Course Project Report

Diabetic Retinopathy Detection

Submitted By

Abhishek Srinivas (221AI004)
Pranav Moothedath (221AI030)

as part of the requirements of the course

Data Science (IT258) [Dec 2023 - Apr 2024]

in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology in Artificial Intelligence

under the guidance of

Dr. Sowmya Kamath S, Dept of IT, NITK Surathkal

undergone at

DEPARTMENT OF INFORMATION TECHNOLOGY
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA, SURATHKAL

DEC 2023 - APR 2024

DEPARTMENT OF INFORMATION TECHNOLOGY

National Institute of Technology Karnataka, Surathkal

C E R T I F I C A T E

This is to certify that the Course project Work Report entitled **"Diabetic Retinopathy Detection and Classification"** is submitted by the group mentioned below -

Details of Project Group

Name of the Student	Register No.	Signature with Date
Pranav Moothedath	2210511	
Abhishek Srinivas	2210522	

this report is a record of the work carried out by them as part of the course **Data Science (IT258)** during the semester **Dec 2023 - Apr 2024**. It is accepted as the Course Project Report submission in the partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Artificial Intelligence.**

(Name and Signature of Course Instructor)
Dr. Sowmya Kamath S
Associate Professor, Dept. of IT, NITK

D E C L A R A T I O N

We hereby declare that the project report entitled **Diabetic Retinopathy Detection and Classification** submitted by us for the course **Data Science (IT258)** during the semester **Dec 2023 - Apr 2024**, as part of the partial course requirements for the award of the degree of Bachelor of Technology in Artificial Intelligence at NITK Surathkal is our original work. We declare that the project has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles elsewhere.

Details of Project Group

Name of the Student	Register No.	Signature with Date
1. Abhishek Srinivas	221AI004	
2. Pranav Moothedath	221AI030	

Place: NITK, Surathkal
Date: 27 March 2024

Diabetic Retinopathy Detection and Classification

Pranav Moothedath[1], Abhishek Srinivas[2]

***Abstract*— Diabetic retinopathy, a complication of diabetes mellitus that affects the retina, is the leading cause of preventable blindness. Early detection and timely treatment are crucial for preventing vision loss. This paper presents a deep learning approach for the accurate classification and detection of diabetic retinopathy from retinal fundus images. Our proposed method employs convolutional neural networks (CNNs) and the concept of transfer learning to automatically identify the presence and severity of diabetic retinopathy lesions. The CNN architecture is designed to capture intricate retinal features, enabling precise classification into different stages of the disease. The model highlighted in our paper exhibits exceptional performance, outperforming traditional methods and achieving accuracy comparable to expert-level grading in diabetic retinopathy assessment. The proposed system offers easy and cheap detection of diabetic retinopathy which would be easily accessible by the poor and needy.**

***Keywords:* Diabetic Retinopathy, Deep Learning, Convolutional Neural Networks, Medical Imaging, Computer-Aided Diagnosis**

EDIT link of Overleaf project: `https://www.overleaf.com/3878629774rtrgrpdbtdsn#2d8ca1`

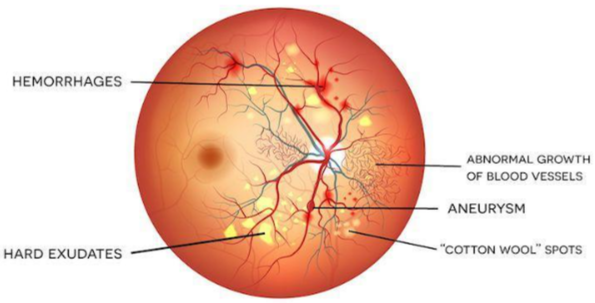


Fig. 1: Diabetic Retinopathic Eye

I. INTRODUCTION

Diabetic retinopathy is a prevalent and potentially vision-threatening complication of diabetes mellitus, affecting millions of people globally. In 2020, the global prevalence of diabetic retinopathy (DR) was estimated to be 103.12 million and is projected to increase to 160.50 million by 2045. A systematic review of 35 population-based studies showed that the prevalence of DR is 34.6% among individuals with diabetes. It is characterized by progressive damage to the retinal line, leading to varying degrees of vision impairment or even blindness if left untreated. Early detection of DR is crucial for timely intervention and effective disease management, thereby preventing irreversible vision loss.

Traditional methods for DR screening and diagnosis involve manual examination of retinal images by trained ophthalmologists. However, this manual process is time-consuming and subject to human error, particularly in resource-constrained settings or when dealing with large patient volumes. Furthermore, the shortage of skilled eye care professionals in many regions worsens the challenges in providing comprehensive and accessible DR screening.

In recent years, deep learning techniques have revolutionized the field of medical image analysis, offering unprecedented accuracy and automation capabilities. Leveraging the powerful feature extraction and pattern recognition abilities of CNNs, researchers have explored their application in DR detection and classification from retinal fundus images.

This study proposes a novel deep learning framework for automated DR detection and classification, aiming to address the challenges associated with manual screening and enable more widespread and efficient DR management. By training CNNs on large datasets of annotated retinal images, our approach learns to recognize the intricate patterns and lesions indicative of five different stages: no diabetic retinopathy, non-proliferative diabetic retinopathy, moderate diabetic retinopathy, severe diabetic retinopathy, and proliferative diabetic retinopathy.

The proposed deep learning system has the potential to revolutionize DR screening by providing an accurate, objective, and scalable solution accessible to healthcare providers worldwide. Early and reliable detection of DR can prompt timely treatment, potentially preventing vision loss and improving the quality of life for countless individuals affected by diabetes.

