Eye Diseases Detection

CSDS 312
Team 5 Randolph Zhao (yxz1648), Sherry Zhao (txz270), Shiqi Li (sxl1350), Zhaoyu Huai (zxh363), Zora Li (jxl1816)

Abstract

With rising sociodemographic status and life expectancy, many countries around the world are seeing a shift in the disease burden, such as the presence of more cases of vision impairment. Unfortunately, many eye diseases that may lead to blindness often remain undetected until they have developed to a vision-threatening condition. Our group aims to develop a classification model for glaucoma and diabetic retinopathy, further assisting physicians in their field. We carefully preprocessed the images to exclude some useless information. We designed our model based on EfficientNetV2M to adapt two outputs and applied multiple optimization methods, including image data augmentation, multiple GPU scaling, XLA, and mixed-precision training. From our experiments, our model achieved 88% accuracy in diabetic retinopathy and 96% accuracy in glaucoma. Carefully scrutinizing the performance of our model using precision, recall, F1 score, and ROC curse, we believe our model performs well in recognizing diabetic retinopathy and glaucoma and should be able to provide reliable information in the real-world environment.

Introduction

Diabetic retinopathy (DR) is one of the most common and insidious complications of diabetes, which is able to progress with no symptoms until leading to visual impairment or even permanent blindness if not discovered in time [2]. Glaucoma is a group of diseases that may lead to blindness. For all types of glaucoma, the nerve connecting the eye to the brain is damaged, often due to high eye pressure. Unfortunately, these eye diseases often remain undetected until it develops into a vision-threatening stage [7]. Retinal screening is one of the conventional solutions for the early diagnosis, but currently in the real world, the state of screening still leaves a significant proportion of undiagnosed patients, which may be partially due to the low frequency of going to eye screening visits [7]. Early detection and prevention of eye disease progression are essential to mitigate the rising threat. Today, the evaluation of the severity and degree of retinopathy is mostly performed by physicians based on the patients' fundus retinal images. However, not only is this task time-consuming and tedious, but it also requires an intensive effort due to the lesions' small sizes and their lack of contrast [2]. Therefore, an automatic diagnosis system is imperative. Our group aims to develop a classification model for DR and glaucoma, alleviating the rising threat of these eye diseases, and supporting medical experts in their work.

Related Work

Many studies have explored the application of deep learning techniques, particularly convolutional neural networks (CNN), for detecting DR. Le et al. conducted a similar study to investigate the practicability of

detecting DR using deep learning techniques [1]. Instead of using raw retina images taken using fundus photography, they worked on classifying optical coherence tomography angiography (OCTA) images, which is a noninvasive imaging technique that produces depth-resolved images of microvasculature in the retina. While our project employs EfficientNet-V2, another deep-learning CNN architecture, VGG-16, is used in this study by Le et al., which is one of the most popular pre-trained models for image classification. Similar to our project where transfer learning was applied to transfer the learned feature maps, the researchers also implemented the process of transfer learning [1]. They trained their small dataset to achieve robust DR classification with transfer learning, for overfitting is common when training CNNs on small datasets, and transfer learning happens to overcome this problem by leveraging the weights in the pre-trained network [1].

In Chetoui and Akhloufi's research, they proposed the use of a deep learning architecture based on EfficientNet to detect referable DR and vision-threatening DR by conducting tests on two public datasets, EyePACS and APTOS 2019, which are also used as part of the datasets in our project [2]. EfficientNet-B7 was used and fine-tuned to be more efficient for DR classification [2]. Compared to EfficientNet-B7, EfficientNet-V2L, which is employed by our group, achieves 0.6% better accuracy [3]. In general, the family of EfficientNetV2 convolutional networks possesses better parameter efficiency and faster speed of training than previous models [3].

To detect early-to-late stages of DR, Dai et al. developed a deep learning system called DeepDR with a grading of DR as mild, moderate, severe, and proliferative [4]. This system consists of three subnetworks, including an image quality assessment subnetwork, a lesion-aware subnetwork, and a DR grading subnetwork [4]. The DR base network was formed by pretraining the ResNet to improve the performance of DR grading and specific retinal lesions detection [4]. Thus, the problem of vanishing gradients can be eliminated, making it more sensitive in extracting features for small lesions compared to other network architectures such as Inception and VGG [4]. In this study, the researchers also employed transfer learning to enhance the performance of the lesion-aware DR grading task where the DR base network was transferred to the three subnetworks of DeepDR instead of training randomly initialized subnetworks directly [4].

Similarly, Li et al. investigated automatic DR detection in retinal fundus images as well. However, they used a deep transfer learning approach with a different network, Inception-v3, as the base CNN [5]. To efficiently reduce the computation process and increase both the network's width and depth, this network clustered similar sparse nodes into a dense structure [5]. Due to the relatively small dataset of retinal images, the Inception-v3 network, pre-trained with the ImageNet dataset, was transferred and further fine-tuned for categorizing fundus images by the researchers to decrease the training time and achieve higher accuracy [5].

