Analysis of Diabetic Retinopathy Fundus Imagery using Convolutional Neural Networks

Liam Tingley
Jodrey School of Computer Science
Acadia University
Wolfville, Canada
161568t@acadiau.ca

I. INTRODUCTION AND PROBLEM STATEMENT

Diabetic retinopathy (DR) is a prevalent eye condition threatening partial or total loss of vision in patients if left untreated. The condition develops when blood vessels in the retina begin to leak into the vitreous, a gel-like substance which fills the centre of the eye. As bleeding becomes more severe, spots and streaks will become more obvious on the eyes as blood builds up and hardens inside the vitreous [1]. Though the condition can develop in individuals without diabetes, those who are diabetic, of any type, are at particular risk of vision loss if the condition is not properly diagnosed and treated quickly.

Thus, we are presented with the problem of swiftly and correctly detecting the presence of diabetic retinopathy. For the purposes of this project, a convolutional neural network, (CNN), will be developed and deployed for classifying fundus image data. For a successful model to be developed, we will look to minimize the false negative (FN) classification rate, as incorrectly concluding that diabetic retinopathy is absent in a patient who does possess it could have severe consequences for the patient and their vision.

II. BACKGROUND

Convolutional neural networks are deep learning models composed of a series of layers typically organized into the following structure: input layers, hidden layers, and output layers. Input layers normally consist of Convolution layers, of which there can be multiple. These layers contain most of the complex computation involved with a neural network - in particular, algebraic matrix operations such as matrix multiplication. Input layers possess kernels, also referred to as filters, that serve as two-dimensional weight arrays known as feature detectors. These filters comb over data, examining small subsections of the data, (the images in the case of this study), in pursuit of emergent patterns.

Next are the hidden layers, which may include pooling layers which serve to reduce the dimensionality of the input after undergoing computational processing.

Finally, the output layer, also referred to as the fully-connected layer, is composed of nodes that directly connect to at least one node in the preceding layer. As output layers are used to return the network's conclusion or prediction regarding the classification of the passed data sample, its activation function is typically represented in a probabilistic manner, with each node within it generating a value between 0 and 1 [2].

Given the vast array of combinations of hyperparameters and architecture configurations available for convolutional neural networks, we will seek to explore a collection of hyperparameter adjustments and their influences on model performance.

Convolutional neural network models are frequently utilized for multidimensional predictive modeling. As such, image data classification is a common use case for CNNs. The use of a CNN model in fundus image classification is convenient, and lends itself to high performance in an area where accurate classification is critical.

The entirety of this project's image preprocessing and model development was conducted in a Google Colaboratory environment. The system on which this environment was utilized was a Dell Latitude 5420 containing an Intel Core i5 processor operating at 2.40GHz, with 16GB of RAM at the system's disposal.

III. METHOD AND APPROACH

A. Project Plan

As previously mentioned, this project will utilize a convolution neural network (CNN) that will be fed a total of 2,838 images across the training, validation, and testing phases. Prior to modeling however, these images must undergo a series of preprocessing measures in order to ensure that they are properly prepared to contribute to the development of the CNN model in a uniform and consistent manner.

Image preprocessing details and reasoning are described in greater detail in *B. Metadata Report.*

Following the image preprocessing phase, the modeling phase will formally commence. There will be research and experimentation conducted into the preparation of the model, including determination of hyperparameters, and whether further image preprocessing is necessary. CNN modules from the Keras Python library will be employed due to their relative implementation simplicity and reasonably high performance.

Once a series of candidate models have been generated, a performance metric, the mean false negative classification count, will be extracted from the base model as well as the generated alternatives, and will be compared against one another in the form of a t-test. Evaluation details are described in greater depth in *D. Evaluation.*

B. Metadata Report

The diabetic retinopathy dataset is composed of 2,838 total images that are sorted into one of two collections within each set: those being one in which images are deemed to possess DR, and the other deemed not to contain DR. These images are presorted into a training set containing 2,076 images, a validation set containing 531 images, and a testing set containing 231 images. Every image is 224 pixels in height, and 224 pixels in width. Prior to any preprocessing measures being enacted on the images, the images possessed a reasonably low level of contrast. Some possessed particularly low light levels. The images were also coloured, with most possessing a dark pink/red hue.

