

[5] Ismael, H. R., Abdulazeez, A. M., & Hasan, D. A. (2021). Detection of Diabetic Retinopathy Based on Convolutional Neural Networks: A Review. Asian Journal of Research in Computer Science, 8(3), 1–15. https://doi.org/10.9734/ajrcos/2021/v8i330200.

4. Methods

4.1 Preprocessing

The retina images within this dataset required various preprocessing techniques before using them in the Machine Learning models.

4.1.1 Train/Validation/Test Split

For this project, the validation set approach was selected, and so it was necessary to split the training dataset consisting of 3,662 samples into training set (2,462 samples) and validation set (1,200 samples).

Additionally, all of the training images were in one folder, but for this multi-class classification problem, it was necessary to have all images for each class in separate folders based on their label. For both the training and validation sets, five folders were created (one for each class), and the data were split into their respective classes using the diagnosis feature within the train_images.csv file. Now that the training, validation, and test set have been split as needed, the images had to be preprocessed.

Table 1: Number of Samples in Training, Validation, and Testing Sets

	Training Set	Validation Set	Testing Set
Number of Samples	2,462	1,200	1,928

4.1.2 Image Resizing

It was observed that the dimensions of the images were different, ranging from 474 x 358 pixels to 3388 x 2588 pixels. For an equal comparison, it is crucial that all images are uniform in size and dimension. To accomplish this, all images within the dataset were resized to 200 x 200 pixels for uniformity. Sample images before and after resizing can be observed in Figures 1 and 2 below.

```
Class: [0] Height: 1050 Width: 1050
Class: [1] Height: 1958 Width: 2588
Class: [2] Height: 1226 Width: 1844
Class: [3] Height: 1944 Width: 2896
Class: [4] Height: 1958 Width: 2588
```

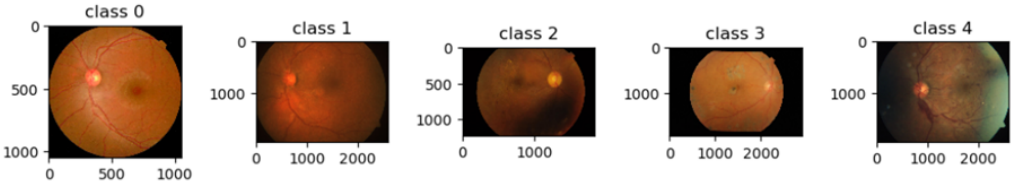

Figure 1 - Showing a sample image from each class along with their dimensions before resizing

```
Class: [0] Height: 200 Width: 200
Class: [1] Height: 200 Width: 200
Class: [2] Height: 200 Width: 200
Class: [3] Height: 200 Width: 200
Class: [4] Height: 200 Width: 200
```

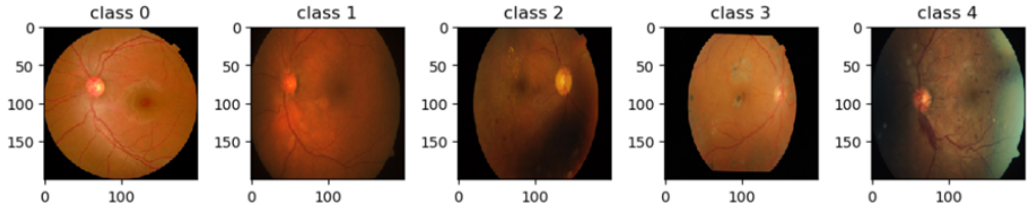

Figure 2 - Showing a sample image from each class along with their dimensions after resizing

4.1.3 Image Augmentation

The final step in the data preprocessing stage was data augmentation. It was observed that the distribution of samples among the five classes varied significantly. Table 2 below shows this distribution which allows us to conclude that the dataset is imbalanced.

Table 2: Number of Images in Training and Validation Set before Image Augmentation

Class	Training Samples	Percentage Distribution	Validation Samples	Percentage Distribution
0 - No DR	1249	51%	556	46%
1 - Mild	244	10%	126	10.5%
2 - Moderate	641	26%	358	30%
3 - Severe	126	5%	67	5.5%
4 - Proliferative DR	202	8%	93	8%
Total:	**2462**	100%	**1200**	100%

As shown above, it can be seen that in both the training and validation set, classes 1, 3 and 4 have significantly less samples than class 0 and 2. Therefore, to prevent bias in training, data augmentation (rotation) was performed on the classes that had less samples to increase the number of samples within those classes and achieve a more balanced distribution.