Methods

2.1 Preprocess

Image preprocessing is the first step we took to format the images before they are used to train the model, cleaning image data for the model input and decreasing the model training time. The black borders are removed with only the circle retained. The images are also resized to desired dimensions. Gaussian noise was added by performing a Gaussian blur operation on the images, thus softening the sphere edges and increasing robustness [17]. Lastly, a circular crop was created around the image center. Although some information may get lost due to this circular crop, the amount lost should be limited. Figure 1 below shows a diabetic retinopathy class 2 (mild) image before and after preprocessing, and Figure 2 shows the fundus image of a patient with glaucoma before and after preprocessing.

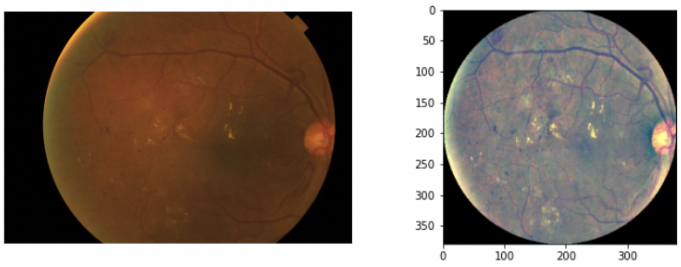


Figure 1. Diabetic Retinopathy Class 2 (mild) before and after preprocessing.

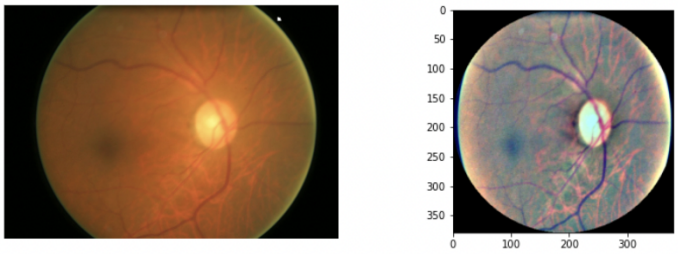


Figure 2. A fundus image of a patient with glaucoma before and after preprocessing.

2.2 Model

In this project, we implemented the EfficientNet model to analyze the medical images in our datasets. Convolutional Neural Networks (CNNs) often start with a fixed number of resources and then be scaled up if more resources are available to increase the accuracy. For example, the ResNet model adds more layers from ResNet-18 to ResNet-200 [8]. However, the scaling up technique of CNNs is not well understood. And the proposal of the EfficientNet model rethinks the way how to scale up CNNs and aims to find a principal approach of scaling up to increase accuracy and efficiency. The common ways to scale up CNNs include three methods: width scaling, depth scaling, and resolution scaling, but these three methods would generate some problems during training. The width scaling can catch more fine-grained features and are easier to train with the wider network. But the problem with width scaling is that wide but shallow networks would be difficult to capture high-level features. Depth scaling is another commonly used scaling-up technique to increase the depth of the network by adding more layers. The intuition of this method is that a deeper network could capture rich and complex features. However, the deep network is difficult to train. Even if we solve the training problem by some techniques such as

adding a batch normalization layer the accuracy would drop. The resolution scaling is to increase the resolution of the input images. With higher resolution, the network could capture more fine-grained patterns. Although higher resolution would increase the accuracy, this increase gradually converges to zero with the increase of resolution [8]. These three scaling-up dimensions are shown in Figure 3. It is common to choose one of the three scaling up dimensions width, depth, and resolution. Some previous work also scales up to two of the three dimensions arbitrarily but often leads to worse accuracy and efficiency due to the randomness [8].

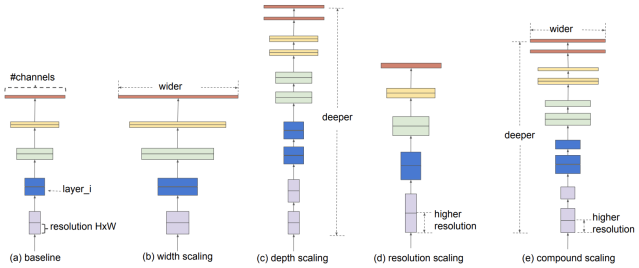


Figure 3. CNNs scaling up dimensions [8]

The key idea of EfficientNet is to balance all dimensions of the network, which is called the compound scaling method. It is found that the three dimensions of scaling up are not independent of each [8]. For example, with a deeper network, we will get better results by increasing the input image resolution because of the larger receptive field. To improve accuracy and efficiency, we have to utilize the association of dimensions and achieve a balance. The compound scaling method uses a compound coefficient φ to scale up each dimension of the network with a constant ratio.

$$
\begin{aligned}
depth\colon d &= \alpha^{\varphi} \\
width\colon w &= \beta^{\varphi} \\
resolution\colon r &= \gamma^{\varphi} \\
s.t.\ \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\
\alpha \geq 1,\ \beta &\geq 1,\ \gamma \geq 1 \text{ [8]}
\end{aligned}
$$

The α, β, and γ here are constants that will be calculated before the complete training with a very small grain. The compound coefficient controls the number of resources available to the network, and α, β, γ connect these resources to the corresponding dimension. According to the research result, EfficientNet achieves state-of-the-art accuracy on ImageNet. In addition, it is more computational efficient with much fewer parameters to analyze compared to other models (Figure 4).

