Scott Song

Diabetic Retinopathy Detection in CFPs using CNNs

Introduction:

The US Center for Disease Control and Prevention estimates that 34.2 million people in the US have diabetes and the World Health Organization estimates that 422 million people have diabetes worldwide. Diabetic retinopathy is an eye disease associated with long-standing diabetes and is the leading cause of blindness in the working-age population of the developed world. Around 40% of Americans with diabetes have some stage of the disease, and it is estimated to affect over 93 million people worldwide. Progression of vision impairment can be significantly hindered if diabetic retinopathy is detected early on; however, detection is inherently difficult as the disease often shows few symptoms until it is too late to provide effective treatment.

Currently, detecting diabetic retinopathy is a time-consuming and manual process that involves a retina specialist evaluating color fundus photographs of the retina based on the presence of lesions associated with the vascular abnormalities caused by the disease. The delay necessitated by a manual diagnosis can lead to lost follow up, miscommunication, and delayed treatment. Furthermore, the expertise and equipment required for diagnosis is lacking in areas where the rate of diabetes in local populations is high, where DR detection is most needed. As the number of individuals with diabetes continues to grow to epidemic proportions, the current infrastructure in place for DR diagnosis will become even more insufficient. Given the pandemic size of the diabetic population, a comprehensive, automated method of diabetic retinopathy screening/detection will be necessary to mitigate the rising threat of the disease.

Objective:

The objective of this project is to train a convolutional neural network that can determine the presence of diabetic retinopathy in color fundus photographs of the retina, a binary classification task.

Dataset:

Part of the original dataset can be found here. Another part of the original data set comes from the Bay Eyecare Institute. The original data set contains high-resolution CFP retina images, and a left or right field is provided for every subject. no diabetic retinopathy with each class representing a distinct stage of DR. Each photo corresponds to one of five classes: Class 0 corresponds to no diabetic retinopathy, Class 1 corresponds to mild, non-proliferative DR, Class 2 corresponds to moderate, non-proliferative DR, Class 3 corresponds to severe, non-proliferative DR, and Class 4 corresponds to proliferative DR. The stage of DR, assigned manually by retina specialists, is determined based on several key criteria, namely the presence of microaneurysms, hemorrhages, hard exudates, cotton wool spots, venous beading/loops, intra retinal microvascular abnormalities, and neovascularization.

The images in the original dataset come from different models and different types of cameras, which affect the visual appearance of left vs. right. Some images are also shown as one would see the retina anatomically while others are shown as one would see through a microscope condensing lens. Like in the real world, there are noise and imaging artifacts present in this dataset, and photos may be out of focus, underexposed, or overexposed.

Due to the presence of these artifacts, the original dataset was not directly used for this project. Courtesy of the Bay Eyecare Institute, the dataset used for training the CNN model

consists of transformed CFP photos from the 2015 Kaggle competition/Bay Eyecare Institute that have undergone extensive image processing. Image processing was done via a program, created by the Bay Eyecare Institute, that uses the OpenCV library in python (cv2). Because the program was created for the institute's own private purposes, I am unable to include the program with the project submission. Some of the main image processing techniques used within the program include Gaussian blur, which uses a low-pass filter to remove high-frequency components, circular cropping, and resizing; application of these techniques produce clearer images from which a model is able to learn features more effectively. See here and here for more details. See next page for an example of a CFP and its processed counterpart, with ROI circled in red.

Due to the binary nature of this classification task, images from class 1/2/3/4 were grouped together as DR-positive images, while images from class 0 were grouped as DR-negative images. The dataset containing the transformed images (slightly imbalanced, with 5% more DR-negative than DR-positive) was split into a training set, a validation set, and a testing set. 852 images were used for training, 213 images were used for validation, and 160 images were used for the testing set. The imbalance in the dataset will be addressed later in this writeup.

The Initial Architecture of the CNN Model:

In order to greatly accelerate training time, code was written and run on Google Colab using GPUs as hardware accelerators. Keras (TensorFlow) was used in order to implement our CNN using a simple sequential model. See next page for a graphic representation of the model. The details of the model's initial architecture are listed below; subsequent modifications based on initial results are described later in this writeup. See code for the newest architecture:

Data augmentation layer that employs random horizontal flipping, random rotation, and random zoom
- Artificially expands the size of the training dataset by creating modified versions of images in the dataset, reducing overfitting.

Rescaling layer that standardizes the RGB channel values from [0, 255] to [0,1]
- Makes input values smaller, improving training performance

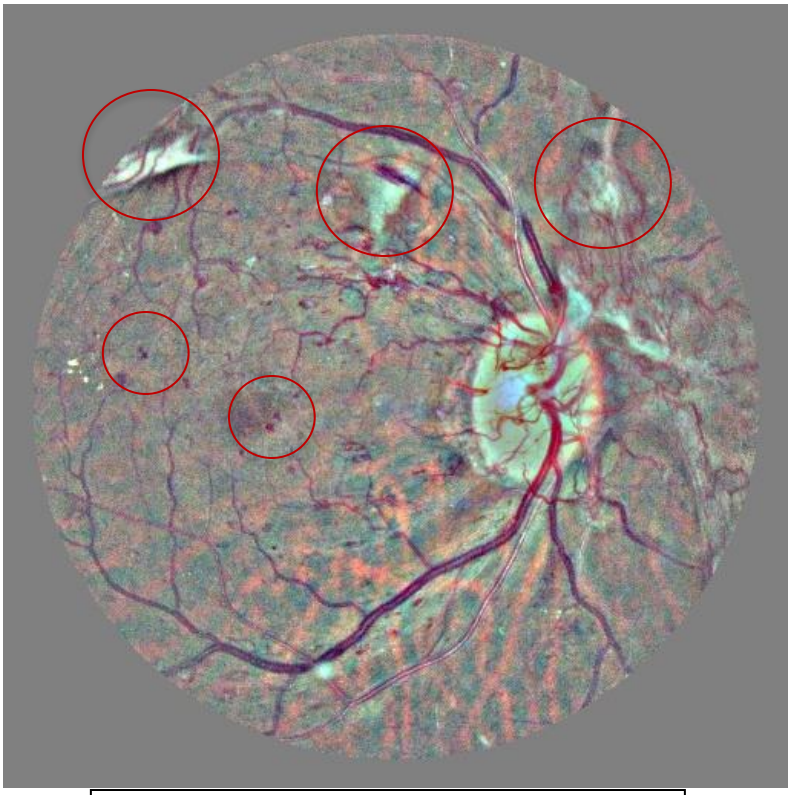
Stack of 3 Conv2D layers with 3x3 kernel size and ReLU activation function
- Each Conv2D layer is followed by a max-pooling 2D layer
- Increasing number of filters in each layer (32->64->128)
- The purpose behind increasing # filters over time is that the first layer captures simple features of the images (edges, color tone, etc). Subsequent layers combine these simple features into bigger, abstract patterns; thus, more filters are needed in order to capture the vast number of abstractions possible.
- Similar to the architectures that Yann LeCun advocated in the 1990s for image classification