II. LITERATURE REVIEW

A. Paper 1

One study by Supriya Mishra et al. (Mishra et al., 2020) proposed a vision-based approach using deep learning techniques to detect diabetic retinopathy. The authors trained transfer learning models on a part dataset of the APTOS Challenge. Their preprocessing techniques included cropping and resizing of images, data cleaning, and removing black images and rotation and mirroring of images to balance the dataset.
Two models were chosen to compare between VGG16 and DenseNet121. These models learn relevant visual features and patterns associated with Diabetic retinopathy. During inference, the model analyzed images and classified them

into 5 classes based on the severity of the occurrence of Diabetic Retinopathy. The DenseNet(95%) model with imagenet weights got a way better accuracy than the VGG 16(77%)

B. Paper 2

Another study by Shivani Joshi et al. (Joshi et al., 2023) utilizes CNN encoders and decoders to extract info. Three decoders are used: Classification head, regressions head, and ordinal regression head. Ensemble models, make multiple CNN models, and predict based on taking into consideration the best aspects of each. 89 % accuracy is obtained with the encoder-decoder structure.

C. Paper 3

The third paper reviewed was that of Shri Kant et al. (Hagos and Kant, 2019). They preprocessed image with pixel local average subtraction. They used the Inceptionv3 with all different parameters, by varying optimizer used, loss function, epochs, learning rate, etc. They were compared by keeping an eye on the accuracy of the model on a dataset of images it hadn't seen yet. A brilliant 90.9% accuracy was obtained from the model with augmentations.

D. Paper 4

The fourth paper, (Mayya et al., 2021), our base paper, provides a brief overview of publicly available early diagnosis databases, popular feature extraction techniques, ML classifiers and popularly used DL methods along with information on hyper-parameters. This paper gives us an intro into different types of techniques that have been used by other papers. This paper helped us to identify the strengths and shortcomings of various current approaches so that we could further develop a robust, fully automated framework for DR diagnosis.

III. Dataset

The dataset we chose for our project is Kaggle's diabetic retinopathy dataset, which is freely available. The distribution of images across the different severity levels is as follows:

- No diabetic retinopathy: 25810 images
- Mild diabetic retinopathy: 5292 images
- Moderate diabetic retinopathy: 2443 images
- Severe diabetic retinopathy: 873 images
- Proliferative diabetic retinopathy: 708 images

In total, the dataset comprises 35216 images. This dataset includes pictures of both the left and right eye retinas of the subjects under consideration.

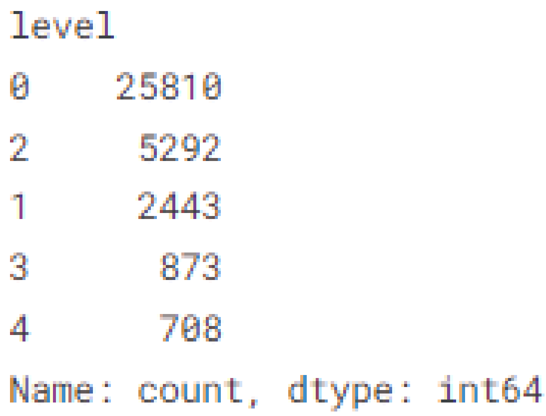


Fig. 2: Diabetic Retinopathic Eye

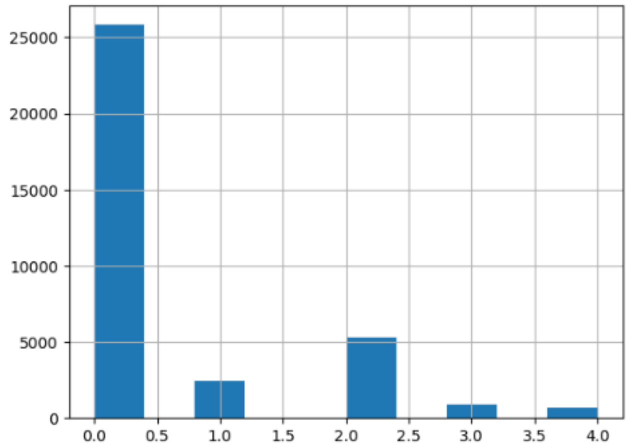


Fig. 3: Class distribution

The figure illustrates examples of retinal fundus images from the dataset, highlighting the varying degrees of diabetic retinopathy lesions present. This dataset is derived from the APTOS 2019 Blindness Detection Challenge hosted on Kaggle, a renowned platform for data science competitions and datasets.

The dataset provides a diverse and representative sample of retinal fundus images, enabling the development and evaluation of robust deep-learning models for the detection and classification of diabetic retinopathy.

IV. METHODOLOGY

The proposed methodology for diabetic retinopathy detection and classification leverages the powerful capabilities of deep learning, specifically convolutional neural networks. The overall approach can be summarized into the following key steps:

A. Data Exploration and Visualization

Before beginning with the creation of the model, we started with exploring and visualizing the dataset of retinal fundus images. This step was aimed to gain insights into the data distribution, identify potential challenges, and inform subsequent design decisions. We see that there is a very high disparity in images between the classes. This is a possible problem to care of, as having a higher number of occurrences of a particular class's image could potentially overfit the model causing it to predict wrongly for classes with fewer instances.

We chose 1 image from each class and applied some visualization techniques to the images. The following techniques were used: warm plot, cool plot, and contour plot. Seeing this we noticed that there was a wide range of value for the pixels in the images.