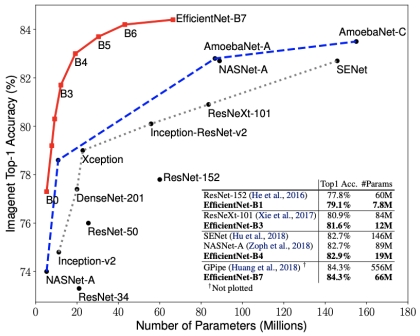
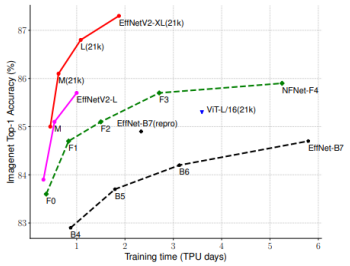


Figure 4. Comparison of different ImageNet models [8]

Although the base EfficientNet model could achieve high parameter efficiency, the training speed is slow due to large image sizes. The large image size leads to serious use of the hardware memory which is fixed. So we have to train with EfficientNet with a small batch size which slows the training process [9]. Another problem is that the depthwise cannot fully utilize the accelerators. Depthwise convolutions are slow in early layers but fast in later layers [9]. The base EfficientNet model has these problems because it equally scales up each stage with a constant ratio. The EfficientNetV2 model modifies the original version and applies a non-uniform scaling-up strategy. The common way to solve this problem is to progressively increase the image size during training, but it often leads to an accuracy drop. This accuracy drop was caused by the unbalanced regularization. As a result, the EfficientNetV2 applied a new progressive learning training method, which jointly increases the image size and regularization during the training [9]. The performance of EfficientNetV2 compared to other models is shown in Figure 5.



(a) Training efficiency.

	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

(b) Parameter efficiency.

Figure 5. Training time and parameter efficiency of different models [9]

In the project, we instantiated an efficientNetV2 model without top layers and loaded pre-trained weights into it. Then we copied the output layers for each output and then added the same top layers for each output. The top layers include Global Average pooling, batch normalization, dropout, and dense layers. We add a batch normalization layer to speed up training. A dropout layer is added to control the overfitting problem. During the design phase, we expected that our model would have an overfitting problem due to the small dataset used. The overfitting, if controlled properly, could allow the model to capture as much information as possible. Thus, we only added one dropout layer in our model top layers. Based on the comparison of three models EfficientNetV2S, EfficientNetV2M, and EfficientNetV2L, we

found that the V2M model takes much less time than the V2L model, and could handle larger input sizes than V2S. As a result, we chose EfficientNetV2M for the final result analysis and comparison.

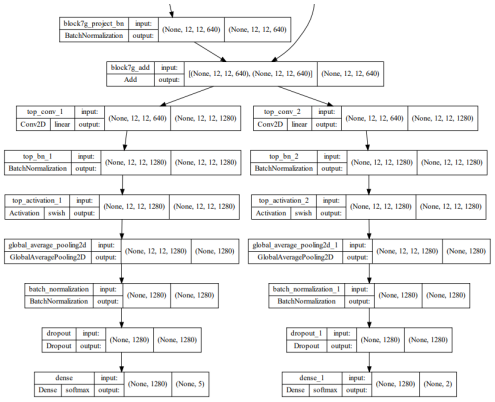


Figure 6. Output layers and top layers

2.3 Optimization

2.3.1 Image Data Augmentation

Overfitting is a problem occurring when the neural network learns the model which only applies to the data in the training set but cannot be used to generalize to solve similar problems [10]. The small dataset is likely to encounter the overfitting problem because the dataset does not have enough variety for learning. The dataset used for eye disease detection is not big enough, and detecting two diseases from various eye colors and disease symptoms is a complex task. To solve the lack of variety problem, image data augmentation is used. The TensorFlow layers are implemented in the neural network to randomly flip the images horizontally or vertically, modify the contrast of the images by a random factor, rotate the images by a random angle, and randomly rescale the image into a range defined by an offset, creating more variants of images from the original dataset [10]. The increase in the variety of datasets by image data augmentation reduces the overfitting problem.

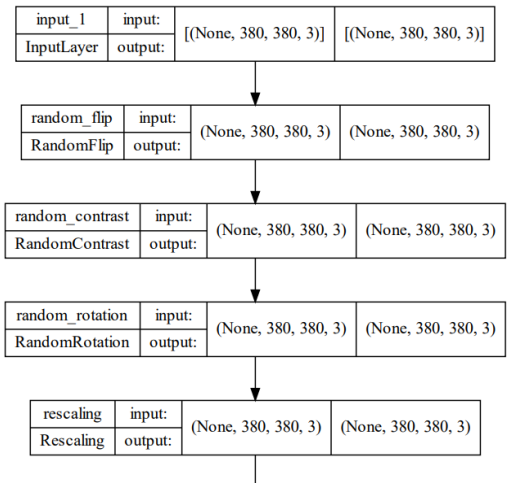
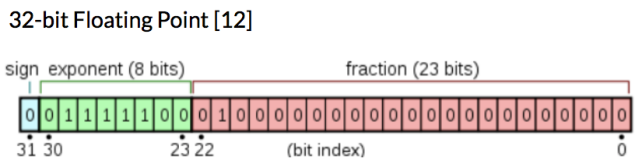


Figure 7. Image Data Augmentation Layers

2.3.2 Mixed precision

By default, the global policy in TensorFlow is set to 32-bit float variable dtype, which means each number is represented by 32 bits [13]. A mixed precision with 16-bit float compute dtype is implemented in this project to increase the computation speed, which means each number is represented by 16 bits during computation.



