Predicting the Severity of Diabetic Retinopathy Disease Using Retinal Scans

Brendan Brasch, David Ho

Abstract

Diabetic retinopathy (DR) is an eye disease associated with long-standing diabetes and currently a leading cause of vision loss within working-age adults. Early detection and treatment of this disease is crucial for improving patient outcomes.

In this paper, we explore several methods for predicting the severity of diabetic retinopathy from retinal scans using k-means clustering and deep learning. We evaluate our method on a dataset of retinal images and show that pretrained convolutional neural networks can predict the severity of diabetic retinopathy with reasonable accuracy. Our results suggest that our approach could be a useful tool for early detection and treatment of the disease.

I. Introduction & Background

Diabetic Retinopathy

DR is a condition that occurs when high blood sugar levels cause damage to the small blood vessels in the retina. There are two main types of DR: nonproliferative diabetic retinopathy (NPDR) and proliferative diabetic retinopathy (PDR). In NPDR, the blood vessels in the retina become weakened and tiny bulges, called microaneurysms, may occur and leak blood or fluid. This can cause swelling of the retina and vision loss. PDR is a more severe form of diabetic retinopathy and characterized by new abnormal blood vessel growth, also known as retinal neovascularization, on the optic disc or elsewhere on the retina. These new blood vessels are often weak and prone to leakage and bleeding, known as retinal hemorrhages when the bleeding is in the retina or preretinal hemorrhage when there is bleeding in the vitreous. The bleeding and fluid build up in the retina can block the light that is needed to see clearly, resulting in vision loss. Additionally, the damage caused by high blood sugar levels can lead to the growth of abnormal blood vessels and scar tissue in the retina, known as fibrovascular lesions. These lesions can distort the retina and also cause blindness[1].

Diagnosis of Diabetic Retinopathy

Early detection and treatment is important to prevent vision loss. DR is typically diagnosed by a trained clinician through a comprehensive dilated eye exam, where the pupil is enlarged for a thorough examination of the retina. Alternatively, the clinician can use imaging tests, such as optical coherence tomography, fluorescein angiography, or fundus photography, for a detailed view of the retina and search for any abnormalities.

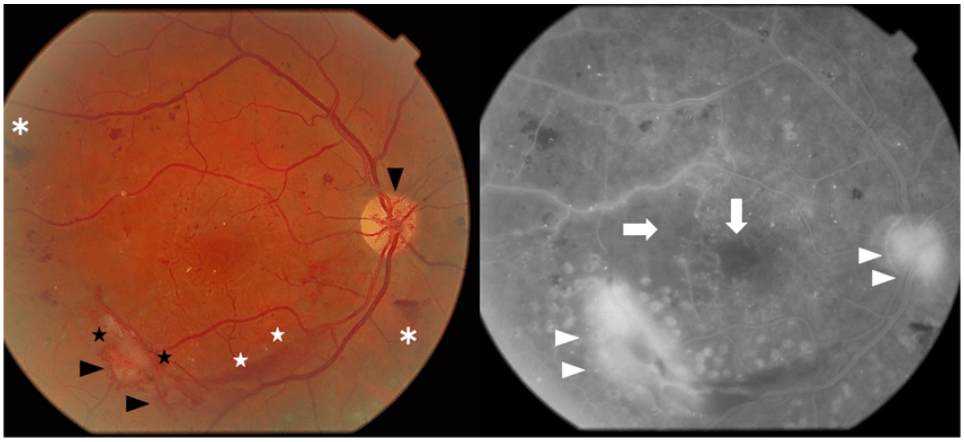


Figure 1. Clinical features of proliferative diabetic retinopathy (PDR)[2].

Figure 1 is a combination of two photos, a color fundus photograph (left) and a fundus fluorescein angiogram (right), of a retina affected by PDR. Preretinal hemorrhage are marked by white stars, retinal hemorrhages by white asterisks, retinal neovascularization with black arrowheads, fibrovascular lesions by black stars, areas of retinal ischemia by white arrows, and vascular leakage from retinal new vessels with white arrowheads.