Fig. 4: Warm Plot

Fig. 5: Cool Plot

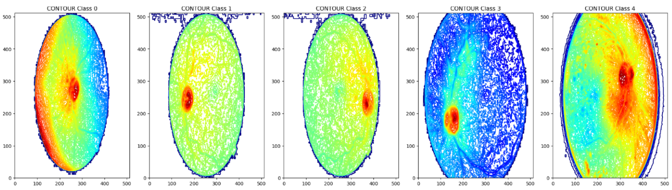


Fig. 6: Contour Plot

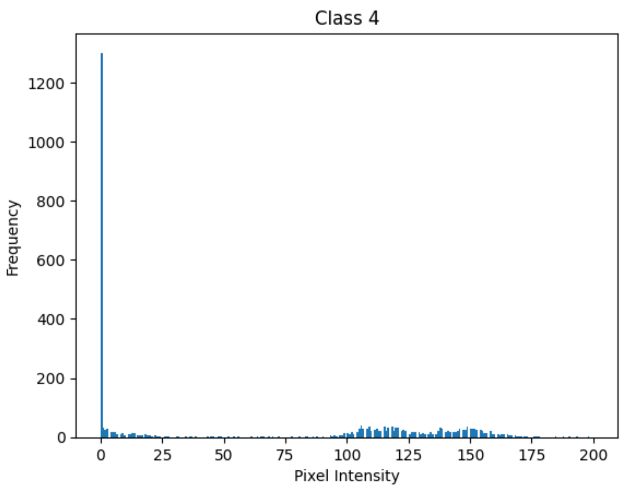


Fig. 7: Pixel distribution

With this observation, we plotted the pixel intensities for each of the 5 images chosen.

From the pixel intensity plot, we see that the majority of the image is composed of the pixel value 0. This is justified as the background of the image presents a majority. Here we note to reduce the image background for all images as they do not present any meaningful information and by cropping the background out, the model can find patterns with better computational efficiency.

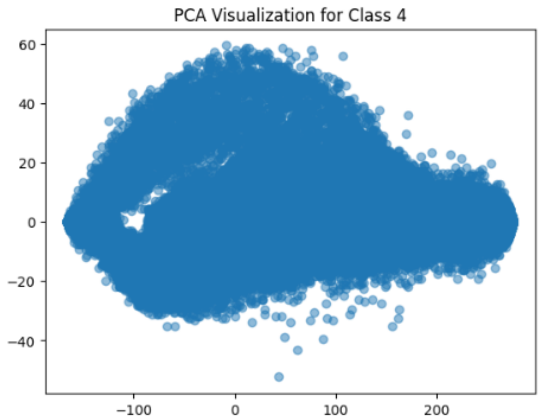
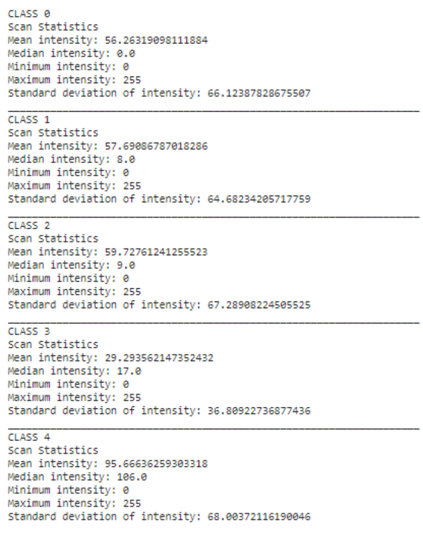


Fig. 8: PCA Visualization

After this, we perform the voxel intensity scatter plot followed by the PCA visualization of the image. The PCA image visualization provides insights into the distribution and structure of the image data in a reduced-dimensional space. It helps in understanding the dominant patterns or features present in the images. It also shows us the direction of maximum variance in the data.



```
CLASS 0
Scan Statistics
Mean intensity: 56.26319098111884
Median intensity: 0.0
Minimum intensity: 0
Maximum intensity: 255
Standard deviation of intensity: 66.12387828675507
____________________________________________________
CLASS 1
Scan Statistics
Mean intensity: 57.69086787018286
Median intensity: 8.0
Minimum intensity: 0
Maximum intensity: 255
Standard deviation of intensity: 64.68234205717759
____________________________________________________
CLASS 2
Scan Statistics
Mean intensity: 59.7276124125552
Median intensity: 9.0
Minimum intensity: 0
Maximum intensity: 255
Standard deviation of intensity: 67.28908224505525
____________________________________________________
CLASS 3
Scan Statistics
Mean intensity: 29.293562147352432
Median intensity: 17.0
Minimum intensity: 0
Maximum intensity: 255
Standard deviation of intensity: 36.80922736877436
____________________________________________________
CLASS 4
Scan Statistics
Mean intensity: 95.66636259303318
Median intensity: 106.0
Minimum intensity: 0
Maximum intensity: 255
Standard deviation of intensity: 68.00372116190046
____________________________________________________
```

Fig. 9: Statistics

Following this we chose to find the min, max, median, mean, and standard deviation of the pixel values in the images. Here we observed that the eyes with Proliferate Diabetic Retinopathy have a significantly high value of median intensity. This goes with the fact that, the presence of aneurysms in the blood vessels in the eyes.

Fig. 10: Cropped Image

B. Preprocessing Techncniques

As noted earlier during the data exploration stage, the black background in all images tends to cause some disturbance while extracting features for classification. To take care of this, we made our preprocessing function crop_image_from_gray. This preprocessing function removes the excess black background from the image and presents it in a zoomed form with only the retina being focused in the image. This would help in the overall detection and classification of the images.

Fig. 11: Gaussian Blurred Image

The next technique tried was Gaussian blur removal. This technique was mentioned in several papers as a good technique to reduce the noise in the image. By smoothing out rapid changes in pixel intensity, Gaussian blur improves the overall quality of the image and makes subsequent processing steps more robust to noise. In addition to noise reduction, Gaussian blur smoothens the edges and details in the image. In addition to smoothening the image, Gaussian Blur also preserves important edge information by averaging neighboring pixels with a Gaussian distribution. Gaussian blur is used to normalize the scale of features in the image, making them more consistent across different images or scenes.