Figure(). 32-bit floating point

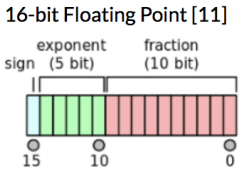


Figure (). 16-bit floating point

	GeForce RTX 2080 Ti
16-bit	56.9 TFLOPS
32-bit	28.5 TFLOPS

Figure (). GPU processor calculating speed comparison

Because each number is shorter with 16-bit floating point, the GPU processor can handle more calculations every second. A mixed precision with 16-bit floating point can perform faster calculations and needs smaller memory for images and models [13]. However, because 16-bit floating point can present less information and cause data loss, it can lead to underfitting or overfitting problems. A really small number near 0 is rounded to 0, and we will have an underflow problem when gradients are equal to 0 due to precision limits. We use loss scaling to solve the underfitting problem during the gradient computation due to the small representation range of 16-bit floats. The loss scaling can multiply the loss value by a scale, and because of the connections in the neural network, all the gradients can be scaled by the same factor which moves them within the range of 16-bit floating point so that underflow problem does not occur [16].

2.3.3 XLA (Accelerated linear algebra)

XLA in TensorFlow is implemented to increase the speed of training. XLA is a domain-specific compiler for linear algebra [14]. It compiles the graphs into optimized vector operations to increase computation speed. If XLA is not used, the graph launches three kernels for computation: one for computing multiplication, one for computing addition, and one for computing reduction. XLA can optimize the graph so that all the computations are done in a single kernel [14]. Also, XLA does not put the intermediate computation result in memory, instead, XLA keeps the results in GPU so that the program does not need to search the memory to compute the next step, thus saving computation time. Because fewer kernels are used and the search time is reduced, the performance of the program with XLA is about 7 times higher than the program run without XLA[14].

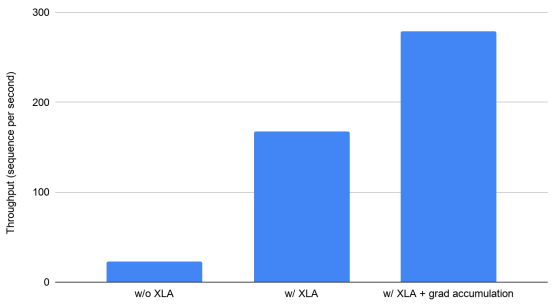


Figure (). Throughout v.s. XLA

2.3.4 Multiple GPUs

Multiple GPUs are used for computation in the project. The program can distribute the calculations to each GPU and run the project in parallel to decrease the time needed. We use a mirrored strategy, which creates a mirrored variable containing replicates of data for each GPU and all-reduce algorithms let the variables communicate their updates so that the data are synchronously trained in multiple GPUs on one device [15]. For this project, 2 GPUs are used and the project is run on HPC.

2.4 HPC techniques

To access the host's file system and devices, we created a Singularity container for our TensorFlow environment and other necessary Python libraries as well in order to provide the same environment for each teammate and for convenience purposes. We also used the bind flag to mirror a certain directory into the environment to access the same dataset with local Python files. Considering that it works well with a scheduler, we used srun to request GPU resources and submit slurm jobs for long-time jobs. Another tool we used is the Globus which moves and shares big data. We set up our own endpoints to transfer files more easily and efficiently.

Results

1. Model Comparison

We compare the performance of three models, V2L, V2M, and V2S as shown in Figure (). There isn't much difference between the three models for both outputs besides a slightly higher accuracy and less loss of the V2S model but relies on the range of uncertainty. Taking the running time into consideration, V2S and V2M take much less time than V2L. From Fig. (), we believe that V2M could have better robustness in complex tasks. Thus, we will develop our own recognition model based on the V2M.

