Grading of Diabetic Retinopathy Severity Level

Group Members: Yunchun Pan, Hongshuo Wang, Zhengxi Li, Peng Zhang

Introduction

Diabetic retinopathy (DR) has caused vision loss of approximately 425 million people in 2017, and the number of people suffering from it is still rising. After seeing the importance of grading DR severity level in helping doctors take effective treatments for patients, we want to develop a deep learning model to determine DR severity level given a diabetic retina image. According to the International Clinical Diabetic Retinopathy Scale, the severity level of DR can be graded into five levels, which are 0 (no apparent DR), 1 (mild DR), 2 (moderate DR), 3 (severe DR), 4 (proliferative DR). To label DR severity levels of retina images, we first follow approaches proposed in link to process retina images, including creating circular crop around the image center to remove uninformative dark areas, improving image illuminations, and adding color scale to images. Then, each team member develops at least one deep learning model themselves, such as EfficientNet, ConvNet, InceptionV3, VGG, and DenseNet. Moreover, we evaluated the model performance on the test set by measuring and visualizing accuracy, precision, F1 score, recalls, and ROC curves for different DR levels. In the end, we make a first attempt to perform retina blood vessel segmentation with U-net and try to remove blood vessels from DR images. Unfortunately, we can't go any further due to the limited time and limited size of the dataset.

Data

We use the diabetic retinopathy image data from Kaggle, which has a total size of ~8GB and 35126 images. Sample images

Figure 1: Sample retina images and their ICDRS grade taken from (Gulshan et al. 2016).

from the data set are shown below:

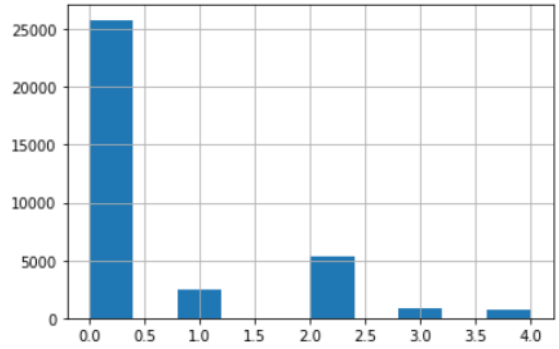
- (https://www.kaggle.com/tanlikesmath/diabetic-retinopathy-resized), which is resized from the website (https://www.kaggle.com/c/diabetic-retinopathy-detection/data)

A subset of data is shown below. Each image has a corresponding image_name and level, which is its ground-truth value of DR severity grading. In the dataset, there are 25810 images of no DR, 2443 images of mild DR, 5292 images of moderate DR, 873 images of severe DR, and 708 images of proliferative DR.

	image	level	image_name
0	10_left	0	10_left.jpeg
1	10_right	0	10_right.jpeg
2	13_left	0	13_left.jpeg
3	13_right	0	13_right.jpeg

Sample Data

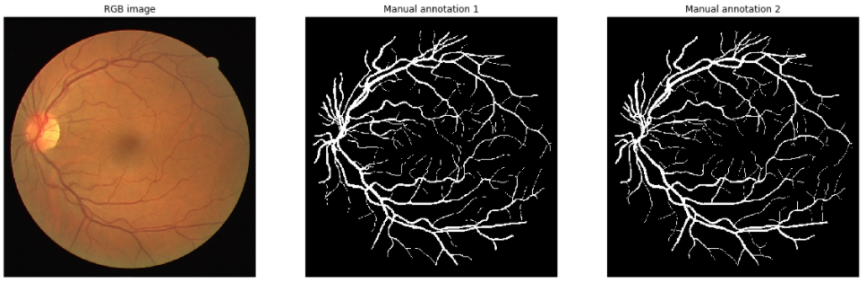
```
0    25810
2     5292
1     2443
3      873
4      708
```



Value counts vs Histogram

Moreover, for blood vessel segmentation, we use the DRIVE dataset. DRIVE or Digital Retinal Images for Vessel Extraction is a minimal dataset that only contains 40 images. It has been divided into a training and a test set, each including 20 images. For each training input, we have one color input, a binary outline, and a manual vessel mask. Samples are shown below:

- (https://drive.grand-challenge.org/)