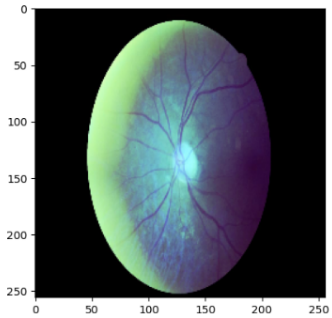


Fig. 12: Histogram Equalized Image

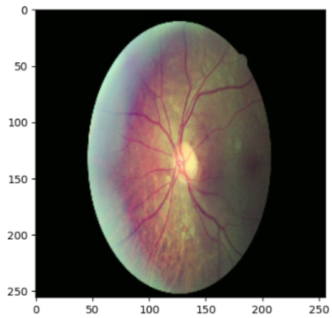


Fig. 13: CLAHE + HE Image

Another set of techniques mentioned in our base paper was that of Histogram Equivalence (HE) and Contrast Limited Adaptive Histogram Equivalence (CLAHE) on the green channel of the image. The human eye is most sensitive to green light, making the green channel crucial for capturing fine details and subtle structures in eye images In medical imaging, particularly in retinal images, the green channel contains vital information about blood vessels and lesions. Further, the green Applying HE and CLAHE to the green channel can help in improving the visibility of these structures by increasing their contrast against the background. We've explored both HE and CLAHE techniques and opted to merge the results of both methods to create a novel preprocessed image. This composite image capitalizes on the strengths of both HE and CLAHE, preserving a wealth of information related to blood vessels and hemorrhages in the eye.

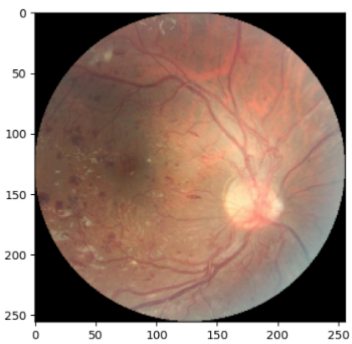


Fig. 14: Circle cropped image

Next, we introduced a unique preprocessing method tailored to address the variability in image shapes. Observing that images varied in shape, with some being circular and others resembling partial circles with sections removed, we recognized that critical information about blood vessels and hemorrhages in the eye predominantly resides within circular regions of the images. Therefore, we devised a technique to crop out circular segments of the images while retaining all pertinent information contained within the original circular regions. This standard in shape would help in pattern finding for the CNN model.

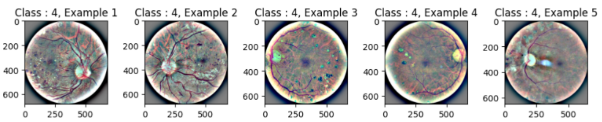


Fig. 15: Gaussian Circle cropped image

Next, we sought to integrate the Gaussian preprocessing technique with the circle crop method, as we believed that combining these approaches could yield synergistic benefits in enhancing the quality and relevance of the preprocessed images.

The results were beneficial, as the new image had a constant circular size as well as high sharpness.

This was the best-preprocessed image we had gotten so far.

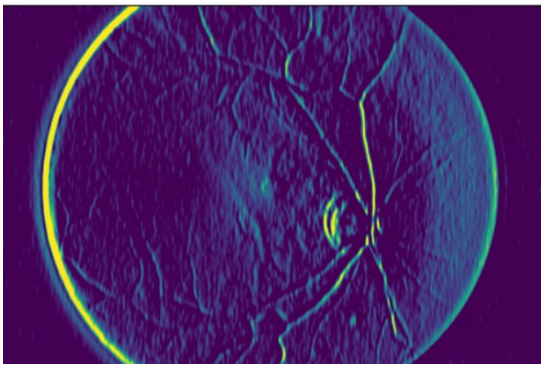


Fig. 16: Gabour Filtered Image

Subsequently, we started with the next preprocessing technique: Gabour Filter. This had been discussed in the paper and was said to be good for images with eye retinas. Gabour filter takes into consideration the following parameters:

sigma - Standard deviation of the Gaussian envelope
lambda - Wavelength of the sinusoidal factor
gamma - Spatial aspect ratio
phi - Phase offset
kernel - Convolution operation kernel

Based on the values passed it produces an image that especially highlights the edges and vessels in the images. It further improves the sharpness of the image.

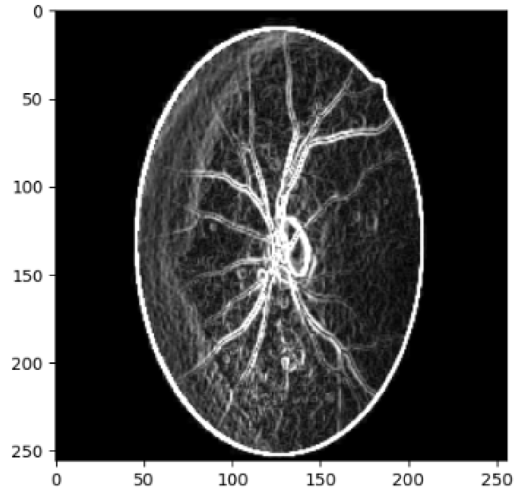


Fig. 17: Krish image

Further, the next preprocessing technique we decided to try was our own creation the Krish cropping method, specially designed to get information from retinal images. In this method, we first extract the green channel from the image, because as discussed before, the green channel is most important for aneurysm detection. Then we utilize 8 different kernels we have specially made to detect certain features/patterns in the image. After this, we convolved the image with these kernels using the Convolution operation. After convolving the image with all 8 kernels, the function extracts the maximum response among all convolutions for each pixel location. This means that for each pixel, the function selects the maximum value across the 8 convolution outputs, effectively highlighting the most prominent features or patterns detected by the different kernels. This method is highly useful for edge detection and provides a brilliant image that highlights the blood vessels present in the image.