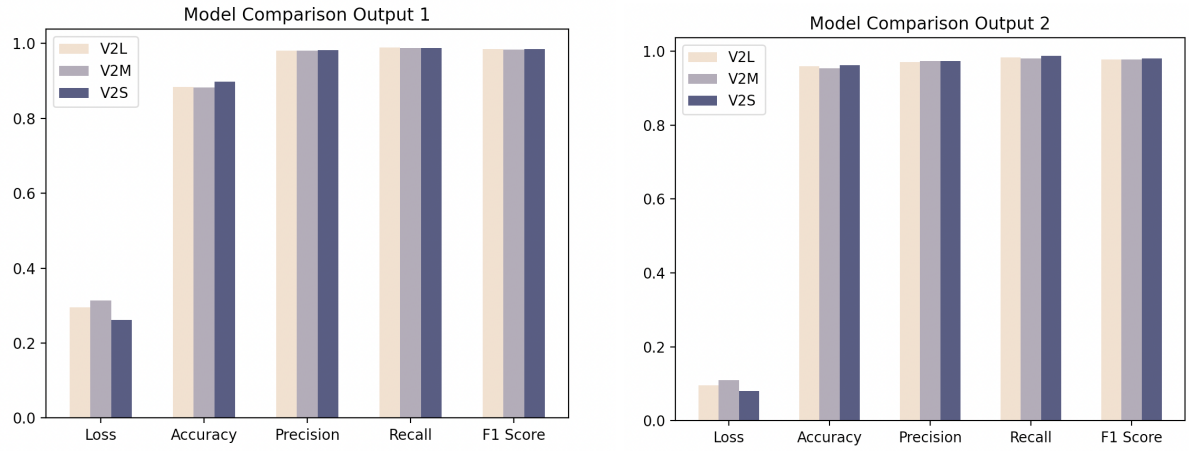


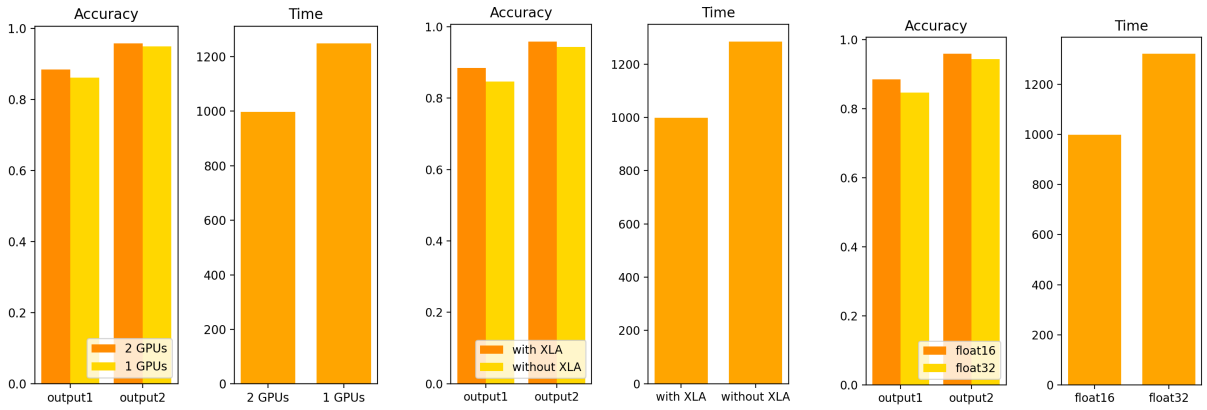
Figure (). Comparison of the performance of three models, V2L, V2M, and V2S for the two outputs.

Total Running Time	
V2L	1026.34
V2M	689.27
V2S	668.42

Table (). A summary of the total run time of the three models.

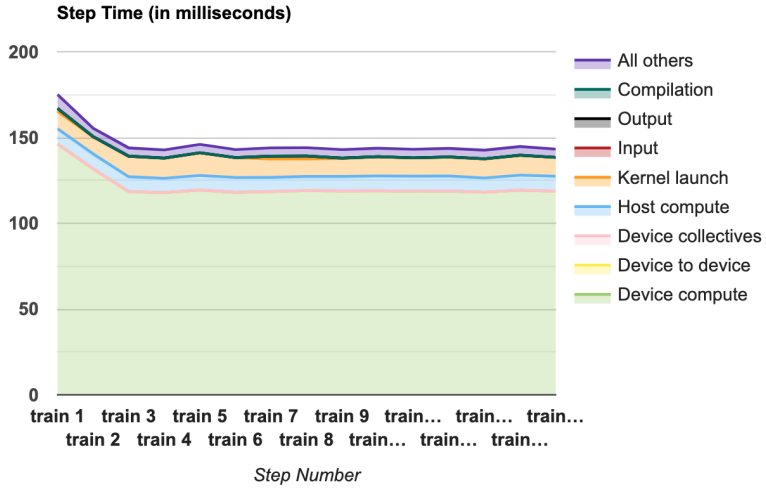
2. Optimization

To maximize the training speed, we further applied multiple GPU scaling, XLA optimization, and mixed-precision training. Fig. () shows the comparison of using 1 or 2 GPUs; Fig. () shows the comparison of using or not using XLAs; Fig. () shows the comparison of using float 16 or float 32. From all three plots, using 2 GPUs with XLA and mix-precision (float 16) enabled results in better accuracy and shorter training time. Thus, we will continue to use this setting in the following experiments.



Figrue (). Accuracy and time comparison of different GPU, XLA, and floating point usages.