Regular DR screening is important to detect DR at an early stage to provide treatment and prevent progression to PDR. However, specialized equipment and trained clinicians to interpret the results are required and resource intensive. However, using machine learning algorithms, it is possible to provide accurate and fast diagnoses while reducing the need for specialized equipment and personnel, allowing for more people to detect and start treatment for DR earlier[2].

K-Means Clustering

K-means clustering is a simple unsupervised algorithm used to divide samples into *k* clusters, where each sample is sorted into the cluster with the "closest" mean. To accomplish this goal, the cluster centers are defined randomly (or intuitively if information regarding the clusters is already known). Each sample is then grouped into the nearest cluster using some distance metric, often the L2 or Euclidean distance. Then, new cluster centers are calculated as the mean of the samples in each cluster. This process of reassignment of samples and recalculation of the means is repeated until cluster assignments stabilize.

K-means clustering has several notable limitations that should be considered during implementation. In the context of image classification, it is important to note that the Euclidean distance between two images can vary dramatically with noise and image artifacts. This can lead to incorrect clustering across a dataset with a high amount of noise and variation[3]. One strategy to address this limitation is image normalization, which may reduce the effect of noise across the whole image. However, this strategy does not address specific artifacts within the image, so normalized images belonging to the same true cluster may still have a relatively high L2 distance.

Deep Learning

Deep learning is a specific variety of machine learning that employs artificial neural networks to extract raw information from an input and interpret high-level features without the need for manual identification. A neural network can then use this high-level interpretation to make decisions and accomplish a specific task. These networks are built as a series of interconnected layers that feed information to different layers in the network. Different use cases often have specific network architectures that are best suited to solve the relevant problem.

Each node contained within the layers of a network has a weight attached to each input channel and a bias. To compute the output of each node, the weighted sum of the input channels is added to the bias, and this result is passed through an activation function, producing the output. The model's backpropagation algorithm calculates the gradient of a specified loss function at the node and updates the weights and biases using gradient descent.

Convolutional Neural Networks

Convolutional neural networks (CNNs) describe a type of network architecture used to analyze image data in the field of deep learning. Each convolutional layer enables a network to "learn" image features by interpreting pixel values in the context of its neighboring pixels. Alone, convolutional layers are only able to identify simple features such as lines, but convolutional layers joined together in series allow each layer to "learn" increasingly complex features from the image data[4].

In the context of image classification, CNNs often employ a series of convolutional layers coupled with batch normalization and pooling layers to normalize and reduce image data throughout the network in order to extract high-level features. The outputs from these layers are often fed through one or more dense layers which allow the network to make decisions from the identified features. These tools are the backbone of image classification using deep learning.

II. Data & Approaches

Data Description

We utilized a public dataset of 35,108 retina images of patients exhibiting various stages of DR. Each image was labeled by a clinician on a scale of 0 to 4 describing the stage of DR according to this scale:

0 - No DR
1 - Mild DR
2 - Moderate DR
3 - Severe DR
4 - Proliferative DR

Importantly, the images within the dataset were taken under a variety of conditions (including different cameras), so there is a large amount of variability between images. The images may also contain noise and artifacts such as poor focus and exposure. Several of the images are inverted, or flipped horizontally, which affects the locality of the macula and optic nerve within

each image[5]. We will be working with a resized version of this dataset where black space around the images has been cropped out in order to standardize the size of the fundus image[6]. It is important to note that due to the noise and variability of the dataset, methods which rely on features having a consistent locality within the images will likely be ineffective.

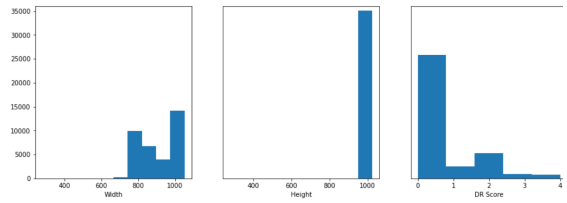


Figure 2. Histograms showing the width (left), height (center), and label (right) occurrences in the dataset.