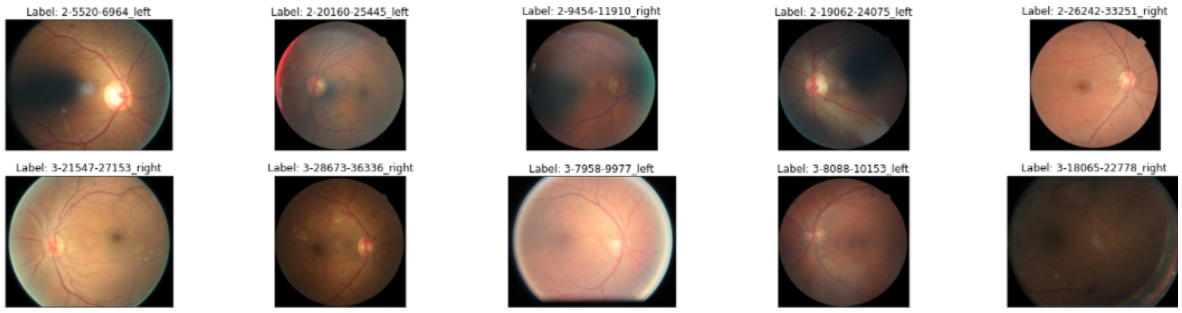


Proposed Approach

- **Data Processing**

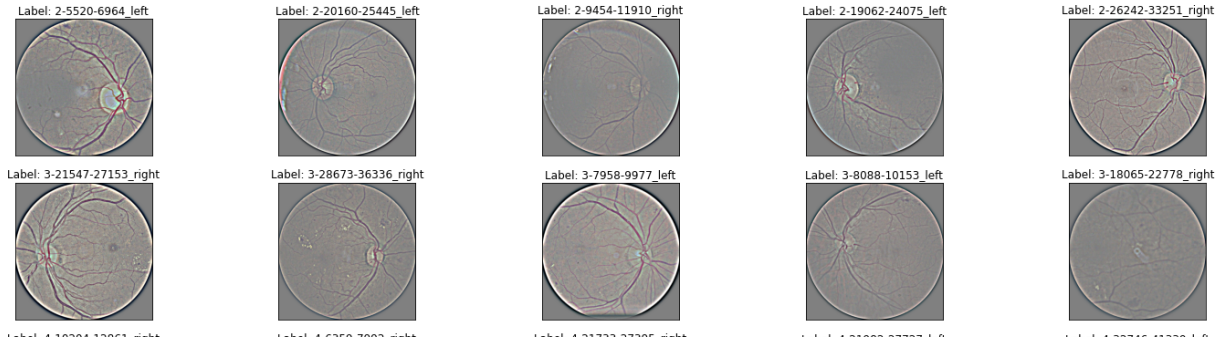
  When we explored retina images in the dataset, we found several issues of images that make it hard for us humans to spot DR severity. First, illuminations for images are different, which leads some images to appear bright and others very dark. Second, some retina images have dark areas that do not contain any useful information for DR severity level labeling.

  Original images



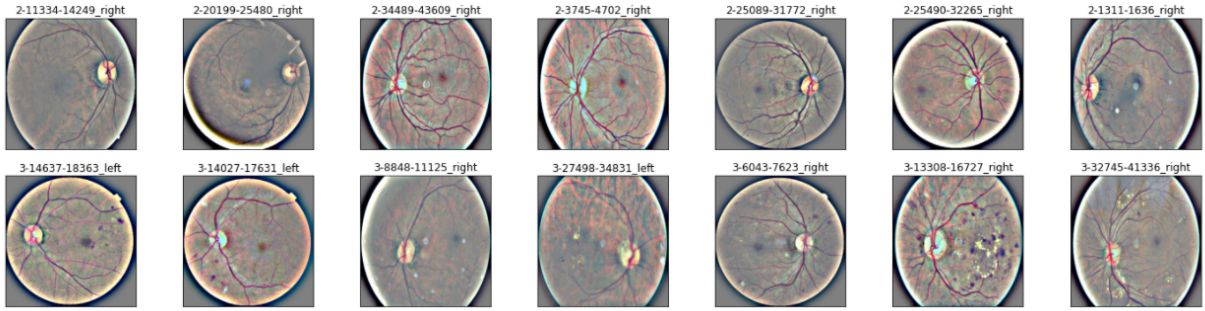
  To solve these two issues, we decided to use gray scale for all image representation, and create a circular crop around the image centre to remove uninformative dark areas.

After circle crop



After trying the gray scale for image representation, we found that the illumination for images was too low. Hence, we proceeded to develop a colored circle crop. For the colored crop version, we used the function load_ben_color proposed in Eye Preprocessing in Diabetic Retinopathy. This function resizes the images to a size of 300 x 300 x 3, uses argument sigmaX = 10 for cv2.GaussianBlur, and normalizes the images in the end.

After colored circle crop



- **Models**

EfficientNet (224 x 224)

In 2019, Tan and Le proposed EfficientNet to improve the efficiency of ConvNets. They used a new scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. This significantly reduces the number of parameters in the framework and fastens its training speed. In this case, we adopted EfficientNetB3 as our first trial. Later, due to the limitation of computational sources, we had to transfer to EfficientNetB0 to run more epochs. In the final model, we removed the top of EfficientNetB0 and rewrote the dense layer to predict a 1*5 vector for each image with a softmax activation function.

Finally, we picked up Adam optimizer with a 0.0001 learning rate, epochs = 10, and batch_size = 32 to train our models. For more details, please look at notebook eficientnet.ipynb .

Convolutional Neural Network (ConvNet)

ConvNet is a typical deep learning model which takes in input images and becomes able to classify different images by assigning learned weights to various parts in each image.

The model has 3 convolutional layers and each layer's filter size decreases by half from 128 to 32. Each filter has a size of 5x5. Maximum pooling layer of size 3x3 is added to each layer. This pooling layer helps reduce the size of the output profile of convolutional layers. ReLU (rectified linear unit) activation with a dropout value of 0.25 is added to each of 3 convolutional layers. This activation layer helps standardize the output of each layer to avoid overfitting.

The model is compiled with Adam optimizer with a learning rate of 0.001, and it is trained for 5 epochs and a batch size of 32. Please refer to ConvNet.ipynb for more details.

InceptionV3

InceptionV3 is the third iteration of Google's Inception CNN architecture developed, which features effective classification performance with its large number of layers while keeping the number of parameters and computation complexity low. An InceptionV3 model pre-trained on ImageNet is available in PyTorch, which is suitable as a first trial for the DR dataset to get a baseline accuracy. WIth a pre-trained model, we can reuse the feature extraction layers of the model and only need to fine-tune the model, which saves a lot of time.

The pre-trained model is fine-tuned using Adam optimizer with a learning rate of 0.0001 on 15 epochs. A batch size of 32 is used. The auxiliary output of InceptionV3 is also used when calculating cross-entropy loss, which has a weight of 0.4. For more details, see inception_v3.ipynb

VGG16 & VGG19

VGGNet is one of the significant classification models invented by the Visual Geometry Group from Oxford (VGG, 2015). VGGNet has remarkably improved over previous classification models and got second place in LSVRC-2014. VGGNet has various versions, and we picked two representative versions, VGG16 and VGG19, which also have relatively high performance than the rest. VGGNet carries some traditional set-up from AlexNet and ZFNet, such as using ReLU as the activation function for each unit and having fully-connected layers at the end connecting with the output layer. However, VGGNet stacks more layers onto AlexNet and further uses smaller filters (2x2 and 3x3) instead of size filters (11x 11 and 5x5). Particularly, VGG16/VGG19 has 16/19 layers, including 13/16 convolutional and three fully-connected layers. Thus, VGGNet has more than twice the number of layers that AlextNet has, which provides enough complexity to handle multi-task classification.