At the first experiment phase, we found our model trained very slowly, which usually took half an hour to train one epoch. After checking the details with the Tensorboard, we found the preprocessing part takes 90+% of each step, which is "input bottlenecked". To address this problem, we preprocessed all images first and properly stored the preprocessed data. After applying the above optimization methods, we got a new Tensorboard report as Fig. (). We can see the input takes less than 5% of each step and 90+% time is consumed by the device computation, which means our model is properly optimized that most time of each step is calculating variables to update the mode



Figure(). Summary of the step time performance

3. Performance Analysis

Our research results are summarized in Table () as we measured the performance using precision, recall, and F1 score, which are defined in Eq. () () (). TP refers to the true positive and FP refers to the false positive. They provide different views about how accurate the model does in each class.

The overall accuracy of output 1 (the DR dataset) is 0.88, which is much higher than randomly choosing or always picking the majority class. Note that the precision, recall, and F1 score of class 0 are much higher than those of other classes. This is reasonable because most of the DR dataset is labeled as class 0, non-DR eyes, as indicated by its support value, providing more information about non-DR eyes when training the model. Thus, we believe that our model is limited due to the number of training samples.

For the second output (the glaucoma dataset), the total accuracy is 0.96 which seems to be amazingly good. Again, the precision, recall, and F1 score of class 0 are much higher than those of class 1. But it's quite suspicious this time given that the glaucoma dataset only contains 1300 images. This is because we labeled the whole DR dataset as class 0 (negative) of the task of recognizing glaucoma. But it's still useful in the sense that the model is extremely sensitive to and efficient at identifying non-glaucoma eyes, which could be helpful for some practical purposes.

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

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

$$F1score = \frac{2(precision \cdot recall)}{precision+recall} = \frac{2TP}{2TP+FP+FN}$$

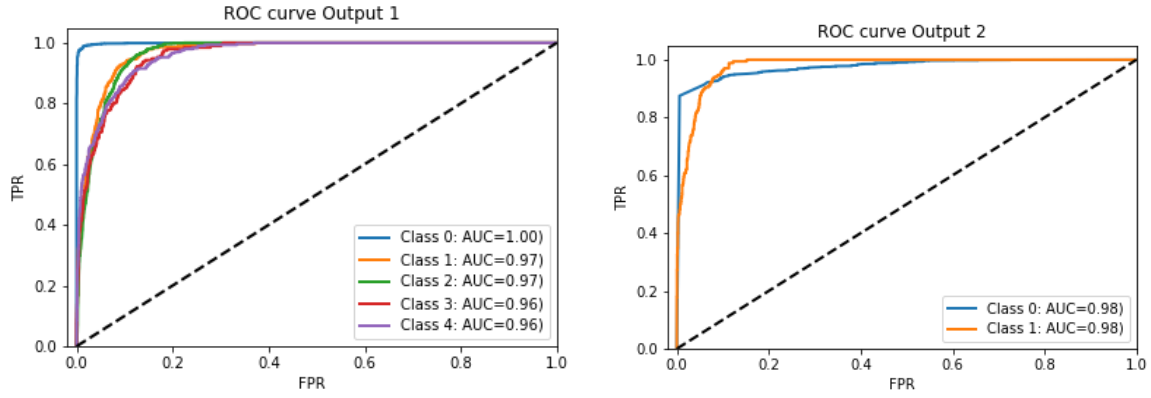
Output 1				
Class	Precision	Recall	F1-score	Support
0	0.98	0.99	0.99	3105
1	0.67	0.62	0.64	370
2	0.75	0.83	0.79	999
3	0.64	0.39	0.48	193
4	0.70	0.55	0.62	295
Accuracy:			0.88	4962

Table (). The overall performance of output 1 from the DR dataset.

Output 2				
Class	Precision	Recall	F1-score	Support
0	0.96	0.99	0.98	4626
1	0.84	0.48	0.61	336
Accuracy:			0.96	4962

Table(). The overall performance of output 2 from the glaucoma dataset.