Dropout layer, with a dropout rate of 30%
- Regularization technique that reduces overfitting

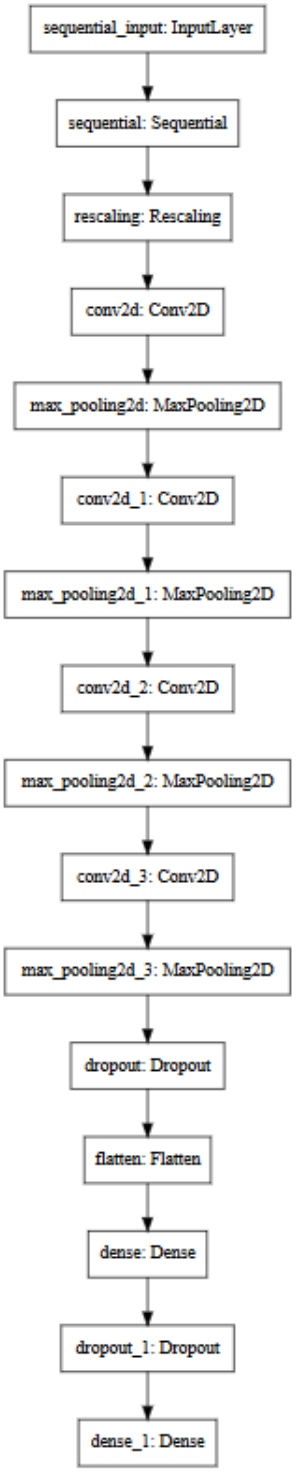
Two fully connected, dense layers with drop out in between
- The first dense layer has 128 units and uses ReLU activation
- Dropout layer with a dropout rate of 30%
- The second (last) dense layer has a single unit and uses sigmoid activation for binary classification.

Scott Song

CFP of Stage 4 Retina, **before** processing



CFP of Stage 4 Retina, **after** processing



Architecture of the Initial CNN (Sequential)

Scott Song

Initial compilation of the CNN model:

Optimizer: RMS prop with learning rate of 0.001

- Adaptive learning rate method designed to optimize gradient descent
- Maintains a moving (discounted) average of the square of gradients
- Divides the gradient by the root of this average
- See here for a thorough explanation of how RMSprop works.

Loss function: Binary Cross Entropy

- Perfect for binary classification tasks
- See here for a thorough explanation of how BCE works.

Metrics – accuracy, sensitivity, and specificity

- Explained in the next section

Results and Significance:

For this project, loss and accuracy were used as the main metrics for analyzing the performance of the model during training and validation. Sensitivity and specificity were also recorded but not used for analysis; this is because the metrics are batch-wise averages, not global metrics of the epoch, which can be potentially misleading. Sensitivity is the ability of the model to correctly identify patients with DR, telling us the probability that the model detects DR, given that the patient really has DR. Specificity is the ability of the model to correctly identify people without the disease, telling us the probability that the model does not detect DR, given that the patient doesn't have DR.

Initial training/validation results (epoch 1):

	Loss	Accuracy	Sensitivity	Specificity
Training	1.8744	53.4%	18.74%	84.6%
Validation	0.6729	60.09%	0%	100%

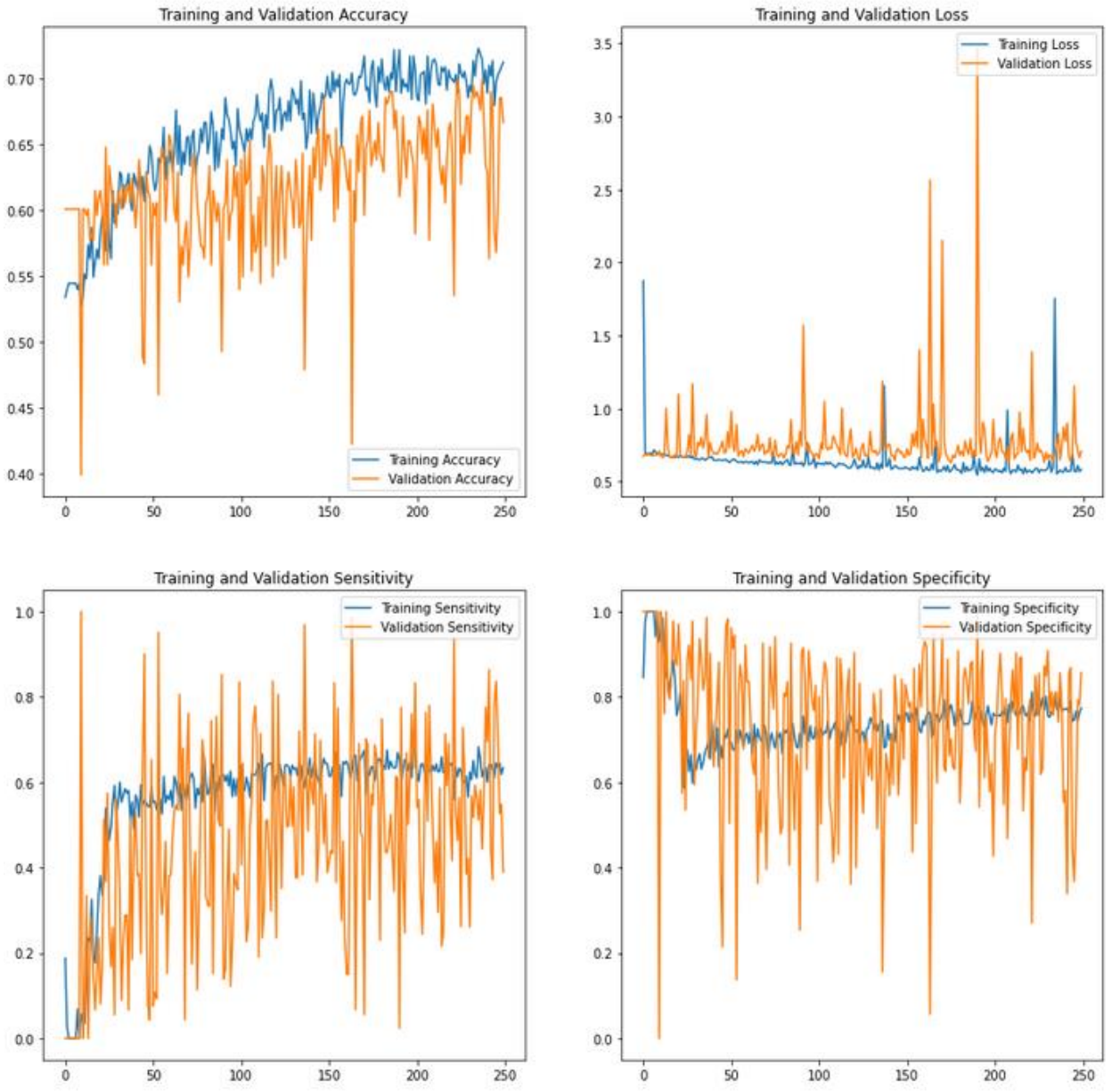
After training for 250 epochs (using the best model within 5 epochs of the end of training):