For training, we decided to fine-tune VGG pre-trained model on our datasets using transferring learning, which saves us lots of time to train our model. We assumed that the low-level features should be helpful in all types of image classification, but some of the high-level features are irrelevant to our task. Moreover, we replaced output layers from 1000 units to 5 units and froze the upper half of layers' weight, and only trained on the rest with Adam optimizer in a relatively low learning rate = 0.0001. We picked epochs = 10 and batch_size = 32 as parameters to train both models. For more details, please look at notebook vgg.ipynb.

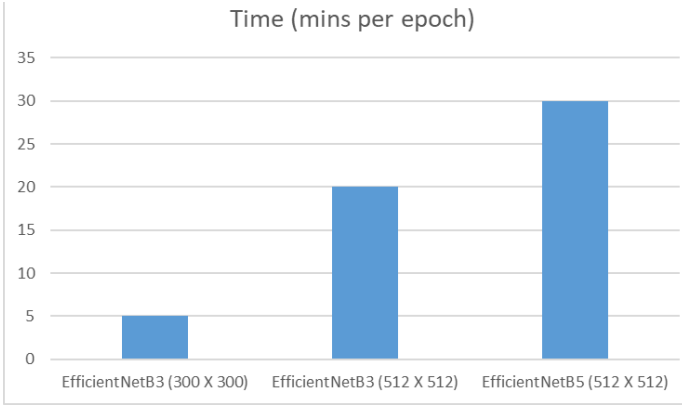
References: loading data, visualization, VGG vs. Alex Net, and Intro to VGG

DensetNet 121 & DenseNet 169

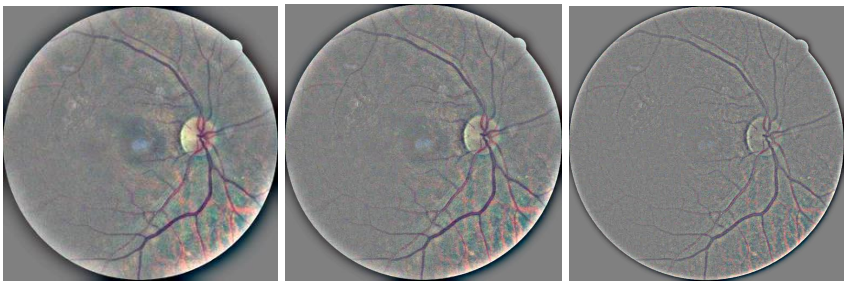
DensetNet is a neural network that connects all the layers directly in each dense block. DenseNet is one of the modern classification models introduced during the class, which has better performance than VGG in deep versions such as DenseNet-169 and DenseNet-201. In this case, besides implementing transferring learning, We decided to train on the raw model; however, the training/validation accuracy never converged even if I picked the adaptive learning rate method with a relatively low learning rate (lr = 10^-4) and significant epoch. Likewise, we tried to freeze several layers to compare each result. The models still converge to ~73%, which predicts all the input images as dominated labels in the dataset. Our group had already tried almost all the models and performed several good data progressing on our dataset but still got poor results. Since it won't be many good choices left, we decided to increase the input size of images to see how it goes. For more details, please check out densenet121_169.ipynb

EfficientNetB3/B5 (512 X 512)

EfficientNetB3, as we mentioned in the previous section, is a model that has a good trade-off between high accuracy and efficiency. It's also one of the most popular classification models and has lots of well-implemented libraries. We picked EfficientNet-Pytorch to construct our models since it's simple to use and supports all the types of EfficientNet models from B0 to B7. On the other hand, training a complex model with high-resolution input images is painful. It's impossible to train a model on online servers like the collab in 24 hours. One of our members has a high-performance laptop, so the rest of the training was on his computer. The time comparison between training three different types of models is shown below:



Each (512 x 512) model took about half-day to train on the local machine with shallow batch sizes <= 10. We used all the data preprocessing mentioned above and heavy data argumentation such as randomcrop, blur, horizontalflip, etc. You can see that in the report. We did transfer learning with a low learning rate lr = 3e-5 and L2 coefficient = 5e-4. Instead of using cross-entropy, this time, we used MSE as our loss function and a piecewise function to map output valueS to expected labels. The outcomes are great, and both EfficientNetB3 and EfficientNetB5 have 80 overall accuracies. This somehow makes sense by looking at the input image closely.

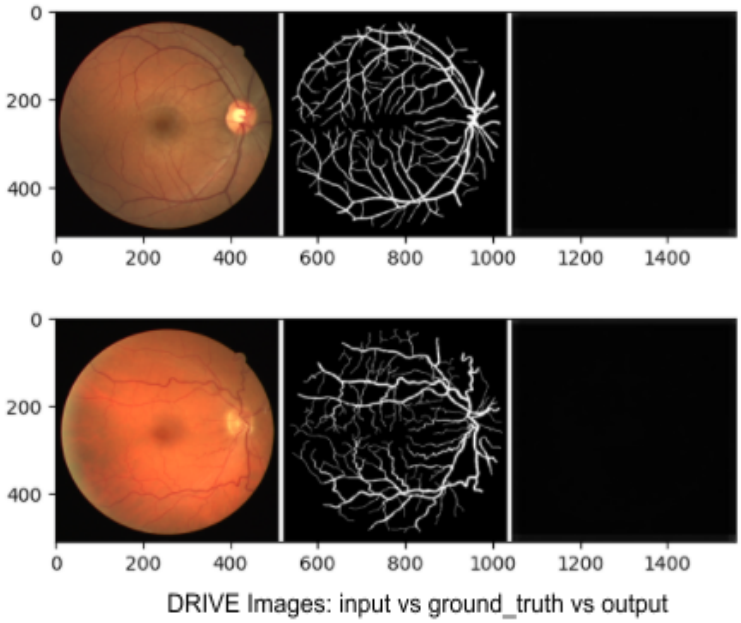


Input Images: 300x300 vs 512x512 vs 1024x1024

The images above are preprocessed images in different resolutions. We resize them into the same size. We can see downsampling input images does wash away lots of details or information. For more information, please check out [EfficientNet_acc80.ipynb]

Retina blood vessel segmentation with U-net

Followed by previous thoughts. We found that most eyeballs beyond the moderate stage (label >= 3) displayed tiny shadows and speckles, but the blood vessel covers those details and makes the image hard to identify; hence, we found a specific dataset called DRIVE, Digital Retinal Images for Vessel Extraction. Based on this dataset, we planned to train a vessel segmentation model as one of our data preprocessing methods to explore whether we can get a better training model. No surprises. It failed. Based on the Kaggle team's notebook, Image segmentation with a U-Net-like architecture, we tried Unet and Unet-Xception-style models with smaller kernel sizes in each conv layer but still got poor results. The learning curve of IoU and loss are shown in the Results section.