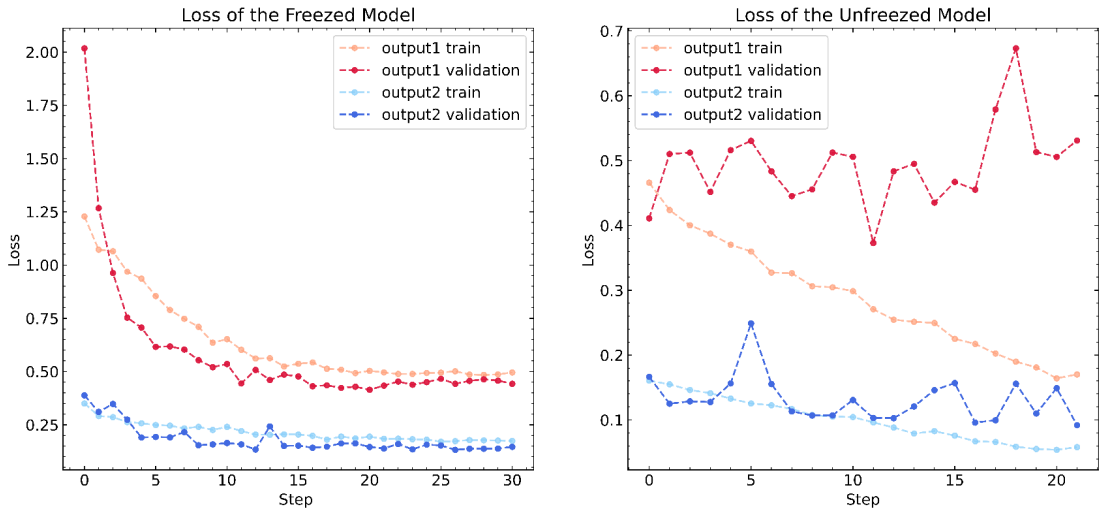
Fig. () and () show the receiver operating characteristic curve (ROC) of the two datasets, revealing the performance of each classification model at all classification thresholds. ROC demonstrates visually the distinguishing capability of a binary system when the threshold is changed over a range. We plotted the TP rate versus the FP rate. For output 1, the area under the class 0 curve is almost perfectly 1, indicating that the model predicts class 0 nearly 100% correctly. The area under the curves of other classes is greater than or equal to 0.96, which are remarkable results. For the second output, the model results in an outstanding AUC of 0.98. Thus, we believe our model has good robustness and high accuracy.



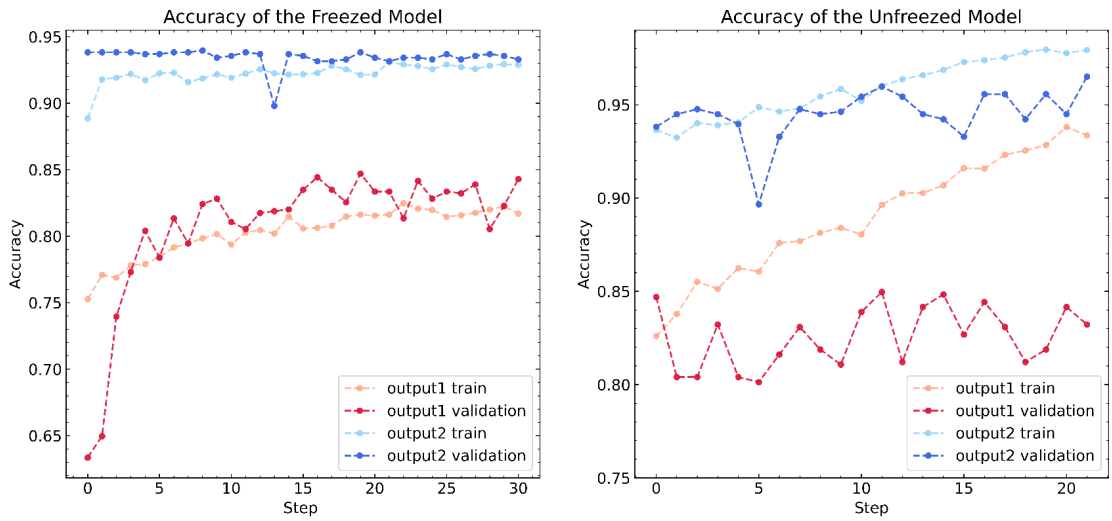
Figure(). The receiver operating characteristic curve (ROC) of output 1.

Figure(). The receiver operating characteristic curve (ROC) of output 2.

Fig. () and () plot the loss as a function of step numbers for the two models. For the frozen model, both train and validation loss curves of the two outputs decrease gradually and seem to reach a plateau, as we would expect. For the unfrozen model, the loss curves of the train model decrease, but the validation curves fluctuate wildly around the plateau values of the frozen model. This issue can be explained by the overfitting that we focused too much on details when training the model but didn't capture the most important characteristics in general. Then the performance of validation of the unfrozen model doesn't improve anymore. Similarly, we plotted the accuracy curves of the two models as shown in Fig. () and (). The train and validation curves of the frozen model both increase and flatten out, while the validation curve of the unfrozen model oscillates about the plateau values. This is the same behavior as the loss curve and is again, caused by the overfitting problem. However, as discussed in the model design section, this overfitting behavior is exactly what we expect: to capture as much information as possible. We also control the overfitting during the training phase of the frozen model, preventing the model overfit too much.



Figure(). The loss curve of the freezed and unfreezed models for the two outputs.



Figure(). The accuracy curve of the freezed and unfreezed models for the two outputs.

Conclusion

In this project, we developed a neural network model based on EfficientNet V2M to address the eye disease recognition problems. We put our emphasis on the DR and glaucoma problems and designed a multi-output model. To maximize the training speed, we implemented multi GPU scaling, XLA optimization, and mixed-precision training. We measured the performance of our model with accuracy, precision, recall, F1 score, and ROC curve. The results showed that our model provides reliable prediction and outstanding robustness for these two eye disease images. Although there is an overfitting problem, we controlled it by using a dropout layer and an early stopping strategy so that it will not overfit too much. The small overfitting issue is desired as our expectation is to allow the model to capture as much information as possible. Thus, we believe that our model performs well for the eye disease recognition task and still has the potential to be further optimized.