	Loss	Accuracy	Sensitivity	Specificity
Training	0.5700	70.89%	61.86%	76.30%
Validation	0.6650	68.54%	54.85%	78.14%

Evaluating the trained model on the testing dataset:

	Loss	Accuracy	Sensitivity	Specificity
Testing	1.5038	40.62%	30.61%	62.41%

Scott Song

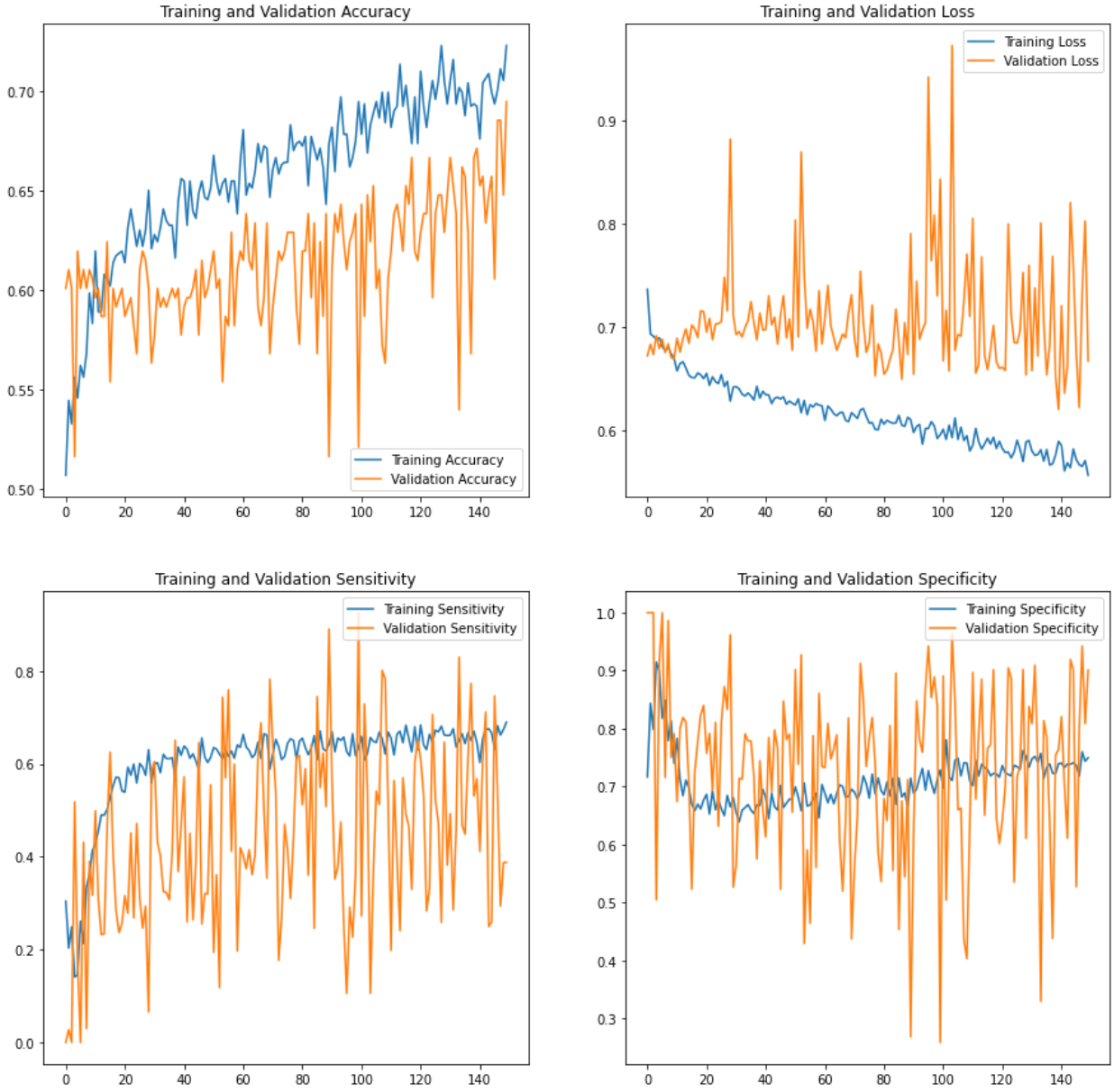


Performance of Model 1

Based on the graphs, model 1 (first model tested) initially appeared to train/learn decently, albeit very slowly, on the training dataset, with loss, sensitivity, and specificity improving with more epochs (see blue curve). However, the trained model exhibited subpar performance on the validation data set, in which the loss shows no significant decrease by the end of training (0.6279->0.6650). Validation loss and validation accuracy fluctuated heavily/randomly during training from epoch to epoch (see orange curve); a case could be made for validation accuracy increasing somewhat consistently, but even this is a stretch. Overfitting was also occurring due to the minor, but consistent increases in validation loss over time. Furthermore, the trained sequential model demonstrated even poorer performance on the testing data set: a loss of 1.5038 and accuracy of 40.62%.