DRIVE Images: input vs ground_truth vs output

The set of 20 images is not enough for fitting a model. If we have more time, we will further perform data argumentation plus self-surprised learning to generate more images to train.

Results

- **Initial Results**

Models	Image Size	Epochs	Batch Sizes	Learning Rates	Accuracies (Validation)
Inception V3	299 X 299	10	32	0.0001	0.7398
ConvNet	224 X 224	5	32	0.0001	0.7348
EfficientNet	224 X 224	5	32	0.0001	0.7035
VGG 16	224 X 224	10	32	0.0001	0.7543
VGG 19	224 X 224	10	32	0.0001	0.7571

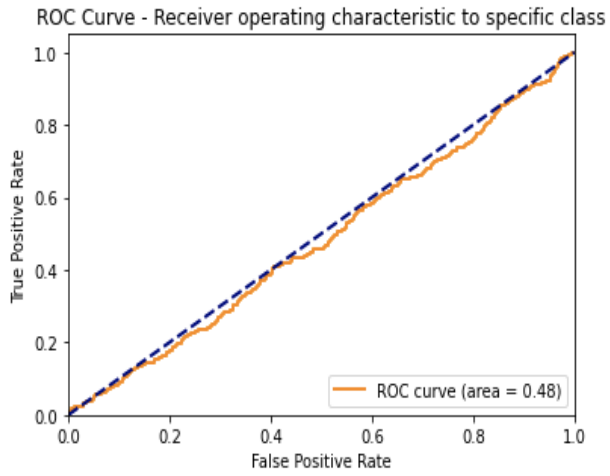
- **Final Results**

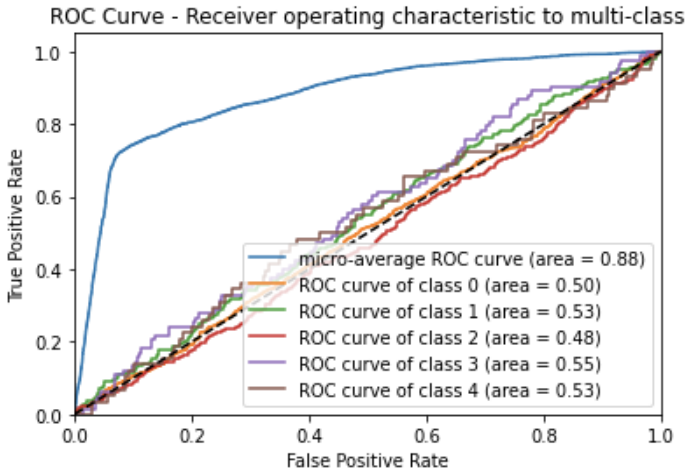
Models	Image Size	Train | Validation | Test Size	Epochs	Batch Sizes	Learning Rates	Accuracies (test)	Precision	Recall	F1
Inception V3	299 X 299	24575|7022|3511	15	32	0.0001	0.7337	0.6011	0.7337	0.6283
ConvNet	300 X 300	24575|7022|3511	5	32	0.001	0.7528	0.5667	0.7528	0.6466
EfficientNet	224 X 224	24575|7373|3160	10	32	0.0001	0.7117	0.5679	0.7117	0.6371
VGG 16	224 X 224	24575|7022|3511	10	32	0.0001	0.6641	0.5649	0.6641	0.6014

VGG 19	224 X 224	24575|7022|3511	10	32	0.0001	0.6874	0.5435	0.6874	0.5998
DenseNet121	300 x 300	24575|7022|3511	15	20	0.0001	0.5770	0.5778	0.5770	0.5730
EfficientNetB3	512 X 512	24575|7022|3511	25	10	3e-5	0.80	0.82	0.8	0.8
EfficientNetB5	512X512	24575|7022|3511	20	6	3e-5	0.80	0.83	0.80	0.81

- **EfficientNet (224 x 224)**

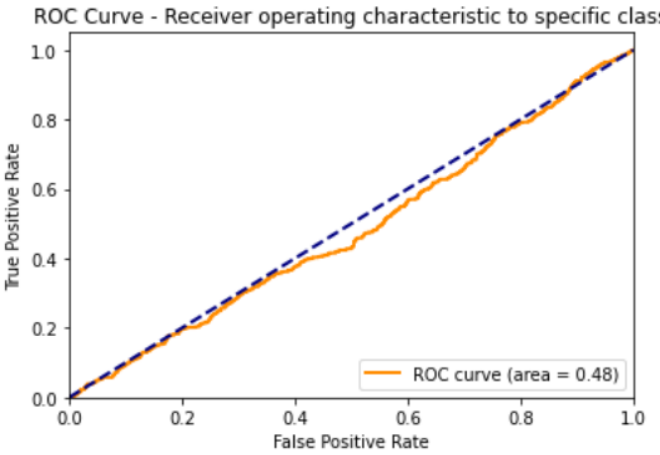
```
classification_report:
---------------------------------------
              precision    recall  f1-score   support

           0       0.74      0.96      0.83      2336
           1       0.00      0.00      0.00       231
           2       0.15      0.03      0.05       452
           3       0.00      0.00      0.00        83
           4       0.00      0.00      0.00        58

    accuracy                           0.71      3160
   macro avg       0.18      0.20      0.18      3160
weighted avg       0.57      0.71      0.62      3160
```