There were several preprocessing measures taken in order to prepare the image data for modeling. These were grayscale conversion, brightness and contrast adjustments, and normalization. The first of these, grayscale conversion, was undertaken to reduce computational complexity. This is because colour differentiation is a dimension that can be eliminated during the conversion from RGB format.

The second preprocessing measure was brightness and contrast adjustments. Following the grayscale conversion, the resulting images were very dark, presenting potential challenges for feature detection in the modeling phase. To avoid this scenario, both the brightness and contrast were increased to accentuate the features present in the images. The final measure taken in the preprocessing phase was to normalize the images. This step was employed to ensure that all images contributed in a uniform and equal manner to the model. This was done by normalizing all image values into the range [0,1], ensuring uniform contribution to the modeling algorithms employed.

C. Modeling Solution

The initial model has been adapted from a tutorial developed by Jaz Allibhai at towardsdatascience.com [3]. It possesses a Sequential structure, due to its ease of use as part of the Keras deep learning library. It consists of two two-dimensional Convolution layers, one containing 64 nodes, the next containing 32 nodes. These values were selected due to their initially positive performance. Furthermore, it is good practice to reduce the number of nodes in a given Convolution layer to half the number of nodes in the preceding layer. Both layers also contain a ReLU activation function due to its proven performance within neural networks. As discussed by Dishashree Gupta of Analytics Vidhya, the ReLU "does not activate all the neurons at the same time [4]." This lends itself well to enhanced computational efficiency, as only the relevant neurons of the layer are activated.

The Flatten layer serves as a connecting layer between the Convolution layers, and the Dense layer serves as the output layer. The Dense layer is a type of layer that is typical for output layers in neural networks, hence its inclusion here. It contains two output nodes, due to the binary classification nature of the problem and dataset.

In a similar vein, for the compilation phase of modeling, the sigmoid function is used as the activation function. This function is useful for outputting results as it is not symmetric around 0, but rather has an entirely positive range. This is important when returning predictions in a probabilistic manner, as these values must be in a range between 0 and 1. Similarly, the sigmoid function presents the opportunity to observe the confidence with which a model makes a prediction about a particular test case, as its returned probabilities will be skewed more intensely toward one of the two classification possibilities depending on its confidence in its prediction.

Another component included in the compilation phase is the optimizer, which will be Adam for the base model, as well as the alternatives. Adam is utilized for its high computation speed, as well as the nature of the dataset that it will be handling. It serves as an extension of stochastic gradient descent by "dynamically adjusting learning rates based on individual weights [5]."

The component of the compilation phase is the binary cross entropy loss function. This function is chosen given the nature of the binary classification problem that is being approached. It is used to quantify the distance between a correct prediction and an incorrect prediction during model training, and penalizing the model accordingly for incorrect predictions [6].

The first alternative to the base model that was considered was the simplification of the model architecture to just one single two-dimensional Convolution layer. In addition to exploring potential performance enhancements, this architecture has the added benefit of serving as a preventative measure for overfitting by simplifying its construction [7]. This model maintained 64 nodes in its solitary Convolution layer, as well as the presence of a single Flatten layer and Dense layer. It also utilized the same ReLU activation functions in the Convolution layer, and sigmoid activation function in the Dense layer.

A third model consisted of an architecture much the same as the base model, with a max pooling layer inserted between the two two-dimensional Convolution layers. This was done as pooling layers are frequent solutions for reducing the dimensionality and feature maps of input data, while simultaneously decreasing variance. Further alterations to the original model included the increase of the kernel size of the first Convolution layer from 3x3 to 5x5. This was done so as to enable the network to examine training images for broader patterns, before undergoing the dimensionality reduction in the pooling layer, and examining the images in finer detail with the second Convolution layer, which maintains the kernel size of 3x3 from the base model.