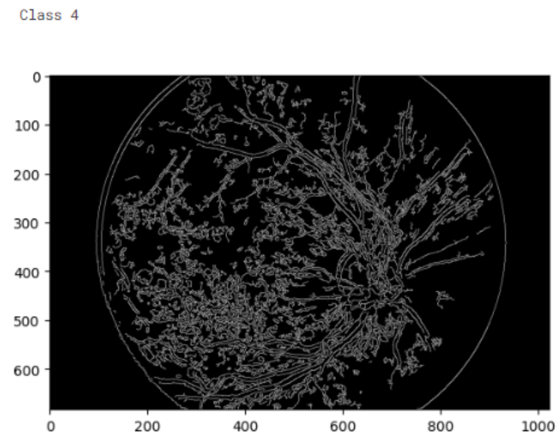


Fig. 18: Canny Edge Detected image

The next method that we tried was that of Canny Edge detection. This technique aims to identify the edges of objects within an image by detecting significant changes in pixel intensity, which often correspond to object boundaries or edges. However, for our images, since the blood vessels in the images are too small to be detected accurately, we get unsatisfactory results for this method.

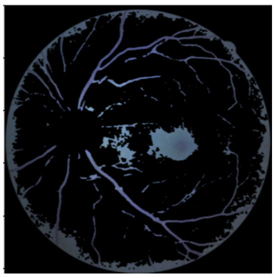


Fig. 19: Vessel Segmentation

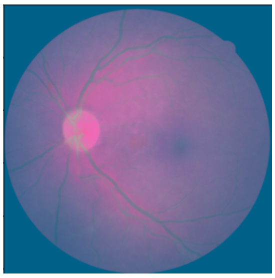


Fig. 20: LUV preprocessing

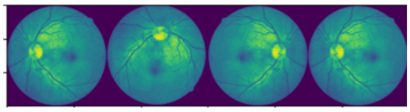


Fig. 21: Augmentations

Further preprocessing techniques were utilized like augmentations, luv preprocessing, color normalization, vessel segmentation, and shade correction. For augmentations, the images are randomly augmented based on parameters passed such as random horizontal flip, vertical flip, and contrast.
Vessel segmentation especially contrasts the blood vessels in the image for better detection and classification.

Regarding the discrepancy in the distribution of the class's images in the dataset, we decided to try three different techniques:

The initial approach involves selecting a fixed number of images from each class, ensuring that the minimum number of occurrences across all classes is accounted for, which amounts to 708 images. Subsequently, we trained the model on training and validation datasets generated from

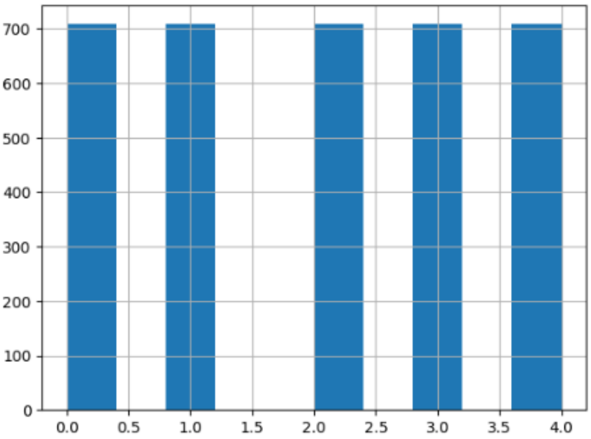


Fig. 22: 708 images per class chosen

this minimized dataset. However, the outcome proved to be disheartening, as the model exhibited a lack of discernible pattern learning when evaluating the images. This was mostly because some information was lost for classes with more than 708 images. This technique is called undersampling,

In our second approach, we tackled class imbalance by oversampling the images. To address this issue, we implemented SMOTE analysis, a technique designed to rectify disparities in class distribution within the dataset. Initially, SMOTE identifies the class with the fewest images, known as the minority class. It then selects a random sample from this minority class and generates synthetic samples by identifying the nearest neighbor images. Additionally, we explored data augmentation techniques such as Random Vertical Flip, Random Angle Rotation, and Random Brightness changes on selected images from the dataset to introduce variability and expand the dataset further. Despite these efforts, we encountered challenges, and unfortunately, they did not yield significant improvements in accuracy.

For our third and final attempt, we employed a more strategic approach to address the class imbalance issue. We randomly selected a subset of 5,000 images from the dataset and maintained the same distribution ratio between the classes as in the original dataset. This step ensured that our training and testing datasets accurately represented the class imbalance present in the real-world scenario.

To do away with the effects of class imbalance and improve the model's performance, we incorporated a specialized loss function x x for imbalanced datasets. This loss function intelligently adjusts the penalty weights based on the class distribution, assigning higher penalties for misclassifications in underrepresented classes and lower penalties for misclassifications in overrepresented classes. By doing so, the model is encouraged to learn more effectively from the minority classes, leading to improved overall accuracy and fairness in predictions.

C. Models

Before sending the images to the models we first made the train and validation generators for loading the images. This method uses image paths from the data frame provided and the file path of the image folder to create a loader of the images.

In addition to loading, we can also preprocess the images. All the images' pixel values are normalized to be between 0 and 1 by dividing the image by 255 (the max value of a pixel is 255, corresponding to white, and the least value is 0, corresponding to black). This is done to get uniformity for the images.