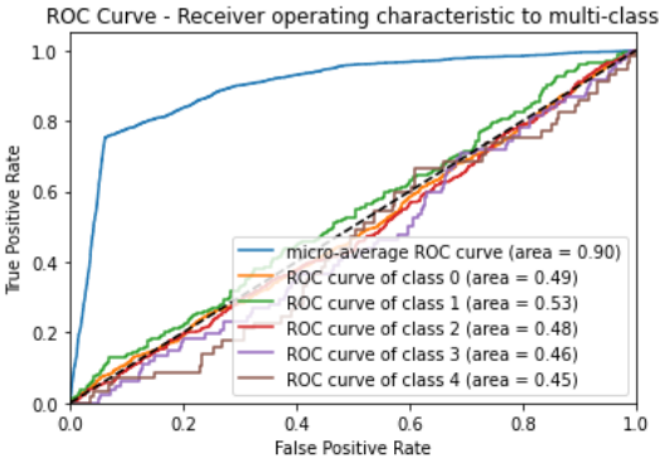




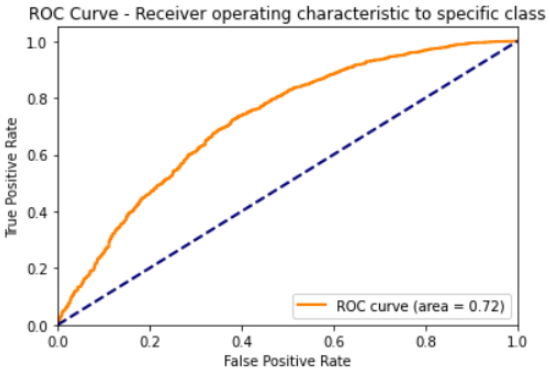
- **ConvNet**

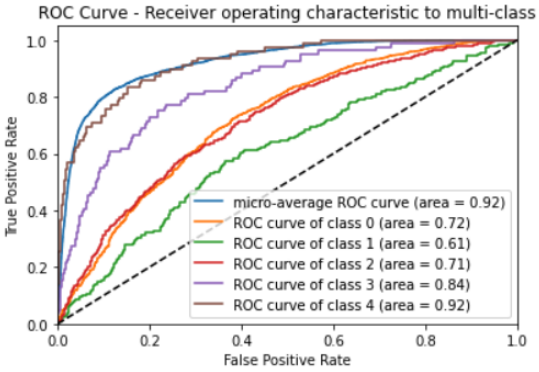
```
classification_report:
---------------------------------------
              precision    recall  f1-score   support

           0       0.75      1.00      0.86      2643
           1       0.00      0.00      0.00       243
           2       0.00      0.00      0.00       486
           3       0.00      0.00      0.00        82
           4       0.00      0.00      0.00        57

    accuracy                           0.75      3511
   macro avg       0.15      0.20      0.17      3511
weighted avg       0.57      0.75      0.65      3511
```





- **InceptionV3**

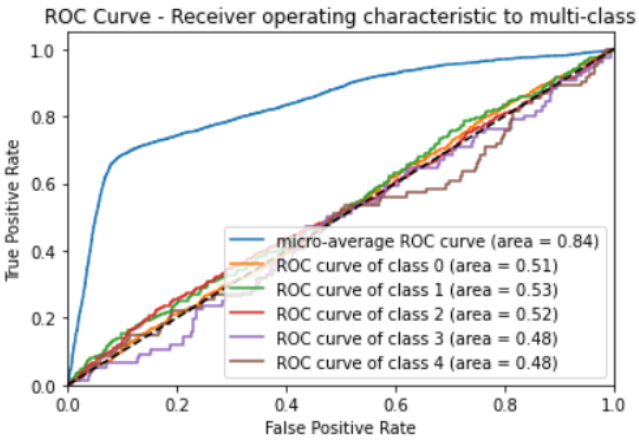





```
classification_report:
---------------------------------------
              precision    recall  f1-score   support

           0       0.74      1.00      0.85      2571
           1       0.00      0.00      0.00       231
           2       0.24      0.02      0.03       546
           3       0.00      0.00      0.00        84
           4       1.00      0.03      0.05        79

    accuracy                           0.73      3511
   macro avg       0.40      0.21      0.19      3511
weighted avg       0.60      0.73      0.63      3511
```

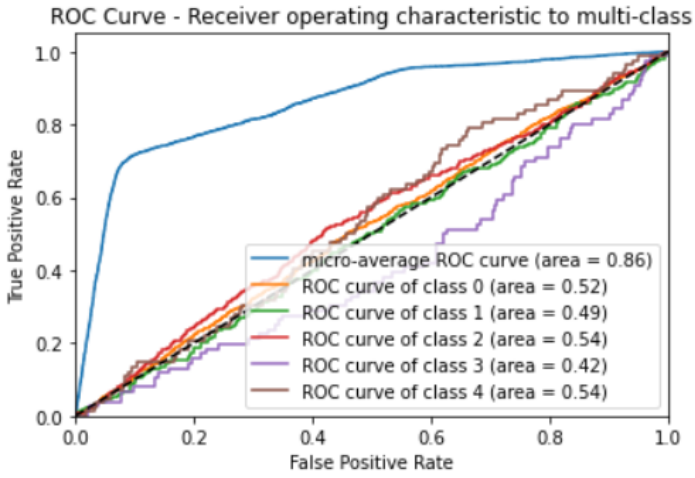
- **VGG 16**

```
classification_report:
---------------------------------------
              precision    recall  f1-score   support

           0       0.72      0.90      0.80      2532
           1       0.11      0.01      0.02       258
           2       0.21      0.09      0.13       565
           3       0.02      0.01      0.02        76
           4       0.02      0.01      0.02        82

    accuracy                           0.66      3513
   macro avg       0.22      0.21      0.20      3513
weighted avg       0.56      0.66      0.60      3513
```