Analysis of the dataset (Figure 2) shows the ranges of image widths, height, and DR scores. Image widths appear to vary largely between 700 to 1024 pixels, and most image heights are exactly 1024 pixels. About 75% of the dataset is labeled 0 for no detected DR, and each class of detected DR has a relatively small occurrence, indicating that data balancing strategies (such as class weighing) may be necessary for training.

Data Preprocessing

Before we began building classifiers, images needed to undergo several preprocessing steps. The first preprocessing step in our k-means clustering approach was to either crop or resize each image to the desired resolution. Images were converted to grayscale to reduce computational complexity after early exploration did not show any significant performance improvement by including the three color channels in each image. Images were then rescaled from 0 to 1.

For our deep learning approach, we similarly resized image inputs to the desired resolution. We elected not to convert images to grayscale as we could not determine whether this preprocessing step would impact performance. The images underwent several data augmentation steps to diversify the training set (these steps were not included in the k-means clustering preprocessing workflow due to the algorithm's dependence on consistent locality of features). This augmentation included randomly flipping the image and rotating it a random value between -90° and 90°. Any pixels left empty from the rotation were filled with zeros. Lastly, images were rescaled from 0 to 1.

K-Means Clustering Approach

We begin our analysis by implementing k-means clustering under the context of our dataset. Five clusters representing the five classes of DR in our dataset were initialized by choosing a random image from each class to act as the initial mean of the cluster. Importantly, k-means clustering is unsupervised, so there is no guarantee that the five returned clusters would correspond to the DR labels in the expected order. Therefore, we employed a confusion matrix to let us analyze the homogeneity of each cluster, as opposed to relying on the expected labeling of the cluster which may have changed during training.

Since k-means clustering is helped by consistent locality of features in image data,

we expected that this approach would be unlikely to yield an accurate classification. However, we still chose to implement this method to demonstrate the complexity of the image dataset.

Naive Deep Learning Approach

We then implemented a naive neural network to perform image classification. We specify that this is a naive approach as it is constructed solely with dense layers of decreasing size, which are eventually fed into an output layer. Dropout layers were included after each dense layer to reduce overfitting of the training data. The number of dense layers, nodes per dense layer, and dropout parameters were varied in between training runs in order to identify functional networks. The output layer was defined as either a softmax function of five nodes, or a sigmoid function to enable categorical classification or binary classification respectively (binary classification representing 0 - heathy or [1, 2, 3, 4] - DR in this case).

Since this network's trainable layers are all dense layers, this network is also dependent on consistent locality of image features, similarly to the k-means approach. Therefore, we would expect that it is unlikely this network could train to be an accurate classifier. However, we decided to include this implementation to demonstrate the complexity of our dataset, and to validate the need for a more complex network architecture.

Convolutional Neural Network Approach

We then implemented a more sophisticated CNN to classify the images. In this case, images are fed into a series of convolutional layers. As convolutional layers get deeper, the number of filters increases to detect more abstract features from the image data. Max pooling was used in between convolutional layers to reduce image size and allow the convolutional layers to detect features across a larger area deeper in the network. After the convolutional layers, a global average pooling layer was used to feed forward the remaining image data while reducing network complexity. Then, a series of dense layers were added to allow the network to learn from identified features from the convolution layers. Dropout layers were included after each dense layer to reduce overfitting of the training data. The output of the dense layers are then fed to the relevant output of the network to allow categorical classification or binary classification.

We expect that this network, being specifically designed for image classification, would yield a better classification than any previously discussed approaches. However, there remains the possibility that the network may still not provide accurate classifications, which would indicate the need for a more complex architecture or a larger training dataset.

Pretrained CNN Approach

To evaluate the performance of our designed classifiers, we implemented an architecture to utilize another pre-built architecture trained on the ImageNet dataset for object

classification. We used this strategy to implement several high-performing image classification networks including InceptionV and DenseNet121. To adapt these networks to our classification problem, the final convolutional layers of the prebuilt architecture were fed into a small number of dense layers to interpret the identified features. Dropout layers were included after each dense layer to reduce overfitting of the training data. Again, the output of the dense layers are then fed to the relevant output of the network to allow categorical classification or binary classification.