The pooling layer configuration takes inspiration from the configuration of the max pooling layer of research conducted by Mishkin et al. [8], who observed that two max pooling layers with size 3x3 and padding 1 achieved an almost 2% accuracy gain over the control model. As such, an adapted solution was implemented into this alternative model with a pooling size of 3x3, and padding appropriately incorporated.

D. Evaluation

Prior to examining evaluation metrics, we must establish a collection of assumptions and constants that are being put in place to cultivate reproducible results, while also simplifying the process of generating alternative models. The first of these, reproducibility, is achieved by utilizing a fixed seed value specified prior to each run of each model. This is accomplished using functions existing within Keras' own library, ensuring the same pseudorandom numbers are generated for each model, eliminating unpredictable randomness as a possible influence on model performance.

In addition, the number of epochs, i.e., the number of times the training and validation sets are iterated over per run of a given model, has been fixed at 5. This decision is supported by the peak in validation accuracy achieved at epoch 4 (the fifth overall epoch) illustrated in Figure A which is followed by an erosion in validation accuracy.

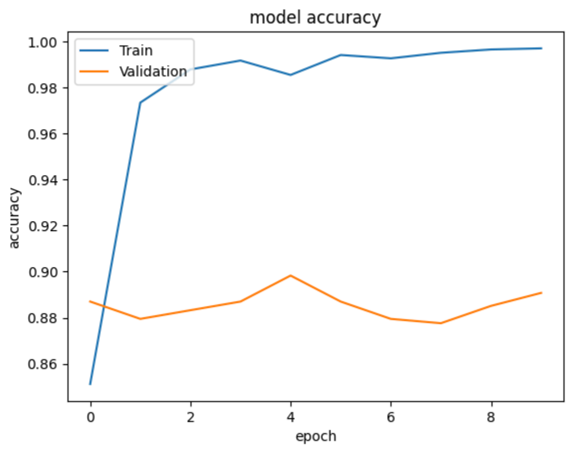

Fig. A. Showcases the peak of validation accuracy (depicted in orange) at the fifth epoch.

Figure A was obtained using a preliminary run of the base model, using a seed value of 1, and 10 epochs. This ensures that each model will comb through the training and validation sets an equal number of times, eliminating exposure levels to the data as a possible contributor to variable performance.

As introduced in I. INTRODUCTION AND PROBLEM STATEMENT, a successful model will minimize its quantity of False-Negative (FN) classifications. This is due to the nature of the domain in which the model operates. Misidentifying a patient as having no diabetic retinopathy present, when in reality DR is present (a false negative result), could lead to partial or complete vision loss in the patient. The method of comparison will be to examine the lowest mean average quantity of FN classifications for each model across five runs. Once the lowest two averages of the three have been obtained, their results will be compared using a difference of means t-test. Should the resulting p-value be less than the predetermined significance level, a conclusion will be drawn that there is a statistically significant performance difference between the models. In the event that the generated p-value is greater than the significance level, further examination of additional metrics will be explored. Several other metrics may be relevant for describing, and thus differentiating, the success or performance levels of a model. These include testing classification accuracy, recall, precision, and F1 score, all of which will be examined in greater detail in IV. RESULTS.

IV. RESULTS

TABLE I
Average False Negative Classification Count

Seed	Base Model	1 Conv. Layer	Max Pooling	p-value
1	10	12	9	0.815
2	10	12	9	
3	11	13	12	
4	12	12	11	
5	8	11	11	
Avg.	10.2	12	10.4	

As previously discussed, our primary success metric is the quantity of False-Negative (FN) classifications produced by each model, with a lower count being more successful. The training set consisted of 231 images, with the base model achieving an average of 10.2 false negatives per run. By comparison, the third model, which introduced max pooling, achieved a comparable 10.4 false negative per run. The second model, with a single Convolution layer, achieved 12 false negatives per run. TABLE I above outlines each model's FN classification rate. Thus, we elect to observe the base model and third models in greater detail with relation to one another. Figure B illustrates the confusion matrix of the best single run performance achieved by the base model.