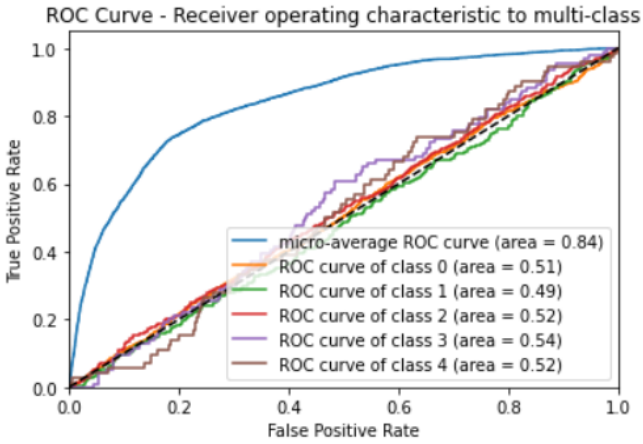
- **VGG 19**

```
classification_report:
---------------------------------------
              precision    recall  f1-score   support

           0       0.72      0.94      0.82      2532
           1       0.00      0.00      0.00       258
           2       0.15      0.04      0.06       565
           3       0.00      0.00      0.00        76
           4       0.00      0.00      0.00        82

    accuracy                           0.69      3513
   macro avg       0.17      0.20      0.18      3513
weighted avg       0.54      0.69      0.60      3513
```



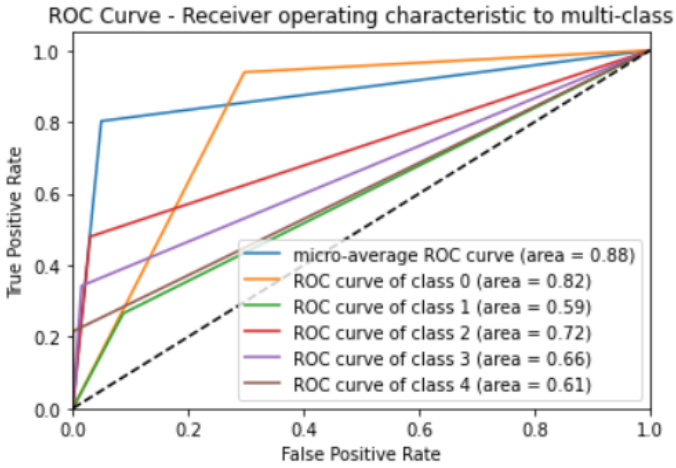
- **DenseNet121**

```
classification_report:
---------------------------------------
              precision    recall  f1-score   support

           0       0.75      0.72      0.73      2601
           1       0.00      0.00      0.00       243
           2       0.16      0.31      0.21       501
           3       0.00      0.00      0.00        94
           4       0.12      0.01      0.02        72

    accuracy                           0.58      3511
   macro avg       0.21      0.21      0.19      3511
weighted avg       0.58      0.58      0.57      3511
```



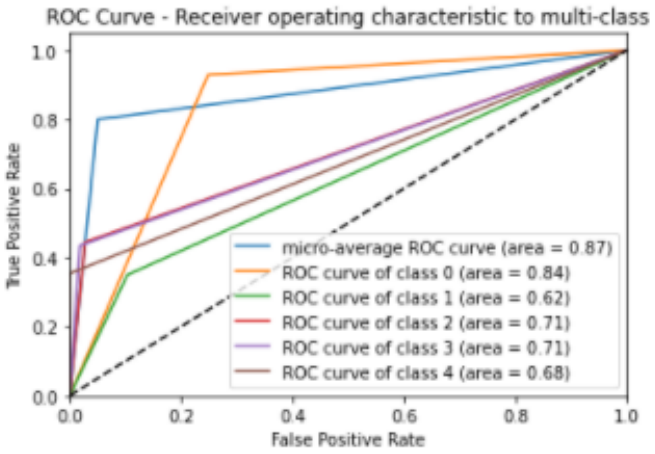
- **EifficientNetB3 (512 X 512)**

```
classification_report:
---------------------------------------
              precision    recall  f1-score   support

           0       0.90      0.94      0.92      2633
           1       0.17      0.27      0.21       226
           2       0.73      0.48      0.58       511
           3       0.33      0.34      0.34        76
           4       0.82      0.22      0.34        65

    accuracy                           0.80      3511
   macro avg       0.59      0.45      0.48      3511
weighted avg       0.82      0.80      0.80      3511

QuadraticWeightedKappa (test): 0.7779714189838315
```