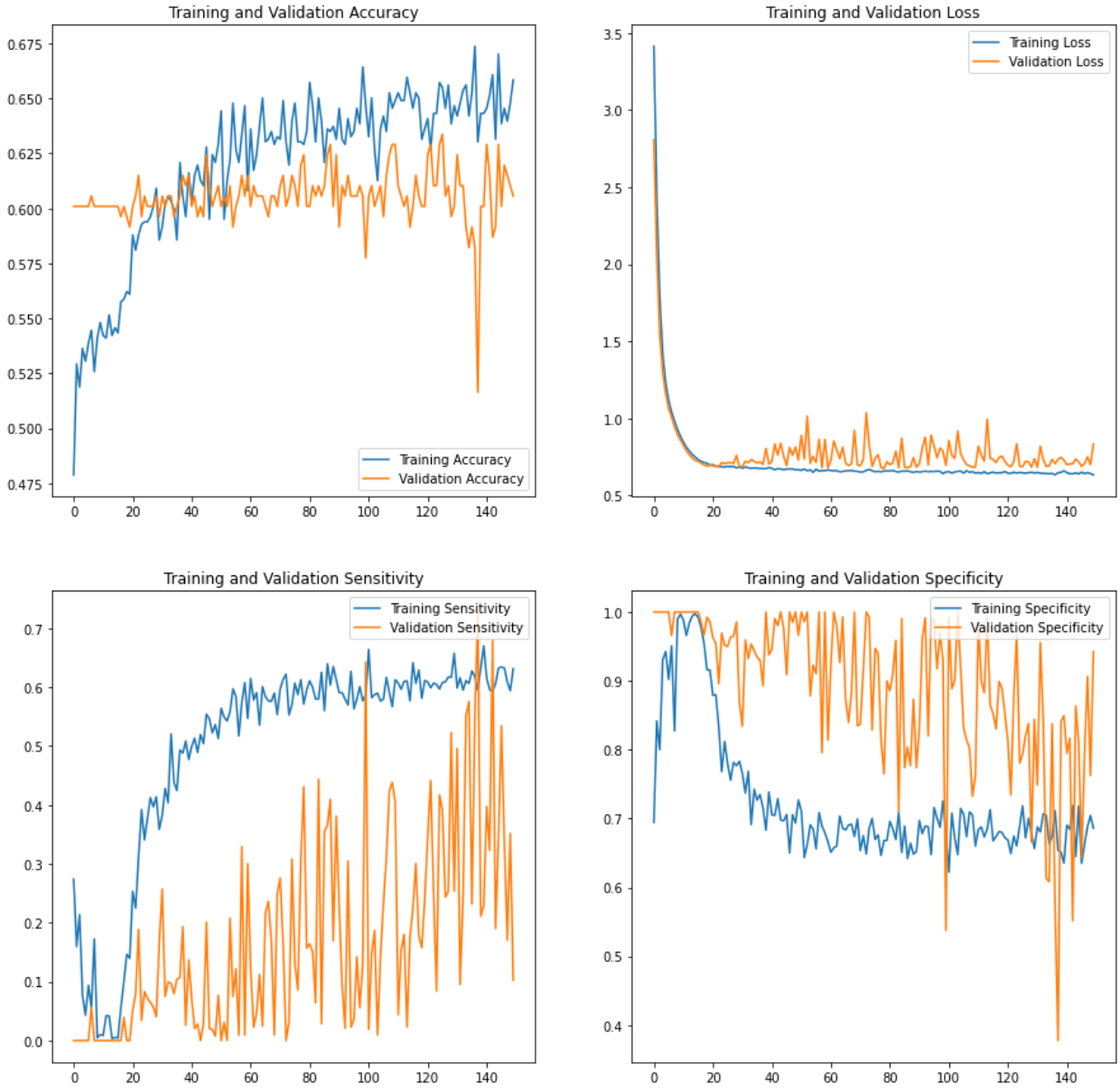
Scott Song

Based on the jaggedness of the graphs, it was possible that the initial default learning rate of 0.001 was too high. Thus, a lower learning rate of 0.0001 with more epochs (500) was tested to see if the jaggedness present in both training and validation performance metrics could be reduced (termed model 2). Unfortunately, adjusting the learning rate did not do much to reduce the jaggedness of the curves. Furthermore, it became clear within 150 epochs that overfitting was still occurring, as validation loss exhibited fluctuations and became consistently larger over time, in contrast to the relatively smooth decrease in training loss:



Performance of Model 2

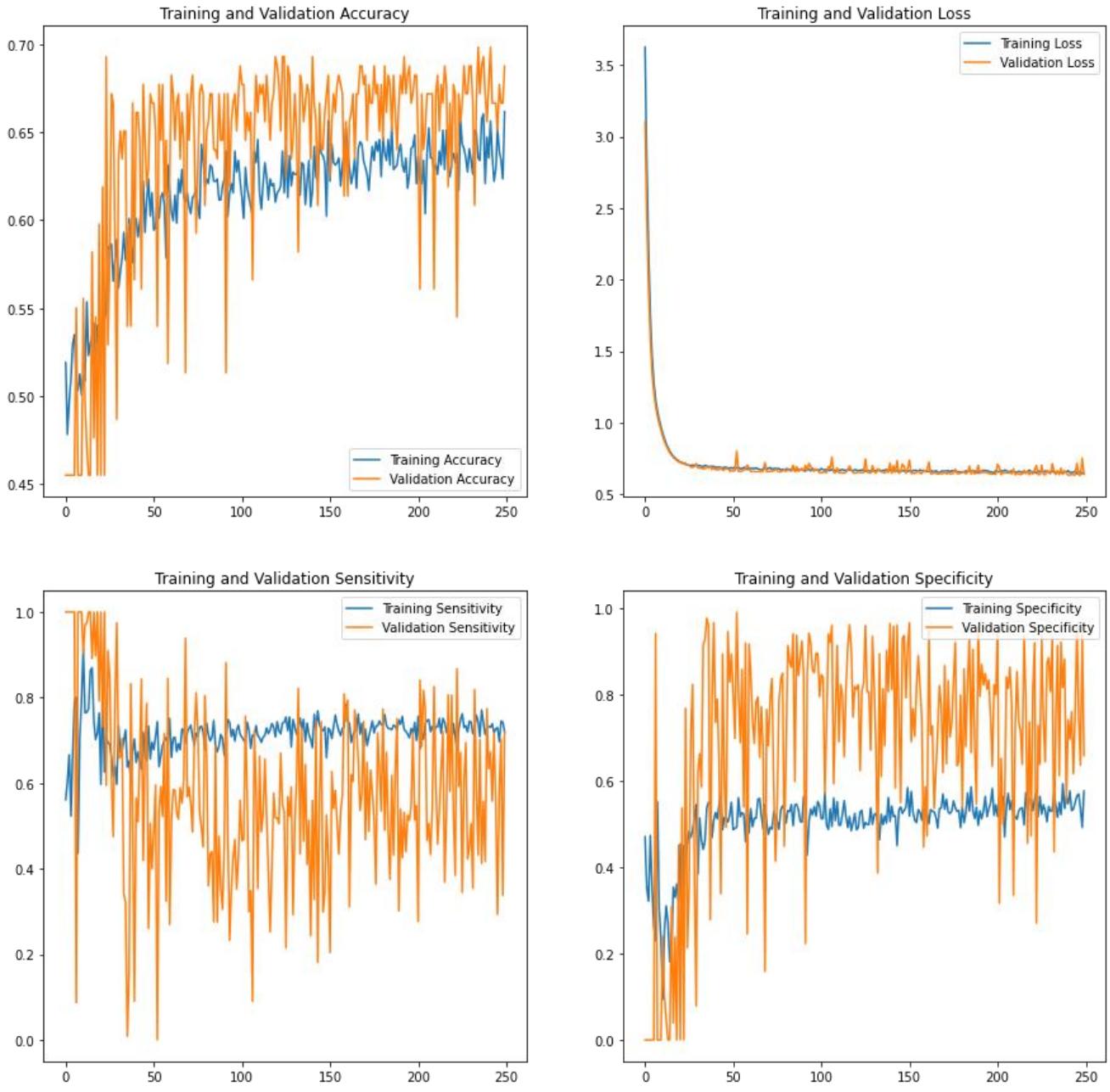
In order to combat the issue of overfitting, L2 regularization (at 128-unit dense layer and 128-filter Conv2D layer) and additional drop out layers (following the 64-filter Conv2D layer) were added to the CNN architecture. Additionally, dropout was increased from 0.3 to 0.4 after the 128-unit dense layer. Using an initial learning rate of 0.0001 and 150 epochs produced the following graphs (termed model 3):



Performance of Model 3

Despite the implementation of data augmentation, dropout, and L2 regularization, training accuracy and training loss demonstrated a gradual increase and decrease over time respectively while both validation accuracy and loss demonstrated random fluctuations after 20 epochs, an indication that over fitting was still occurring. Thus, the final step towards reducing overfitting was to eliminate the imbalance in the dataset (5% more DR-negative than DR-positive) via

random under-sampling. This was done by manually removing a subset (~120 photos) of the DR-negative images from the overall dataset; the subset was determined randomly to introduce as little bias as possible. Using the newly balanced training and validation datasets (473 DR-negative/473 DR-positive images total) with an initial LR of 0.0001 produced the following results (termed model 4):



Performance of Model 4

Scott Song

Initial training/validation results (epoch 1):

	Loss	Accuracy	Sensitivity	Specificity
Training	3.6253	51.92%	56.13%	46.99%
Validation	3.1041	45.50%	100%	0%

After training for 250 epochs:

	Loss	Accuracy	Sensitivity	Specificity
Training	0.6432	66.18%	71.93%	57.59%
Validation	0.6382	68.78%	71.92%	65.82%

Evaluating the trained model on the testing dataset:

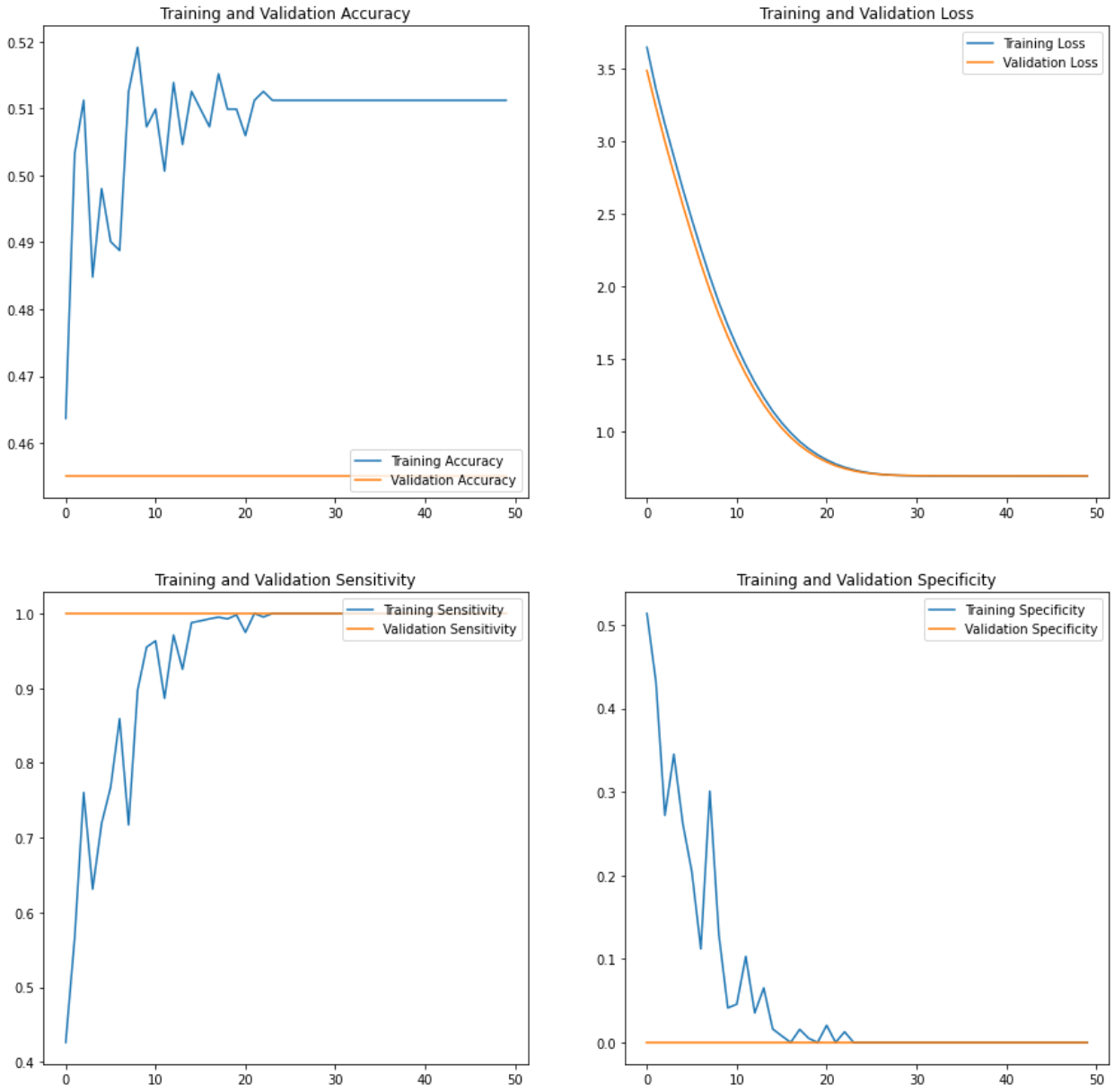
	Loss	Accuracy	Sensitivity	Specificity
Testing	0.7986	50%	50.33%	43.33%

After training for 250 epochs, validation loss (0.6382) and validation accuracy (68.78%) are nearly equal in magnitude to training loss (0.6432) and training accuracy (66.18%) respectively. This is a noticeable improvement when compared to model 1, which had a larger validation loss of 0.6650. Furthermore, the drastic fluctuations in validation loss are no longer present, reflected by the smoothness of the validation loss curve relative to curves produced by earlier models. The closeness of validation loss and training loss throughout the entirety of training is an indicator that overfitting has been significantly reduced. Compared to the model 1, model 4, when evaluated on the testing dataset, yields a significantly lower loss ($0.7968 < 1.5038$) and a greater accuracy ($50\% > 40.62\%$). The difference in accuracy and loss between the validation and testing datasets may be due to the small size of the testing dataset, which only has 40 images per class. Furthermore, it is also highly likely that the test dataset contains more poor-quality images and mislabeled images relative to the validation/training datasets (based on manual inspection).

Interestingly, validation accuracy is consistently greater than training accuracy after ~60 epochs. The same trend can be observed in validation loss, where validation loss is consistently smaller than training loss after ~60 epochs. These observations can be explained by the usage of dropout in several places in the CNN architecture. When training, a percentage of features (30%/40%) are set to zero. When validating, all features are used and scaled appropriately; thus, the model at validation time is more robust, leading to higher validation accuracy and lower validation loss relative to their training counterparts. These observations can also be explained by the usage of L2 regularization during training, which penalizes the CNN for having large values for weights by increasing training loss. Thus, training loss being slightly greater than validation loss is a good sign that dropout and L2 regularization are working properly to reduce overfitting.