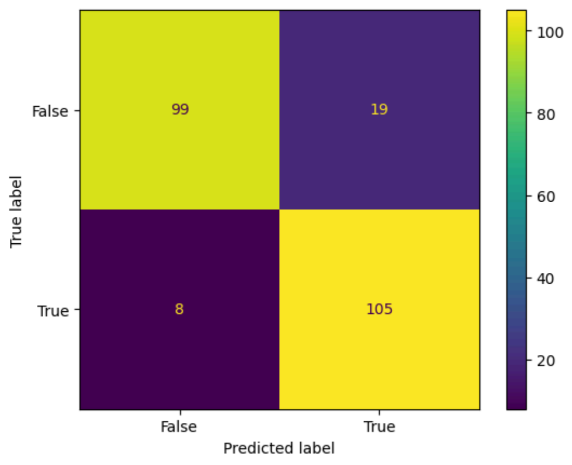

Fig. B. Showcases best single run confusion matrix of base model.

Meanwhile, Figure C illustrates the confusion matrix of the best single run performance achieved by the max pooling model based on the number of false negative classifications identified.

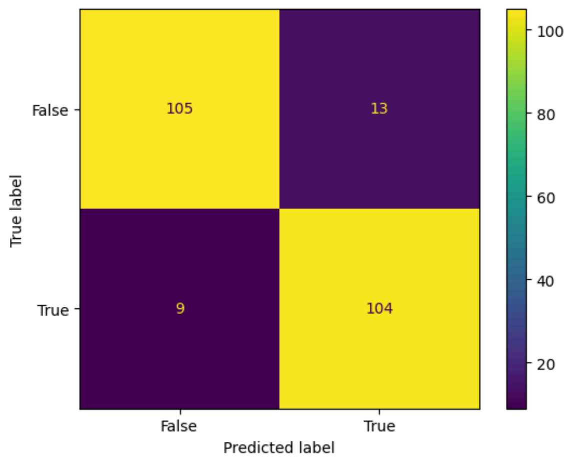

Fig. C. Showcases best single run confusion matrix of the max pooling model.

Upon conducting a difference of means t-test, with a significance level of 0.05, using the base model and the max pooling model, we obtained a p-value of approximately 0.815, as indicated by TABLE I. This value is significantly higher than the p-value required to conclude a statistically significant difference between the models' performance was present. As such, further examination was required. It is worth mentioning that a t-test was also conducted comparing the base model with the single Convolution layer model. The outcome of that test indicated statistical significance between the models.

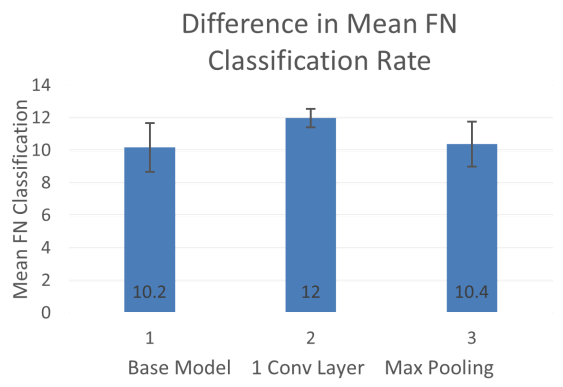

Fig. D. Showcases model false negative classification rate with standard deviation.

The base model achieved an average validation accuracy of 88.55%, with a peak of 89.83%, and a testing accuracy of 88.74%, peaking at 90.48%. By comparison, the max pooling model achieved a slightly better 89.91% average validation accuracy, peaking at 90.02% twice, and an average 89.78% testing accuracy, peaking at 90.48%. As both of these sets of metrics are also quite similar, confusion matrices were constructed and analyzed to observe the precision, recall, and computed F1 score for each model across their five runs.