III. Results

K-Means Clustering Approach

As expected, this approach yielded a very poor clustering accuracy without any improvement over random cluster assignment. Specifically, the clustering algorithm clustered images into the five clusters with equal probability regardless of the image's true class. These results were consistent for both of the tested resolutions (128x128 and 256x256). A representative confusion matrix is shown in Figure 3 to demonstrate the probability of each clustering assignment given the image's actual class.

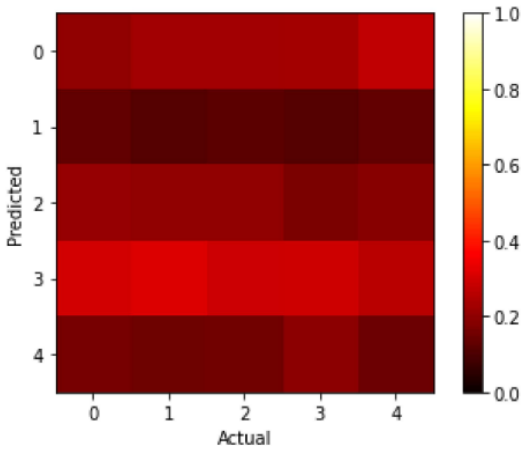


Figure 3. Confusion matrix visualizing clustering distribution for each class after k-means clustering.

This approach was fairly time efficient on smaller resolutions, but increasing the resolution of images exponentially increased the runtime. The 128x128 resolution images were clustered in 471 seconds on average (~9 seconds per iteration, ~53 iterations per run on average). The 256x256 resolution images were clustered in 3,952 seconds on average (~42 seconds per iteration, ~ 93 iterations per run on average).

Hyperparameter Selection

Each of our tested networks were evaluated under a large variety of hyperparameters and variations of architectures (i.e. number of nodes, number of hidden layers, number of convolutional filters, convolutional filter size, activation functions, etc.) in order to identify functional networks. For each deep learning approach, the best performing specific network architecture was evaluated. Additionally, the set of hyperparameters that enabled the best network performance were used for general accuracy and timing analysis between the different approaches.

Due to the large memory requirements of some of the networks, a batch size of 8 was selected. Varying this batch size had minimal effect on the long-term convergence of each network, although increasing the batch size caused memory errors for some network architectures. The networks were optimized using stochastic gradient descent with an initial learning rate of 1e-4, which enabled the quickest convergence of the networks. It became evident that this problem required a dynamic learning rate as the initial learning rate didn't always converge to a high enough accuracy. Therefore, a callback was added to decrease the learning rate by a factor of 5 upon a plateau in validation loss (defined as a decrease of validation loss of less than 1% from the previous epoch). Classes were weighted to account for the large class imbalance in the dataset. An image resolution of 256x256 was selected for analysis as this resolution was high enough to enable successful identification of DR-structures in the images, but it was low enough to enable relatively efficient network training. Lastly, 10 epochs were selected for training as this allowed for network convergence while enabling efficient training.

All network training was performed on an NVIDIA GeForce RTX 2060 Max-Q GPU using cuDNN 8.1.

Naive Deep Learning Approach

As expected, the naive deep learning approach also yielded minimal success at DR classification. Over several variations of network size and hyperparameters, this network architecture converged to the same result, only a single class predicted across most (or all) of the images. When observing the raw softmax prediction of the network, it becomes clear that all classes are predicted with nearly equal likelihood, and a miniscule increase in likelihood for any one class was observed consistently throughout the dataset. Therefore, the accuracy of this network was simply the proportion of the single predicted class within the dataset, which varies between 0.025 and 0.75. A representative confusion matrix is shown in Figure 4 demonstrating the network's tendency to predict a single class.