The primary method discussed in numerous papers for detection and classification relies on Deep Learning Models, notably Convolutional Neural Networks (CNNs). However, training a model from scratch is a challenging and time-consuming endeavor, necessitating specialized accelerators like GPUs or TPUs, which were limited in our resources. Therefore, we opted for transfer learning.

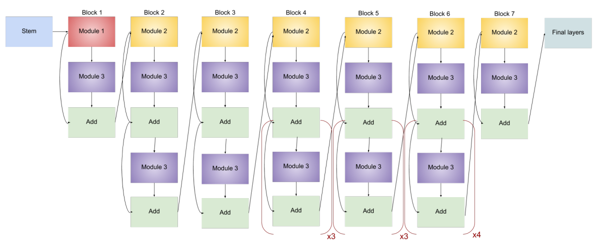


Fig. 23: Efficient Net B3

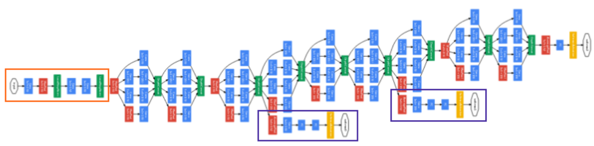


Fig. 24: Inception Net V3

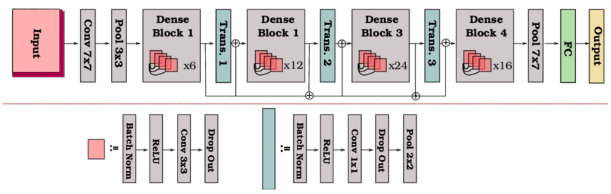


Fig. 25: Dense Net 121

Transfer learning involves leveraging pre-trained deep neural network models developed by organizations with ample resources and utilizing the learned weights from these models. We then further fine-tune these weights to

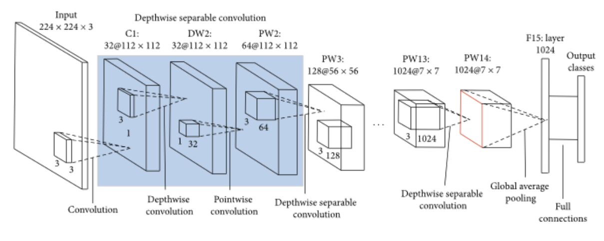


Fig. 26: Mobile Net

adapt the model for classifying the specific task at hand.

Different models have been tried by the various papers mentioned in our base paper. These include AlexNet, VGG16, ResNet, DenseNet, InceptionNetv3, EfficientNetB3, EfficientNetB0. We decided to go with the three most popularly mentioned models including DenseNet, InceptionNetv3, and EfficientNetB3. In addition to this, we have chosen another model architecture MobileNet into consideration.

Altering the hyperparameters such as batch size, and epochs helps us to get good accuracy for our models. Utilising the random images from the dataset we see the following results.

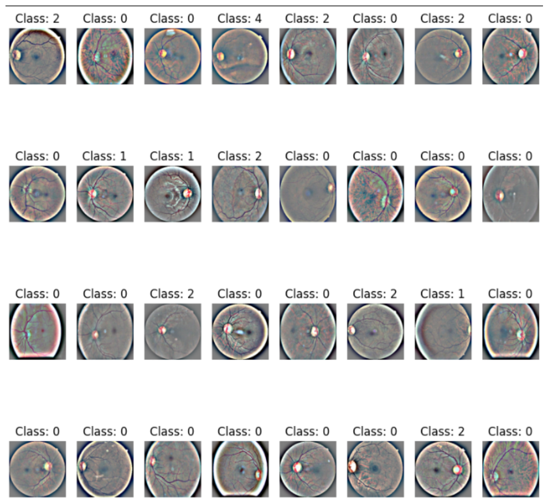


Fig. 27: Data Loader for Gaussian Blur

D. Training

For model training, we have divided the procured dataset into train and validation datasets and have made different generators for each. We use 32 as the batch size for the train and the valid generator.

When you set batch size = 32, it means that the training data will be divided into smaller groups of 32 samples. The neural network will process these 32 samples together, computing the gradients and updating the model parameters based on these 32 samples. This process is then repeated for the next batch of 32 samples until the entire dataset has been processed.

```
data_gen = ImageDataGenerator(rescale=1/255.,
                              zoom_range=0.15,
                              fill_mode='constant',
                              cval=0.,
                              horizontal_flip=True,
                              vertical_flip=True,
                              preprocessing_function=gauss_preprocessing)
```

Fig. 28: datagen with random augments

The generator for the train has some augmentations in place so that it can learn to determine features better, in case it comes across a slightly deformed image.
In the data loaders, in addition to augmentations, we can also pass through any custom preprocessing function we may require.

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
model.fit(train_generator, validation_data = val_generator, epochs = 30)
```

Fig. 29: Compiling and Fitting

We compile the model with sparse categorical cross entropy because the labels are given in the form of integers 0,1,2,3 and 4 in the given CSV file.
The model is made to fit on the train dataset images and the accuracy and loss of the validation dataset is generated to be viewed at the end of each epoch.
If the validation loss is decreasing then we say that the model is learning to detect and classify diabetic retinopathy and its stages respectively.

V. RESULTS

After looking through all the preprocessing techniques, we have decided to utilize the preprocessing techniques including Gaussian Blur, Krish Cropping, Gabor Filter, and Gaussian Blur with Circle Crop. We utilize the models MobileNet, DenseNet, InceptionNetv3, and Efficient B3 with each of the preprocessing techniques chosen.

When all images are passed through directly just with normalization MobileNet and DenseNet give the most accuracy.