Future Work

For future work, our group would like to build a deep learning model that is able to detect more eye diseases besides diabetic retinopathy and glaucoma. Some possible choices of eye diseases include cataracts, clouding of the lens in the eye, and macular degeneration, an eye disease that causes loss in the center of the field of vision. Thus, more patients can be provided with the most-needed and timely treatment corresponding to their specific eye conditions, enhancing the control and prevention of different eye diseases. We would also like to investigate whether there is a model that could have a better performance compared with our current model. Testing with the models employed by other studies, such as those mentioned in related works, would be a good idea to start with. Training the deep learning neural network with the stochastic gradient descent (SGD) algorithm may be another way of leveraging the current model, which could provide high precision and make it easier to fit in the memory, leading to faster computation since only one sample is processed at a time.

Reference

1. Le, D., Alam, M., Yao, C. K., Lim, J. I., Hsieh, Y.-T., Chan, R. V., Toslak, D., & Yao, X. (2020). Transfer learning for automated OCTA detection of diabetic retinopathy. *Translational Vision Science & Technology*, *9*(2), 35. https://doi.org/10.1167/tvst.9.2.35
2. Chetoui, M., & Akhloufi, M. A. (2020). Explainable diabetic retinopathy using efficientnet*. *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. https://doi.org/10.1109/embc44109.2020.9175664
3. Tan, M., & Le, Q. V. (2021, June 23). *EFFICIENTNETV2: Smaller models and faster training*. arXiv.org. Retrieved April 10, 2022, from https://arxiv.org/abs/2104.00298
4. Dai, L., Wu, L., Li, H., Cai, C., Wu, Q., Kong, H., Liu, R., Wang, X., Hou, X., Liu, Y., Long, X., Wen, Y., Lu, L., Shen, Y., Chen, Y., Shen, D., Yang, X., Zou, H., Sheng, B., & Jia, W. (2021). A deep learning system for detecting diabetic retinopathy across the disease spectrum. *Nature Communications*, *12*(1). https://doi.org/10.1038/s41467-021-23458-5
5. Li, F., Liu, Z., Chen, H., Jiang, M., Zhang, X., & Wu, Z. (2019). Automatic detection of diabetic retinopathy in retinal fundus photographs based on Deep Learning algorithm. *Translational Vision Science & Technology*, *8*(6), 4. https://doi.org/10.1167/tvst.8.6.4
6. Weinreb, R. N., Aung, T., & Medeiros, F. A. (2014). The pathophysiology and treatment of glaucoma. *JAMA*, *311*(18), 1901. https://doi.org/10.1001/jama.2014.3192
7. Arcadu, F., Benmansour, F., Maunz, A., Willis, J., Haskova, Z., & Prunotto, M. (2019). Deep learning algorithm predicts diabetic retinopathy progression in individual patients. *Npj Digital Medicine*, *2*(1). https://doi.org/10.1038/s41746-019-0172-3
8. Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. ICML, 2019a.
9. Tan, M., & Le, Q. V. (2021, June 23). *EFFICIENTNETV2: Smaller models and faster training*. arXiv.org. Retrieved April 10, 2022, from https://arxiv.org/abs/2104.00298
10. Mutasa, S., Sun, S., & Ha, R. (2020). Understanding artificial intelligence based radiology studies: What is overfitting?. Clinical imaging, 65, 96-99.
11. Wikimedia Foundation. (2022, March 9). Half-precision floating-point format. Wikipedia. Retrieved April 10, 2022, from https://en.wikipedia.org/wiki/Half-precision_floating-point_format
12. Wikimedia Foundation. (2022, April 6). Single-Precision Floating-point format. Wikipedia. Retrieved April 10, 2022, from https://en.wikipedia.org/wiki/Single-precision_floating-point_format
13. Mixed precision: Tensorflow Core. TensorFlow. (n.d.). Retrieved April 10, 2022, from https://www.tensorflow.org/guide/mixed_precision#training_the_model_with_modelfit
14. XLA: Optimizing Compiler for machine learning: tensorflow. TensorFlow. (n.d.). Retrieved April 10, 2022, from https://www.tensorflow.org/xla
15. Distributed training with tensorflow: Tensorflow Core. TensorFlow. (n.d.). Retrieved April 10, 2022, from https://www.tensorflow.org/guide/distributed_training
16. Mellempudi, N., Srinivasan, S., Das, D., & Kaul, B. (2019). Mixed precision training with 8-bit floating point. arXiv preprint arXiv:1905.12334.
17. Jerbić, B., Švaco, M., Chudy, D., Šekoranja, B., Šuligoj, F., Vidaković, J., Dlaka, D., Vitez, N.,

Župančić, I., Drobilo, L., Turković, M., Žgaljić, A., Kajtazi, M., & Stiperski, I. (2020). Ronna G4—robotic neuronavigation: A novel robotic navigation device for stereotactic neurosurgery. *Handbook of Robotic and Image-Guided Surgery*, 599–625. https://doi.org/10.1016/b978-0-12-814245-5.00035-9