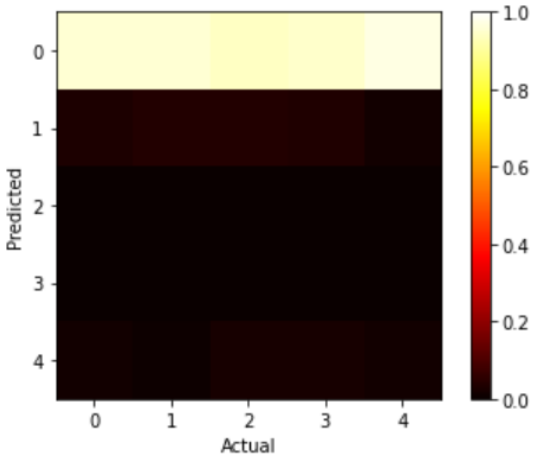


Figure 4. Confusion matrix visualizing prediction accuracy and distribution for each class by the trained naive neural network.

This network trained relatively efficiently, completing 10 epochs in 1,237 seconds (~124 seconds per epoch). Validation loss converged after only 2 epochs for this network.

Convolutional Neural Network Approach

Despite this network's architecture which was suitable for image classification, this

approach similarly yielded minimal success at DR classification. Similarly to the naive approach, this network was unable to learn any differences between the classes, and ended up predicting exclusively one of the five classes at convergence. The softmax output of this network similarly showed roughly equal probabilities for all classes. A representative confusion matrix is shown in Figure 5 to illustrate this network's tendency to predict a single class.

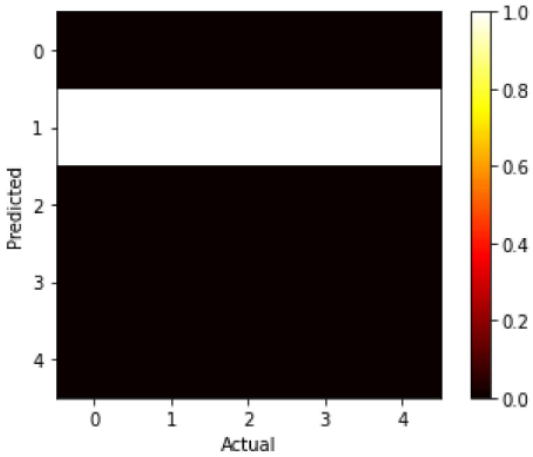


Figure 5. Confusion matrix visualizing prediction accuracy and distribution for each class by the trained convolutional neural network.

This network trained relatively efficiently, completing 10 epochs in 979 seconds (~98 seconds per epoch). Validation loss converged after 8 epochs for this network.

Pretrained CNN Approach

As expected, the InceptionV3 and DenseNet121 pretrained network approaches were able to classify DR with a significantly higher accuracy than random assignment.

The InceptionV3 model architecture was able to yield the following prediction accuracies on validation data:

Class	**Prediction accuracy**
0 - No DR	0.4592
1 - Mild DR	0.3990
2 - Moderate DR	0.2629
3 - Severe DR	0.6915
4 - Proliferative DR	0.7183

Overall, the InceptionV3 model architecture was able to predict with a validation accuracy of 0.4131, with notably increased accuracy for severe and proliferative DR. The confusion matrix is shown in Figure 6. This network trained relatively slowly, completing 10 epochs in 5,245 seconds (~524 seconds per epoch).