```
Epoch 10/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 53s 403ms/step - accuracy: 0.7578 - loss: 0.5768 - val_accuracy: 0.7350 - val_loss: 0.9398
Epoch 11/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 396ms/step - accuracy: 0.7553 - loss: 0.6234 - val_accuracy: 0.7330 - val_loss: 0.9819
Epoch 12/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 51s 393ms/step - accuracy: 0.7572 - loss: 0.6052 - val_accuracy: 0.7370 - val_loss: 1.0829
Epoch 13/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 400ms/step - accuracy: 0.7746 - loss: 0.5136 - val_accuracy: 0.7370 - val_loss: 1.0770
Epoch 14/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 398ms/step - accuracy: 0.7652 - loss: 0.5009 - val_accuracy: 0.7360 - val_loss: 1.1850
Epoch 15/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 53s 401ms/step - accuracy: 0.7562 - loss: 0.5100 - val_accuracy: 0.7310 - val_loss: 1.0643
Epoch 16/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 400ms/step - accuracy: 0.7640 - loss: 0.4808 - val_accuracy: 0.7030 - val_loss: 1.0406
Epoch 17/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 400ms/step - accuracy: 0.7657 - loss: 0.4900 - val_accuracy: 0.7390 - val_loss: 1.2534
Epoch 18/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 400ms/step - accuracy: 0.7874 - loss: 0.4246 - val_accuracy: 0.7100 - val_loss: 0.9507
Epoch 19/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 401ms/step - accuracy: 0.7783 - loss: 0.4962 - val_accuracy: 0.7100 - val_loss: 1.1200
Epoch 20/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 395ms/step - accuracy: 0.7766 - loss: 0.4458 - val_accuracy: 0.6710 - val_loss: 1.1336
```

Fig. 30: Gauss in InceptionV3

```
Epoch 10/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 53s 408ms/step - accuracy: 0.7405 - loss: 0.9705 - val_accuracy: 0.7360 - val_loss: 0.8889
Epoch 11/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 81s 401ms/step - accuracy: 0.7406 - loss: 0.9144 - val_accuracy: 0.7360 - val_loss: 0.8819
Epoch 12/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 82s 400ms/step - accuracy: 0.7585 - loss: 0.8589 - val_accuracy: 0.7360 - val_loss: 0.8664
Epoch 13/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 401ms/step - accuracy: 0.7509 - loss: 0.8787 - val_accuracy: 0.7360 - val_loss: 0.8611
Epoch 14/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 400ms/step - accuracy: 0.7568 - loss: 0.8470 - val_accuracy: 0.7360 - val_loss: 0.8766
Epoch 15/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 53s 406ms/step - accuracy: 0.7463 - loss: 0.8807 - val_accuracy: 0.7360 - val_loss: 0.8994
Epoch 16/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 399ms/step - accuracy: 0.7340 - loss: 0.9119 - val_accuracy: 0.7360 - val_loss: 0.9115
Epoch 17/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 398ms/step - accuracy: 0.7502 - loss: 0.8758 - val_accuracy: 0.7360 - val_loss: 0.8809
Epoch 18/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 398ms/step - accuracy: 0.7485 - loss: 0.8718 - val_accuracy: 0.7360 - val_loss: 0.8856
Epoch 19/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 53s 405ms/step - accuracy: 0.7532 - loss: 0.8633 - val_accuracy: 0.7360 - val_loss: 0.8823
Epoch 20/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 52s 399ms/step - accuracy: 0.7575 - loss: 0.8437 - val_accuracy: 0.7360 - val_loss: 0.8751
```

Fig. 31: Gauss in EfficientNetB3

```
Epoch 10/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 61s 464ms/step - accuracy: 0.7726 - loss: 0.5775 - val_accuracy: 0.7400 - val_loss: 0.9598
Epoch 11/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 59s 452ms/step - accuracy: 0.7653 - loss: 0.5505 - val_accuracy: 0.7390 - val_loss: 0.8840
Epoch 12/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 60s 461ms/step - accuracy: 0.7545 - loss: 0.5630 - val_accuracy: 0.7310 - val_loss: 0.8893
Epoch 13/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 60s 460ms/step - accuracy: 0.7667 - loss: 0.5281 - val_accuracy: 0.7360 - val_loss: 1.0691
Epoch 14/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 61s 463ms/step - accuracy: 0.7857 - loss: 0.4886 - val_accuracy: 0.7420 - val_loss: 0.9485
Epoch 15/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 60s 461ms/step - accuracy: 0.7979 - loss: 0.4673 - val_accuracy: 0.7350 - val_loss: 0.9835
Epoch 16/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 82s 464ms/step - accuracy: 0.8116 - loss: 0.4433 - val_accuracy: 0.7460 - val_loss: 0.9747
Epoch 17/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 62s 472ms/step - accuracy: 0.7849 - loss: 0.4892 - val_accuracy: 0.7170 - val_loss: 0.9903
Epoch 18/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 61s 470ms/step - accuracy: 0.8132 - loss: 0.4206 - val_accuracy: 0.7110 - val_loss: 1.1235
Epoch 19/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 61s 465ms/step - accuracy: 0.7953 - loss: 0.4514 - val_accuracy: 0.7370 - val_loss: 1.5168
Epoch 20/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 61s 470ms/step - accuracy: 0.7324 - loss: 0.5470 - val_accuracy: 0.7290 - val_loss: 1.0314
```

Fig. 32: Gauss in DenseNet