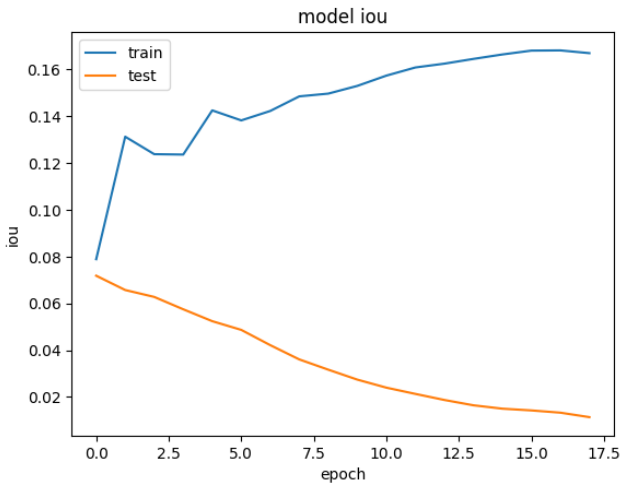


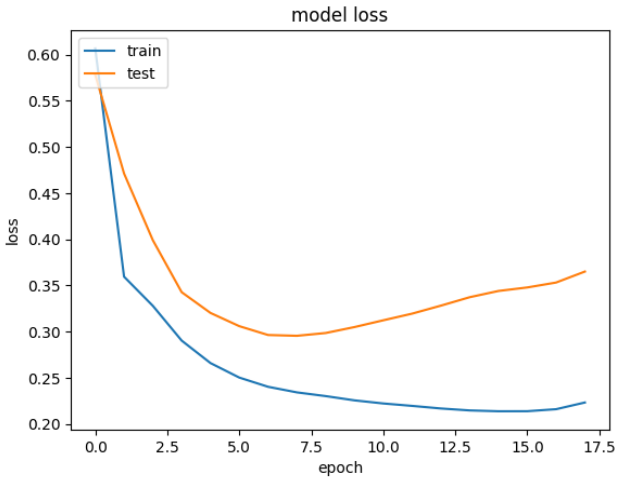
- **EificientNetB5 (512 X 512)**

```
classification_report:
---------------------------------------
              precision    recall  f1-score   support

           0       0.92      0.93      0.92      2633
           1       0.19      0.35      0.24       226
           2       0.73      0.44      0.55       511
           3       0.35      0.43      0.39        76
           4       0.96      0.35      0.52        65

    accuracy                           0.80      3511
   macro avg       0.63      0.50      0.52      3511
weighted avg       0.83      0.80      0.81      3511

QuadraticWeightedKappa (test): 0.7921888933978357
```



- **U-Net Xception-style model**



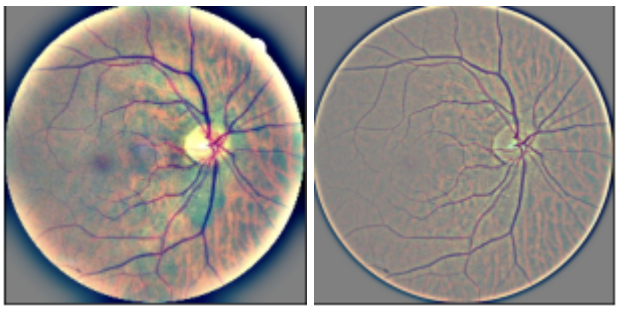


Discussion & Conclusion

Our models, which take 300 x 300 resolution images as input, all reached accuracies over 0.70 but could not break the bottleneck further. We found that the models were more accurate in predicting Class 0 and Class 1, but were almost unable to predict Class 3 and Class 4. The ROC curves also showed us similar patterns. We can see a very high micro-average ROC curve, but for each class, especially Class 3 and Class 4, the ROC curves are not that impressive. We first thought that this may be due to the imbalance of the dataset since Class 0 almost accounts for 73% of the whole dataset. However, after using a stratified split for the whole dataset, the results did not improve that much. We also tried to adjust the structure of our models, such as letting our models fit the dataset without pre-trained hyperparameters, adjusting loss functions, and the corresponding evaluation criteria, but these efforts did not work as well.

Moreover, we tried to apply more data-augmentation techniques onto our dataset, such as transferring gray images into colored versions or using circle crop to keep the eyeballs a perfect circle, and we found that resolution of the images can be an important influencing factor. For example, the following two images are actually the same eyeball and the 2 images were both preprocessed with circle cropping techniques. The only difference is that the left image is 300*300 pixels and the right one is 800*800 pixels. Obviously, the right image can be easier for models to learn useful features. However, the high-resolution images took too much RAM so that we were unable to finish training on Google Colab. Lucky, we did train models with high-resolution images and had our final accuracies reach 80%.

We thought image resolution would be the main reason that prevented us from getting better results. We were looking for lesions in the eyeballs and the main lesions were blocked vessels which were shown as red dots in the images. However, in 224*224 inputs, even we ourselves were hard to tell if certain features were associated with the lesions, let alone our models.



In conclusion, we tried 5 different CV models (Vgg, Conv, DensetNet, EfficientNet, Inception) and applied these models to our dataset. Although the Diabetic retinopathy competition was over six years ago, comparing our metrics with leaderboard score, we got the top 15 scores. This can be a good start. With higher-resolution images and more computational resources, we believe deep learning models can be a great help in future healthcare areas.

Statement of individual contribution

As a team: To share our code, we created a repository on github and uploaded our notebooks to the repo. To keep track of each other's progress, we meet on Thursday night every week. In the early stages of the project, we discussed appropriate models for the DR severity level grading task, and each member chose one to train. Then, we shared initial models results and addressed questions about training. After progress report submission, we researched and applied good image processing approaches, optimized our models further, and reported each model's performance on the test data by plotting ROC curve and confusion matrix.

Yunchun Pan: Process retina images by adding color scales, improving image lighting conditions, and cropping center images. Responsible for developing the Convolutional Neural Network Model (ConvNet) for the DR severity level grading task. Adjust convolutional filter size, kernel size, and max pooling filter size. Fine-tune batch size, learning rates, and steps per training epoch. Handle validation loss rising problem by adding activation methods and trying different dropout values. Plot the ROC curves and confusion matrix for different DR severity level labeling.

Hongshuo Wang: Responsible for constructing and training a DR severity level classifier using transfer learning via VGG, DenseNet, and EfficientNet pre-trained model. Comparing outcomes between VGG16, VGG19, DenseNet 121, DenseNet 169, EfficientNetB3 (512 X 512) and EfficientNetB5 (512 X 512). Fine Tuning models to meet project goals, including changing the structure of previous models, tuning parameters such as batch size and learning rate, and choosing different kinds of optimizers. Helping teammates to create a proper virtual environment and download necessary packages for our projects. Exploring Retina blood vessel segmentation with U-net to further improve Dr detection models.

Zhengxi Li: Responsible for building a DR severity level classifier based on a pre-trained InceptionV3 model. Tuning parameters such as batch size and learning rate to achieve maximum accuracy. Explored data augmentation methods such as weighted grey-scale to see whether it helps with overall accuracy. Also tried to deal with the class imbalance of data through oversampling minority classes.

Peng Zhang: Responsible for applying a pre-trained EfficientNet to solve the DR severity level grading task. Tuning necessary parameters such as the number of layers and the structure of each layer in the model. Plot the ROC curves and confusion matrix for different DR severity level labeling. Try circle crop to produce high-resolution and better images for models to learn. Try to deal with imbalanced data through stratified division.