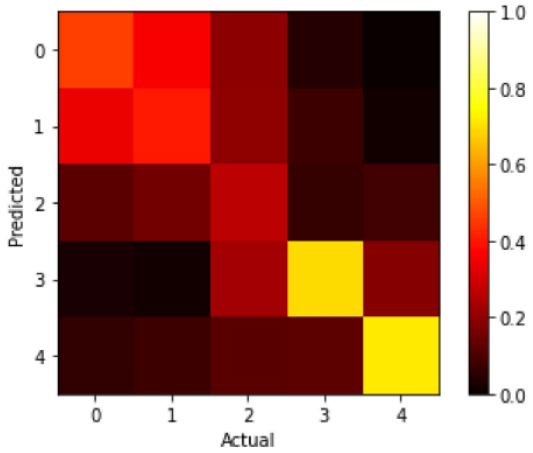
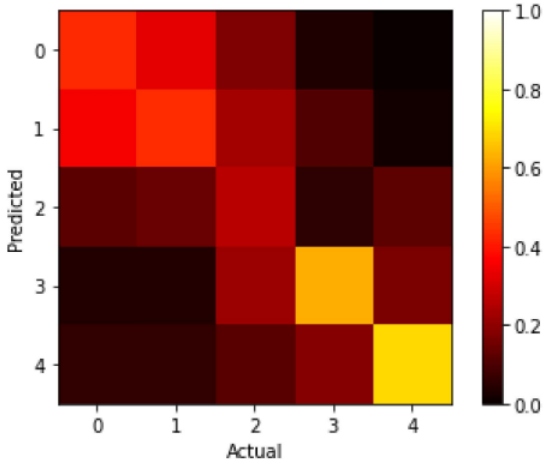


Figure 6. Confusion matrix visualizing prediction accuracy and distribution for each class by the trained InceptionV3 neural network.

The DenseNet121 model architecture was able to yield the following prediction accuracies on validation data:

Class	Prediction accuracy
0 - No DR	0.4258
1 - Mild DR	0.4327
2 - Moderate DR	0.2610
3 - Severe DR	0.6277
4 - Proliferative DR	0.6901

Overall, the DenseNet121 model architecture was able to predict with a validation accuracy of 0.4077, with notably increased accuracy for severe and proliferative DR. The confusion matrix is shown in Figure 7. This network trained relatively slowly, completing 10 epochs in 7,212 seconds (~721 seconds per epoch).



Figue 7. Confusion matrix visualizing prediction accuracy and distribution for each class by the trained DenseNet121 neural network.

IV. Discussion and Future Directions

Overall, we had mixed success in developing models to classify DR. Our initial approach through k-means clustering failed to yield any predictive value superior to random assignment. Upon manual review of the clusters, it became apparent that this approach had a tendency to form clusters based on image artifacts as opposed to the degree of DR. For example, one training run produced a cluster of largely oversaturated images, and another cluster of largely undersaturated images, regardless of any image's true labeling (figure 8). Therefore, it is unlikely that a k-means clustering approach could ever demonstrate robust predictive accuracy for classification of DR due to its dependence on consistently located image features and its sensitivity to noise and artifacts.

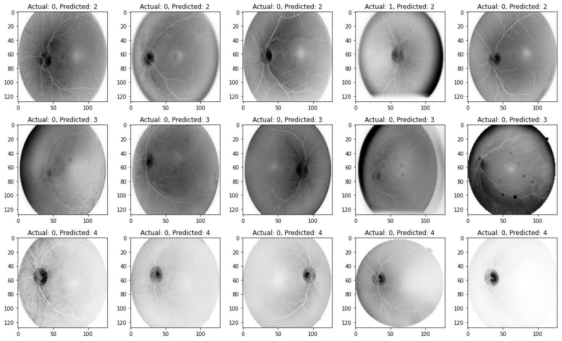


Figure 8. Random images sampled from 3 clusters, with each row representing a different cluster generated using k-means clustering. The middle cluster has many undersaturated images, while the last cluster has many oversaturated images, demonstrating how this method tends to cluster by artifacts.

Unsurprisingly, a similar result is seen from our naive deep learning classifier. Since this method is supervised (as opposed to the

unsupervised k-means clustering algorithm), this model is more robust to noise and artifacts since they are present throughout each of the known classes. However, this model produces near equal softmax probabilities between each of the five classes, indicating that the network is unable to identify any features that would allow it to intelligently make predictions. This indicates that this model is either too simple for the task at hand, or that the model architecture is unsuitable for image classification. Since our dataset is small but diverse in terms of feature locality, this naive network is likely unsuitable for this problem and would be unlikely to demonstrate robust predictive accuracy for the classification of DR.