```
Epoch 10/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 61s 464ms/step - accuracy: 0.8542 - loss: 0.3205 - val_accuracy: 0.7180 - val_loss: 1.3074
Epoch 11/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 61s 467ms/step - accuracy: 0.8676 - loss: 0.3142 - val_accuracy: 0.7480 - val_loss: 2.2206
Epoch 12/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 82s 469ms/step - accuracy: 0.8797 - loss: 0.2835 - val_accuracy: 0.7330 - val_loss: 1.6577
Epoch 13/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 60s 457ms/step - accuracy: 0.8824 - loss: 0.2695 - val_accuracy: 0.7440 - val_loss: 1.7776
Epoch 14/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 60s 456ms/step - accuracy: 0.8656 - loss: 0.3191 - val_accuracy: 0.7290 - val_loss: 1.3013
Epoch 15/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 59s 451ms/step - accuracy: 0.8151 - loss: 0.4066 - val_accuracy: 0.7410 - val_loss: 1.6523
Epoch 16/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 59s 454ms/step - accuracy: 0.8494 - loss: 0.3186 - val_accuracy: 0.7200 - val_loss: 1.3519
Epoch 17/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 60s 459ms/step - accuracy: 0.8504 - loss: 0.3204 - val_accuracy: 0.7400 - val_loss: 1.4506
Epoch 18/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 60s 454ms/step - accuracy: 0.8802 - loss: 0.2755 - val_accuracy: 0.7360 - val_loss: 1.5104
Epoch 19/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 60s 457ms/step - accuracy: 0.8853 - loss: 0.2648 - val_accuracy: 0.7550 - val_loss: 1.8323
Epoch 20/20
125/125 ━━━━━━━━━━━━━━━━━━━━ 81s 453ms/step - accuracy: 0.8876 - loss: 0.2674 - val_accuracy: 0.7490 - val_loss: 2.2324
```

Fig. 33: Guass in MobileNet

Upon passing Gaussian preprocessed normalized images through the models we see the following training parameters.

InceptionV3: 77% train 67% val
DenseNet121: 73% train 73% val
EfficientNetB3: 75% train 73% val
MobileNet: 89% train 75% val

As can be seen from the results on the next page, the best accuracy is obtained by MOBILENET ARCHITECTURE along with Gaussian blur preprocessing.

MobileNet achieves high efficiency and accuracy by leveraging depthwise separable convolutions, width and resolution multipliers, and a carefully designed architecture tailored for mobile and embedded applications.
Further, MobileNet offers flexibility through hyperparameters like the width multiplier and resolution multiplier, making it more robust and efficient compared to other models.
Since MobileNet and DenseNet give the best accuracies among the rest of the models, we decided to use them for training the Gabor filter, circle crop, and Krish crop. The following are the results:

Maximum accuracy is got by GAUSSIAN BLUR CIRCLE CROP with MOBILENET of: 94% ACCURACY.

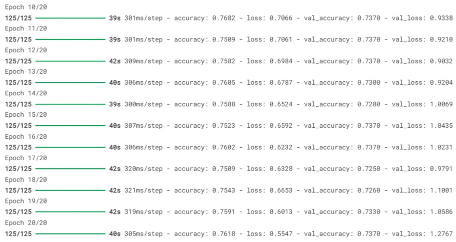


Fig. 34: Gabor Filter in Dense

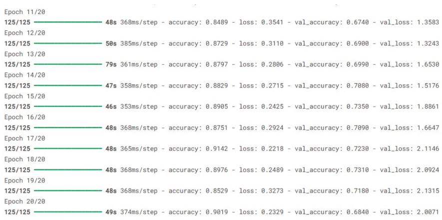


Fig. 35: Gabor Filter in MobileNet

Fig. 36: Gauss Circle Crop in MobileNet

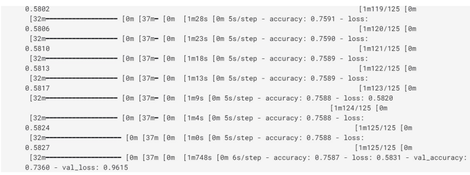


Fig. 37: Krish Crop in DenseNet

VI. CONCLUSIONS

By combining Gaussian blur and circle cropping and using it as an overall preprocessing technique for MobileNet architecture, we achieved an architecture with excellent results, outperforming our base paper. Moving forward, we aim to enhance accuracy and address the class imbalance issue present in the dataset.

Our CNN-based model offers a portable and cost-effective solution for diabetic retinopathy screening, making it accessible to the masses. It leverages deep learning algorithms and computer vision to analyze retinal images and accurately detect signs of diabetic retinopathy.
Moreover, widespread adoption of this technology can significantly reduce the strain on healthcare resources, enabling efficient allocation of funds and personnel towards treatment and prevention efforts.

REFERENCES

Hagos, M. T. and Kant, S. (2019). Transfer learning based detection of diabetic retinopathy from small dataset.

Joshi, S., Kumar, R., Rai, P. K., and Garg, S. (2023). Diabetic retinopathy using deep learning. In *2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*, pages 145–149.

Mayya, V., Kamath S, S., and Kulkarni, U. (2021). Automated microaneurysms detection for early diagnosis of diabetic retinopathy: A comprehensive review. *Computer Methods and Programs in Biomedicine Update*, 1:100013.

Mishra, S., Hanchate, S., and Saquib, Z. (2020). Diabetic retinopathy detection using deep learning. In *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, pages 515–520.

APPENDIX

Add the first page of Plagiarism Report here, after I provide the report to you (must have less than 15% similarity). Each team member should add their signature on the report page

project_report .pdf

ORIGINALITY REPORT

7%	4%	2%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Aston University Student Paper	1%
2	Submitted to Imperial College of Science, Technology and Medicine Student Paper	1%
3	www.dellemc.com Internet Source	1%
4	docs.oracle.com Internet Source	1%
5	Submitted to CSU, San Jose State University Student Paper	1%
6	Submitted to UNITEC Institute of Technology Student Paper	1%
7	docplayer.net Internet Source	<1%
8	www.i-scholar.in Internet Source	<1%