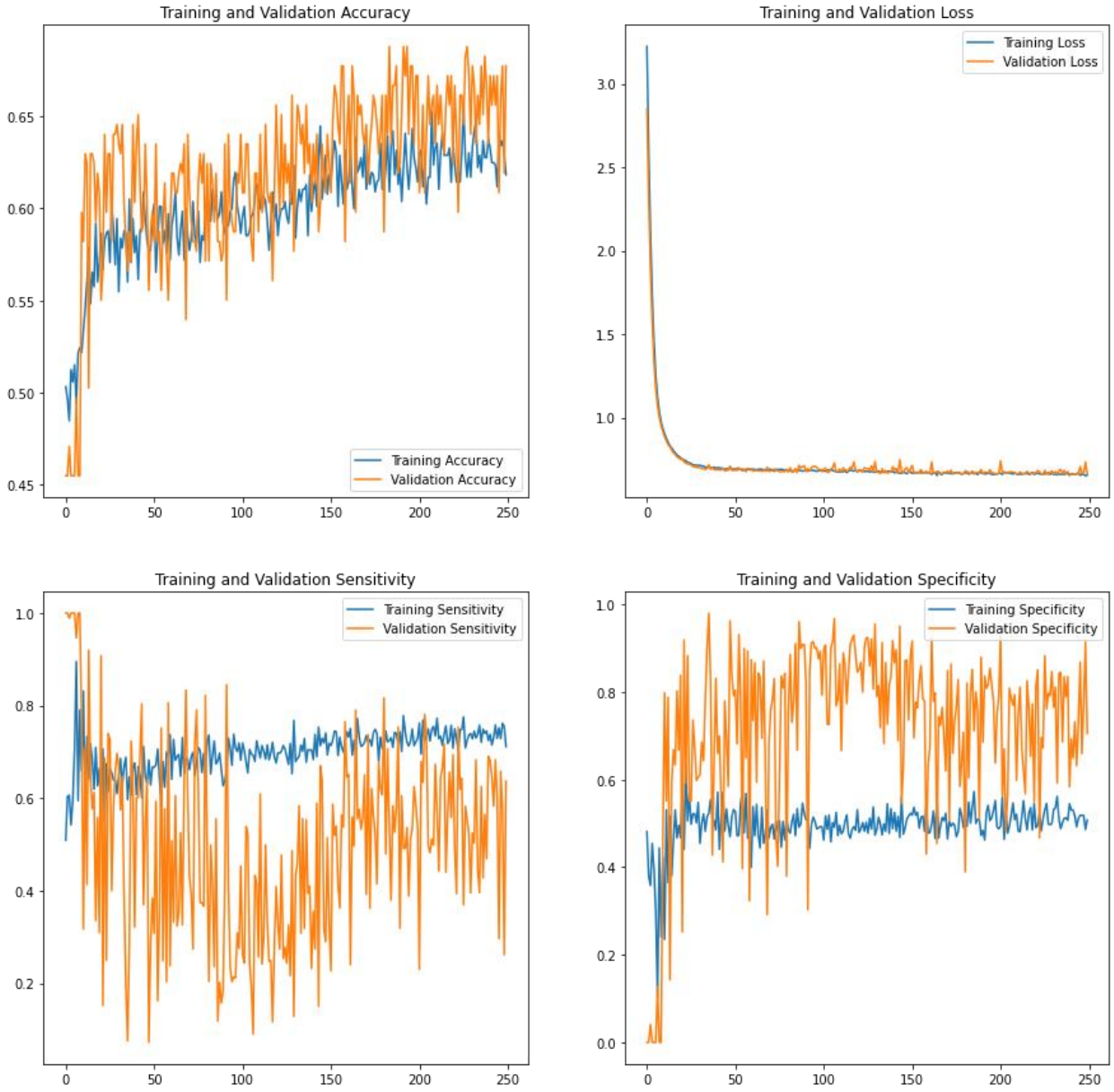
Scott Song

Although adding additional convolutional 2D layers and dense layers increases model complexity, an increasingly large network without a large training set (as in our case with < 500 images per class) is more likely to overfit. Furthermore, adding more layers does not guarantee that a model will necessarily train/learn better; performance can also decrease. This can be seen by adding two more 2D convolutional layers to the architecture of model 4, which results in unchanging training/validation accuracy (51.12%/45.50% respectively) and loss after only ~21 epochs:



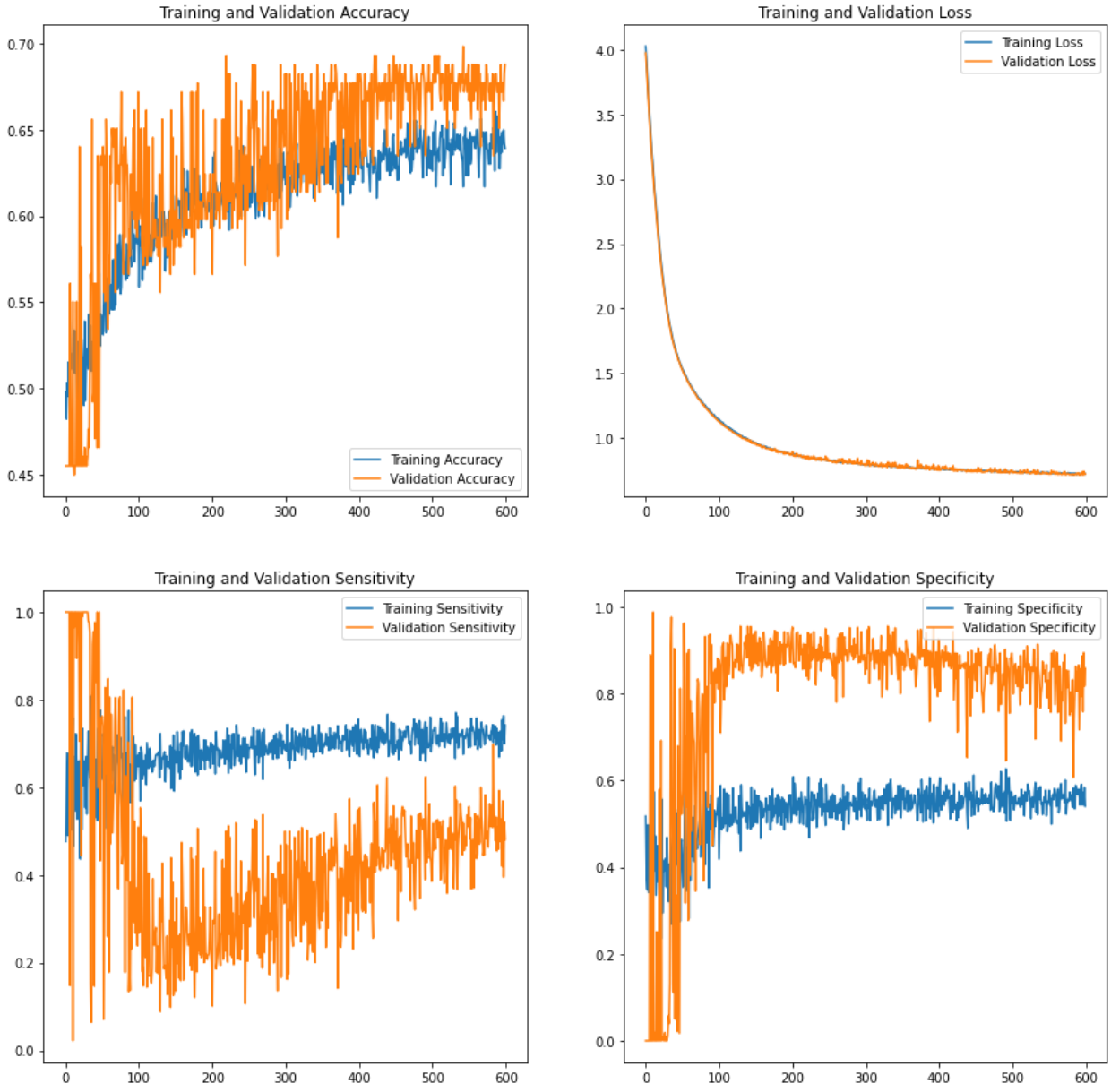
Decrease in performance after adding 2 more 2D Conv2D Layers

Adjusting the filter sizes from 3x3 to 5x5 and decreasing/increasing the number of units from 128 to 64/512 in the first dense layer resulted in poorer model performance for both accuracy and loss (not depicted). Altering the number of filters used in the stack of 3 Conv2D layers from 32->64->128 to 16->32->64 resulted in slightly lower but overall similar performance (depicted below): 0.6635 training loss, ~62% training accuracy, 0.6639 validation loss ~65% validation accuracy.