Table 3 below shows the distribution of the images within each class after performing data augmentation on the three classes.

Table 3: Number of Images in Training and Validation Set before Image Augmentation

Class	Training Samples	Percentage Distribution	Validation Samples	Percentage Distribution
0 - No DR	1249	30%	556	27%
1 - Mild	976	23%	504	24%
2 - Moderate	641	15%	358	17%
3 - Severe	504	12%	268	13%
4 - Proliferative DR	808	19%	372	18%
Total:	**4178**	100%	**2058**	100%

4.2 Data Augmentation

Although data augmentation had been performed within the preprocessing stage as explained above, that was only on 3 out of the 5 classes with the purpose of

achieving a more balanced distribution of samples within each class to mitigate any effect of bias during training.

During training, for each model that is created, it will be trained on training data with and without data augmentation to see how increasing all of the samples affects the model's ability to learn and generalize better. In this stage when performing data augmentation, all samples within the training set will be augmented and various methods will be used, such as rotation, width shift, height shift, shear, zoom, and horizontal flip. Table 4 below shows the range/values for each parameter used during the data augmentation process.

Data Augmentation Parameter	**Value**
rotation_range	40
width_shift_range	0.2
height_shift_range	0.2
shear_range	0.2
zoom_range	0.2
horizontal_flip	True
fill_mode	'nearest'

Table 4 - Showing each parameter used in data augmentation during training and their values

4.3 Custom Model - CNN Architecture

The custom model for this project consists of a Convolutional Neural Network (CNN). CNN's are one of the most commonly used techniques in Deep Learning for image recognition and classification. A CNN consists of various types of layers:

- Convolutional layer
- Pooling layer
- Fully-connected (FC) layer
- Batch Normalization layer
- Dropout layer

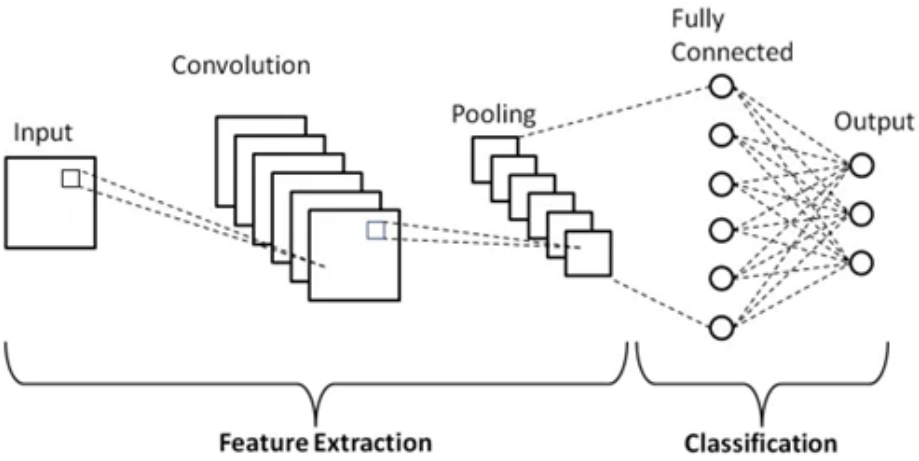

Figure 3 - Showing a CNN Architecture for image classification (Source)

As shown above, the CNN performs feature extraction. The early convolution layers would extract simple features such as learning directional derivatives (they look for edges in a particular direction). The deeper convolution layers would learn more complex patterns other than edges and direction, and as the layers go deeper and deeper, it learns to identify even more complex patterns (higher-level features) that the early layers are not able to detect, such as hard exudate, a feature for classifying diabetic retinopathy.

The CNN Architecture for the custom model in this project consists of various convolution layers with different filters at each stage, 3x3 kernel size, a stride of 1, and relu activation. After each convolution layer, a 2x2 max pooling layer was added along with a batch normalization layer. After the final convolution block, a flatten layer was added along with a dropout of 0.5 as a regularization technique to help reduce overfitting. This network used softmax activation in the output layer, Adam optimizer with a learning rate of 0.001, and categorical cross-entropy as the loss function.