The first of these values, precision, is meant to indicate the "quality of positive predictions [9]." That is to say, of those who have been diagnosed with diabetic retinopathy, which subjects were correctly identified as actually possessing the condition. It is calculated by obtaining the number of true positive classifications, those who were correctly identified as having diabetic retinopathy, divided by the sum of true positive and false positive classifications.

$$Precision = \frac{TP}{TP+FP}$$

Upon examining this metric, it can be seen that average precision obtained by the base model was 86.77%, with a maximum value of 90.18%, and a minimum of 84.43%. Meanwhile, the max pooling model obtained an average precision of 88.64%, peaking at 90.99%, and achieving a minimum of 86.44%.

The second metric we will observe is recall, which measures the portion of DR-possessing patients who were correctly identified as having possessed it. This metric is calculated by dividing the quantity of true positive classifications, by the sum of the false negative and true positive counts.

$$Recall = \frac{TP}{TP+FN}$$

The base model achieved an average of 90.97% in this regard, with a maximum of 92.92% and a minimum of 89.38%. By comparison, the max pooling model achieved an average of 90.80%, achieving a maximum of 92.04% twice, and a minimum of 89.38%.

The final of these metrics, F1 score, is the harmonic mean of precision and recall. This metric is quite useful for problems such as diabetic retinopathy image classification, as it seeks to identify the performance of the model with regards to the positive class, in our case, that diabetic retinopathy is present. By contrast, this also gives us insight into the accuracy with which the models classify instances of the negative class - that diabetic retinopathy is not present. It can be calculated using the following equation:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

With this in mind, the base model achieved an average F1 score of 88.79%, with a maximum of 90.35% and a minimum of 87.55%, while the max pooling model achieved an average of 89.69%, peaking at 90.43% and bottoming out at 88.31%.

Thus, with all of these metrics presented, the max pooling model was deemed to be the better performing of the two models. While it achieved a slightly worse 10.4 average false negative classifications over its 5 runs compared to the base model's 10.2 average false negative classifications, the t-test proved this was a statistically insignificant difference. By examining other metrics such as accuracy as well as those derived from confusion matrices, it is observable that the max pooling model consistently, albeit marginally, performs better in all of these metrics, and therefore should be more reliably performant. TABLE II provides a summary of the results regarding the performance of the two models in these metrics.

TABLE II
Base Model vs. Max Pooling Model Extended Performance Summary

Average	**Base Model**	**Max Pooling**
Testing Accuracy	88.75%	89.78%
Precision	86.77%	88.64%
Recall	90.97%	90.80%
F1 Score	88.79%	89.69%

V. DISCUSSION

The best performing model, deemed marginally to be the max pooling model, achieved an average of 10.4 false negative classifications over 5 runs of the model using 231 testing samples. This equates to approximately 4.50%, which for practical purposes is likely too high a rate to be considered for deployment at present. For context, related work conducted by Bajwa et. al. [10] produced a model that obtained a false negative classification rate of 0.5%. That is, just 2 false negatives out of a testing set of 398 images. It is important to note however that the model used in the paper was trained and validated using a set of 57,625 images, compared to the comparatively small 2,838 images used to develop the models here.

Amidst the development of the three models presented here, there were several challenges that presented themselves. Initially, issues arose with the usage of PyTorch as data type mismatches proved a recurring issue, ultimately forcing a switch to the Keras library due to time constraints. Additional exploration of this project's models is possible using alternative libraries and systems, such as PyTorch, and may yield differing results for comparison and analysis.

Following the shifting of modeling systems, additional challenges were presented when attempting to ensure the consistency of input shapes for each level of the model, as well as during compilation and fitting phases. As such, data had to be reformatted on several occasions within the implementation to provide the model with the fundus imagery in the required shape. This issue extended to the construction of confusion matrices as well, as the data was previously in a listed format, to simplify the process of generating model predictions, but had to be adjusted to a single integer (0 for DR not present, 1 for DR present) in order to produce the confusion matrices.

Through these challenges came success, as the achieved modeling accuracy, considering the dataset size, time constraints, and extent of preparation methods, was much higher than expected. Additionally, the platform provided to conduct this research has lent itself in a positive manner to enhancing the knowledge base surrounding the implementation and optimization of convolutional neural networks for image classification.