Slight decrease in performance upon altering the # filters used in the Conv2D layers of Model 4

Having tested multiple alterations in the CNN, the architecture used in model 4, when trained with the balanced data set, is the best performing seq. model. Using this CNN architecture with an even lower initial LR of 0.00001 over 600 epochs (see notebook for code) noticeably reduced the range of the fluctuations in accuracy/loss/sensitivity/specificity and consistently produced models with training/validation accuracies of ~65%/~68% respectively:



Increased performance after decreasing LR to 0.00001 (600 epochs)

Conclusion and Implications:

Despite employing image preprocessing, implementing multiple regularization techniques, and balancing the dataset, the final sequential model is still far from producing useful predictions. However, this result is not unexpected. Given the nature of the classes, with subtle differences separating DR-positive from DR-negative images, this is a hard classification problem. Furthermore, having less than 500 images for training per class further complicates the problem, as a small dataset substantially limits the complexity of models that can be created due to the potential for overfitting with big networks; less complex models, like my sequential model, will

be unable to capture the differences that differentiate DR-negative images from DR-positive images. Although this simple sequential model is not powerful, its ability to do moderately better than random guessing (~68% accuracy on validation set) suggests that the end goal of this field of work, which is to create a model that achieves 90+% accuracy with high sensitivity and specificity, is definitely possible. Such a model would have realistic clinical potential, performing on par with an experienced retina specialist. The process of creating such a model will involve experimenting with many CNN architectures, coupled with extensive trial and error. I will be continuing this project next summer (see future work section).

Future Work:

1. Greatly expand the size of the training dataset, such that each class has at least over 5,000 photos.
    a. This will allow us to train models with increasingly complex architectures (like VGG16/19, which have more than 135 million parameters) without having to worry much about overfitting (provided that mechanisms to prevent overfitting like dropout are also implemented).
2. Cropping out the region of interest (i.e., the bounding box of the eye region) for the training and validation dataset.
    a. The image features distinguishing between 0 and 1 are very subtle; thus, higher resolution is needed.
3. The input image size is only 180x180. Higher resolutions (512x512) may be necessary for 'exposing' subtle image features distinguishing between DR-positive and DR-negative images.
    a. A CNN searches for thousands of patterns. It will find some patterns with some sizes and other patterns with other sizes. Experimenting with different input sizes is necessary for finding the best input size capable of recognizing patterns present in DR-positive images.
        i. See here for an in depth explanation
    b. I tried using 512x512, but it significantly increased the time taken for each epoch. Furthermore, training became significantly slower (tiny changes in accuracy from epoch to epoch). This is likely due to the simplistic sequential model I'm using; thus, using increased input image sizes should be looked into when transitioning into more complex CNNs
4. Transitioning from a sequential network model, as it is much too simple for this task.
    a. Future work should transition into using pretrained networks included with Keras as a baseline/starting point, such as MobileNet and NasNet. (Transfer Learning)
    b. Look into the Keras functional API, which supports arbitrary model architectures that are more flexible than Keras sequential models. The functional API supports models with non-linear topology, shared layers, and multiple inputs or outputs.
5. Moving from doing a binary classification task to doing a complex multi-class classification task, assigning one of the five classes (0/1/2/3/4) to a transformed image of the retina.

Scott Song

Related Work:

Listed below are 2 Kaggle Competitions which involved building a ML model to detect DR:

https://www.kaggle.com/c/diabetic-retinopathy-detection/
https://www.kaggle.com/c/aptos2019-blindness-detection/

Listed below are several papers related to using ML to diagnose DR (not an exhaustive list):

Harry Pratt, Frans Coenen, Deborah M. Broadbent, Simon P. Harding, Yalin Zheng, Convolutional Neural Networks for Diabetic Retinopathy, Procedia Computer Science, Volume 90, 2016, Pages 200-205, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2016.07.014.

Gayathri S., Varun P. Gopi, P. Palanisamy, A lightweight CNN for Diabetic Retinopathy classification from fundus images, Biomedical Signal Processing and Control, Volume 62, 2020, 102115, ISSN 1746-8094, https://doi.org/10.1016/j.bspc.2020.102115.

Shaban M, Ogur Z, Mahmoud A, Switala A, Shalaby A, Abu Khalifeh H, et al. (2020) A convolutional neural network for the screening and staging of diabetic retinopathy. PLoS ONE 15(6): e0233514. https://doi.org/10.1371/journal.pone.0233514

Lam, C., Yi, D., Guo, M., & Lindsey, T. (2018). Automated Detection of Diabetic Retinopathy using Deep Learning. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, *2017*, 147–155.

A. Kwasigroch, B. Jarzembinski and M. Grochowski, "Deep CNN based decision support system for detection and assessing the stage of diabetic retinopathy," *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, Swinoujście, 2018, pp. 111-116, doi: 10.1109/IIPHDW.2018.8388337.

Interestingly, researchers have also begun using ML to predict future DR progression:

Arcadu, F., Benmansour, F., Maunz, A. *et al.* Deep learning algorithm predicts diabetic retinopathy progression in individual patients. *npj Digit. Med.* **2,** 92 (2019). https://doi.org/10.1038/s41746-019-0172-3

Code Instructions:

1. The file must be run on Google Colab.
2. Add the final_photos_all.zip file to your google drive (My Drive). Make sure that it is NOT located in any folders, otherwise the program will not run properly.
3. To make training faster, make sure to enable GPU usage (Edit -> Notebook Settings -> GPU -> save).
4. Upon first run, you will have to authenticate Google Drive in the first cell.
5. If it doesn't run properly (might've missed something to add here), please contact me by slack or by email, ssong21@amherst.edu.