References

Foo, A., Hsu, W., Lee, M. L., Lim, G., & Wong, T. Y. (2020). Multi-task learning for diabetic retinopathy grading and lesion segmentation. Proceedings of the AAAI Conference on Artificial Intelligence, 34(08), 13267–13272. https://doi.org/10.1609/aaai.v34i08.7035

Persson, A. (2021, May). Diabetic retinopathy detection. Kaggle. Retrieved December 16, 2021, from https://www.kaggle.com/c/diabetic-retinopathy-detection/discussion/242755

Kloosterman, B. (2020, June 3). Tensorflowcollab/capsulenet_tf2_mnist.ipynb at master · BKLOOSTE/tensorflowcollab. GitHub. Retrieved November 11, 2021, from https://github.com/bklooste/tensorflowcollab/blob/master/CapsuleNet_tf2_MNIST.ipynb.

Raj, B. (2020, July 31). A simple guide to the versions of the inception network. Medium. Retrieved November 11, 2021, from https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202.

Siyah, B. (2019). *ImageNet winning CNN Architectures (ILSVRC): Data Science and Machine Learning*. Kaggle. Retrieved November 12, 2021, from https://www.kaggle.com/getting-started/149448.

Ilovescience. (2019, May 8). Diabetic retinopathy (resized). Kaggle. Retrieved November 11, 2021, from https://www.kaggle.com/tanlikesmath/diabetic-retinopathy-resized.

Roychowdhury, D. S. (2021, April 14). Top 10 projects for beginners in Computer Vision and medical imaging. Medium. Retrieved November 11, 2021, from https://towardsdatascience.com/top-10-projects-for-beginners-in-computer-vision-and-medical-imaging-c138a646e44e.

Tan, M. & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (cite arxiv:1905.11946Comment: Published in ICML 2019)

TOMAR, N. I. K. H. I. L. K. U. M. A. R. (2020, May 16). Polyp segmentation using UNET in tensorflow 2.0. Idiot Developer. Retrieved December 16, 2021, from https://idiotdeveloper.com/polyp-segmentation-using-unet-in-tensorflow-2/

Siddique, N., Paheding, S., Elkin, C. P., & Devabhaktuni, V. (2021). U-Net and its variants for Medical Image Segmentation: A review of theory and applications. IEEE Access, 9, 82031–82057. https://doi.org/10.1109/access.2021.3086020

Copied, J. (2019, September 9). Aptos : Eye preprocessing in Diabetic Retinopathy. Kaggle. Retrieved December 16, 2021, from https://www.kaggle.com/ratthachat/aptos-eye-preprocessing-in-diabetic-retinopathy#3.A-Important-Update-on-Color-Version-of-Cropping-&-Ben's-Preprocessing

Staal, J., Abramoff, M. D., Niemeijer, M., Viergever, M. A., & van Ginneken, B. (2004). Ridge-based vessel segmentation in color images of the retina. IEEE Transactions on Medical Imaging, 23(4), 501–509. https://doi.org/10.1109/tmi.2004.825627

Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, .

Team, K. (2019, March 20). Keras Documentation: Image segmentation with a u-net-like architecture. Keras. Retrieved December 16, 2021, from https://keras.io/examples/vision/oxford_pets_image_segmentation/

MADER, K. S. C. O. T. T. (2017, November 3). CapsuleNet on Mnist. Kaggle. Retrieved November 11, 2021, from https://www.kaggle.com/kmader/capsulenet-on-mnist.

Porwal, P., Antiga, L. (2018, February 26). Orobix/retina-unet: Retina blood vessel segmentation with a convolutional neural network. GitHub. Retrieved December 16, 2021, from https://github.com/orobix/retina-unet

Pachade, S., Kamble, R., Kokare, M., Deshmukh, G., Sahasrabuddhe, V., & Meriaudeau, F. (2018). Indian Diabetic Retinopathy Image Dataset (IDRiD): A Database for Diabetic Retinopathy Screening Research. Data, 3(3), 25. MDPI AG. Retrieved from http://dx.doi.org/10.3390/data3030025

Ronneberger, O., Fischer, P. & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation (cite arxiv:1505.04597Comment: conditionally accepted at MICCAI 2015)

Prasanna Porwal, Samiksha Pachade, Ravi Kamble, Manesh Kokare, Girish Deshmukh, Vivek Sahasrabuddhe, Fabrice Meriaudeau. (2018). Indian Diabetic Retinopathy Image Dataset (IDRiD). IEEE Dataport. https://dx.doi.org/10.21227/H25W98.

Pailla, S. (2021, November 5). MACHINEVISION_CAPSNET_DR. Kaggle. Retrieved November 11, 2021, from https://www.kaggle.com/swethapailla/machinevision-capsnet-dr.

Tsang, S.-H. (2020, October 19). Review: Vggnet - 1st runner-up (image classification), winner (localization) in ILSVRC 2014. Medium. Retrieved November 12, 2021, from https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11.

Sharmaml, V. (2020, April 25). VGG19 DR. Kaggle. Retrieved November 12, 2021, from https://www.kaggle.com/varunsharmaml/vgg19-dr.

Sindhupriyak. (2021, October 13). Diabetic retinopathy(pytorch). Kaggle. Retrieved November 12, 2021, from https://www.kaggle.com/sindhupriyak/diabetic-retinopathy-pytorch.

Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). PMLR.

Qian, Y., & Woodland, P. C. (2016). Very deep convolutional neural networks for robust speech recognition. 2016 IEEE Spoken Language Technology Workshop (SLT). https://doi.org/10.1109/slt.2016.7846307.

Inkawhich, N. (n.d.). Finetuning Torchvision Models. Google Colab. Retrieved December 3, 2021, from https://colab.research.google.com/github/pytorch/tutorials/blob/gh-pages/_downloads/finetuning_torchvision_models_tutorial.ipynb?authuser=1#scrollTo=YP5ig6ESfiIN