Interestingly, our tested CNN architecture behaved very similarly to our naive neural network. The CNN architecture is designed to be sensitive to noise and artifacts, and it isn't dependent on consistent locality of image features. However, this model also produces near equal softmax probabilities between each of the five classes under a large variety of training conditions. This may indicate that all of the tested CNN architectures were too simple to identify relevant features in the data. Alternatively, there may be additional architecture techniques that could improve this model's predictive ability.

As expected, the pretrained network architectures for InceptionV3 and DenseNet121 significantly outperformed our manually designed networks. This is likely due to the large and intricate network architecture. InceptionV3 is described as a very deep neural network due to its large number of convolution layers. It also contains improvements on the original Inception network by leveraging suitably factorized convolutions and aggressive regularization[7]. The DenseNet121 architecture is based on the principle that convolutional networks can be more accurate and train more efficiently if there exist shorter connections between early and late layers in the network[8]. This differs from the traditional serial CNN design used in our network. Importantly, this architecture is specifically designed to alleviate the vanishing-gradient problem in which a CNN quickly converges to an inaccurate local minima and loses the ability to train further, which may have been a source of error for our CNN architecture. Therefore, the deep convolutional layers and intelligent design of network architecture enabled the InceptionV3 and DenseNet121 to classify diabetic retinopathy more accurately than our k-means clustering, naive, and CNN approaches.

Going forward, there are several steps that could be taken to further investigate neural networks for the classification of DR. Since our designed CNN was not able to accurately predict DR severity, it would be advantageous to compare the size of the network architecture to an established CNN architecture One could investigate how much these deep layers contribute to predictive accuracy be iteratively adding convolutional layers to the CNN following the architecture principles outlined by InceptionV3 or DenseNet121. This may

allow for identification of exactly how complex CNN network architecture must be to identify features from this DR dataset.

Importantly, this dataset was labeled for DR severity by a single clinician. In order to develop a DR classification system into a useful clinical tool, the training dataset would likely need to be much larger and include labeling by several different clinicians to represent a real-world ground truth and to capture the variability in diagnosis of DR severity.

V. Conclusion

There are several techniques that could be applied to the classification of DR severity based on retina images. To build a useful classifier, one must consider a classification strategy's robustness to noise and artifacts in images which could otherwise be detrimental to classification accuracy. One promising approach to solving this problem is taking advantage of pretrained image classification networks such as InceptionV3 or DenseNet121 and retraining them on a DR image dataset. While certain classification approaches are promising, there are still several major steps to take before this could serve as a useful clinical tool.

References

[1] L. Dai et al., "A deep learning system for detecting diabetic retinopathy across the disease spectrum," *Nature Communications*, vol. 12, no. 1, p. 3242, May 2021, doi: 10.1038/s41467-021-23458-5.

[2] M. Mesquida, F. Drawnel, and S. Fauser, "The role of inflammation in diabetic eye disease," *Seminars in Immunopathology*, vol. 41, no. 4, pp. 427–445, Jun. 2019, doi: 10.1007/s00281-019-00750-7.

[3] Liwei Wang, Yan Zhang, and Jufu Feng, "On the Euclidean distance of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1334–1339, Aug. 2005, doi: 10.1109/tpami.2005.165.

[4] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, Jun. 2018, doi: 10.1007/s13244-018-0639-9.

[5] E. Dugas, J. Cukierski, J. Cukierski, and W. Cukierski, "Diabetic Retinopathy Detection," *Kaggle*. https://www.kaggle.com/c/diabetic-retinopathy-detection (accessed Dec. 16, 2022).

[6] "Diabetic Retinopathy (resized)," *Kaggle*. https://www.kaggle.com/datasets/tanlikesmath/diabetic-retinopathy-resized (accessed Dec. 16, 2022).

[7] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv.org*, Sep. 2014, doi: 10.48550/arXiv.1409.1556.

[8] G. Huang, Z. Liu, L. van der Maaten, and K. Weinberger, "Densely Connected Convolutional Networks," *arXiv.org*, Aug. 2016, doi: 10.48550/arXiv.1608.06993.