4.4 Transfer Learning

Transfer learning is another popular technique in Deep Learning which consists of using a pre-trained model on a new problem. This is especially useful for reduced training time, increased performance, and when there is less data available. Since the pre-trained models have been trained on a large amount of data, they tend to generalize well on new tasks and can be used on datasets such as the one for this project.

In this project, various transfer learning models will be used such as VGG16 and ResNet. These models will then be compared to the custom CNN model in 4.3 during evaluation.

4.4.1 VGG 16

VGG16 is a type of CNN that is one of the most popular models today. It utilizes very small 3x3 convolution filters with a stride 1 and contains approximately 138 trainable parameters. It is most commonly used for object detection and classification being able to achieve an average accuracy of 92.7%. The architecture of VGG16 is explained further below.

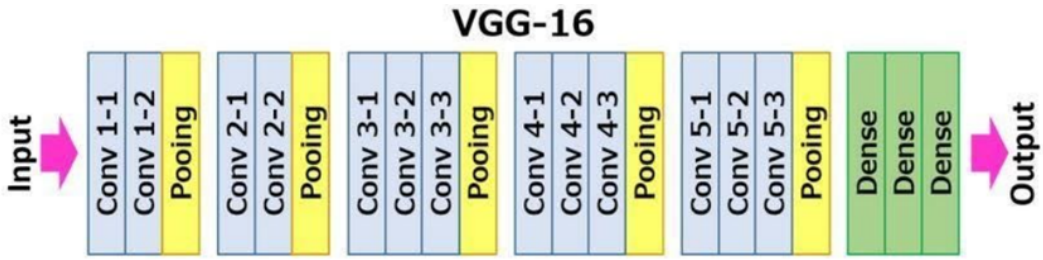

Figure 4 - Showing the VGG16 architecture

The VGG16 model contains 16 learnable parameter layers but a total of 21 layers (13 convolutional layers, 5 max-pooling layers, and 3 dense layers). Each max pooling layer consists of a 2x2 filter with a stride of 2. In figure 4 above, the Conv 1 layers have 64 filters, the Conv 2 layers have 128 filters, the Conv 3 layers have 256 filters, the Conv 4 and Conv 5 layers have 512 filters.

4.4.2 ResNet

The ResNet model which was introduced in 2015 introduced a concept called Residual Network which solves the problem of vanishing/exploding gradient. The vanishing gradient problem occurs when we increase the number of layers which causes the gradient to become 0 or too large. This gradient in turn increases the error while training and testing. The technique of skip connections was introduced in this model which skips some layers and connects earlier layers to later layers hence allowing information to bypass layers. This creates a residual block and a stack of residual blocks were used to create the ResNet model. The purpose of this skip connection is so that if there are layers which will hinder the performance of the model, those layers will be skipped during regularization which solves the problem of vanishing/exploding gradient. An illustration of this architecture is shown below in Figure 5.

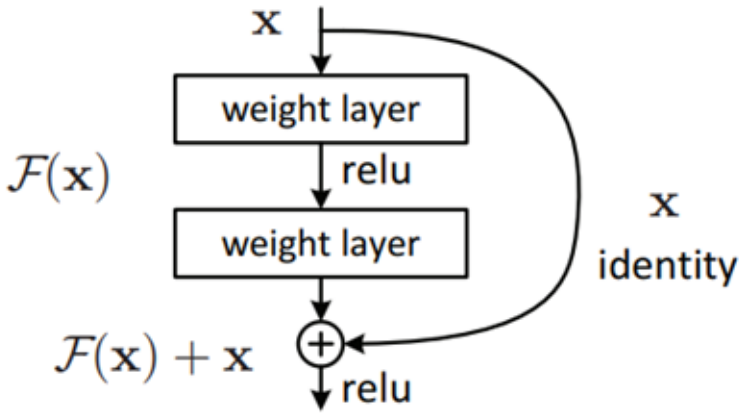

Figure 5 - Showing the ResNet architecture

5. Results/Discussion (to be completed, work in progress)

For evaluation - Accuracy, Sensitivity, Specificity, Precision, F-Measure, ROC curve, Recall

6. Conclusion (to be completed, work in progress)

7. References

Dataset:

https://www.kaggle.com/competitions/aptos2019-blindness-detection/data

8. GitHub Link for Project

Project code and data are available at
https://www.github.com/Girish-Rajani/Blindess_Detection_Deep_Learning