VI. CONCLUSION

The best-performing model developed as part of this project provides a platform of respectable performance. However, to entirely satisfy the success criteria, the model would need to reduce its false negative classification rate to a lower value, ideally to such an extent that the value is routinely 0. There are several facets of the project that can be explored in greater depth to optimize its performance further.

First and foremost, exploring the influence of learning rate as a layer hyperparameter could yield significant performance gain. Learning rate specifies the permissible change in the values of weights as part of the gradient descent algorithm employed by a model [11]. In the case of the models developed in this project, both the base model as well as the alternatives utilized the default learning rate provided by the Adam optimizer: 0.001.

Further exploration of modeling structures and architectures with regards to layering and node count are worth consideration. Deeper networks are more typical constructions of higher performing CNN models - the extent of which could be fascinating to investigate. Should a larger, comparably balanced data set be obtained or made available, exploring deeper networks, and those consisting of greater volumes of nodes are worth exploring. This was previously not entirely possible due to the risk of overfitting and generalizability suffering as a result.

With regards to obtaining larger and higher quality datasets, further preprocessing measures may be considered in order to elevate model performance. Obtaining additional test samples is possible through various artificial augmentation methods including skewing, shearing, rotating, and flipping images.

Thus, with these acknowledgements made, it is advisable that further development of these models might include the exploration of adjustments to the learning rate hyperparameter for accelerated training, and increased classification accuracy, which, by extension, may produce lower false negative classification counts.

Furthermore, returning to the preprocessing phase to heighten the quality of the dataset through augmentation, as evidenced by Bajwa et. al. [10], significant performance improvements are possible with larger datasets.

REFERENCES

[1] National Eye Institute, “Diabetic Retinopathy | National Eye Institute,” Nih.gov, 2019. https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/diabetic-retinopathy.

[2] IBM, “What are Convolutional Neural Networks? | IBM,” www.ibm.com, 2023. https://www.ibm.com/topics/convolutional-neural-networks.

[3] E. Allibhai, “Building a Convolutional Neural Network (CNN) in Keras,” Medium, Nov. 15, 2018. https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5

[4] D. Gupta, “Activation Functions | Fundamentals Of Deep Learning,” Analytics Vidhya, Jan. 29, 2020. https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/.

[5] A. Gupta, “A Comprehensive Guide on Optimizers in Deep Learning,” Analytics Vidhya, Oct. 07, 2021. https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers.

[6] S. Saxena, “Binary Cross Entropy/Log Loss for Binary Classification,” Analytics Vidhya, Mar. 03, 2021. https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/.

[7] A. Sagar, “5 Techniques to Prevent Overfitting in Neural Networks,” KDnuggets, Dec. 06, 2019. https://www.kdnuggets.com/2019/12/5-techniques-prevent-overfitting-neural-networks.html.

[8] D. Mishkin, N. Sergievskiy, and J. Matas, “Systematic evaluation of convolution neural network advances on the Imagenet,” Computer Vision and Image Understanding, vol. 161, pp. 11–19, Aug. 2017, doi: https://doi.org/10.1016/j.cviu.2017.05.007.

[9] “Confusion Matrix, Precision, and Recall Explained,” KDnuggets. https://www.kdnuggets.com/2022/11/confusion-matrix-precision-recall-explained.html

[10] A. Bajwa, Neelam Nosheen, Khalid Iqbal Talpur, and S. Akram, “A Prospective Study on Diabetic Retinopathy Detection Based on Modify Convolutional Neural Network Using Fundus Images at Sindh Institute of Ophthalmology & Visual Sciences,” Diagnostics, vol. 13, no. 3, pp. 393–393, Jan. 2023, doi: https://doi.org/10.3390/diagnostics13030393.

[11] Jason Brownlee, “Understand the Impact of Learning Rate on Neural Network Performance,” Machine Learning Mastery, Jan. 24, 2019. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/.