Group 17
Histopathological Cancer Detection

Sayak Kundu, Vaibhav Tiwari, and Krish Vadodaria

***Abstract*—Cancer is one of the deadliest diseases faced by mankind today, largely because of its mortality rate and cost of treatment. Accurate diagnosis of the tumor is a pivotal factor in determining the course of treatment and severity of the illness. Detection of metastatic cancer in pathological scans using machine learning and neural networks is a topic of great interest and use in medical diagnosis. This project explores various machine learning algorithms as potential models for cancer detection: Logistic Regression, Convolutional Neural Network, MobileNetV2 and ResNet50.**

I. Introduction

Among the many steps in cancer identification, one conclusive test involves sampling tissue from suspicious lumps and observing it under a microscope to ascertain if it is malign. Manual analysis of slides under a microscope is a painstaking and time-consuming task, often prone to human error. This project aims to explore machine learning and neural networks as a tool for accurate detection of metastatic cancer. We use the PatchCamelyon dataset which contains histopathologic scans of lymph node sections of size (96 x 96 px) which are labelled either 0/1 for absence/presence of cancer which we are trying to predict.

We conducted experiments with three different deep learning models and performed data augmentation on the input dataset to add regularization, making it more robust to unseen data. We also performed test-time augmentation to further improve model efficiency. The CNN model from scratch and models built from ResNet50 and MobileNetV2 were trained with 3 different methods of setting learning rates across epochs:

- **Constant learning rate:** Fixed learning rate across all epochs.
- **One Cycle Policy:** Learning rate is initially gradually increased on a slope upto a maximum value, and then reduced. Extremely low learning rate applied in the final epochs.
- **Reduce Learning Rate on Plateau:** Learning rate reduced by a fixed factor when model's validation accuracy does not improve over some epochs.

We have used TensorFlow and Keras API[10] to implement all the models.

II. Related Work

The work of [12] performed a detailed study of breast cancer histology image classification using transfer learning[8]. In this work Vesal et al. used ResNet50 and Inception V3 as the backbone of their transfer learning model and they classified the images as normal, benign, in situ carcinoma, or invasive carcinoma. Using ResNet50 they were able to achieve 97.50% of accuracy on test dataset. Prior to this work Kothari et al. tried to solve similar problem using support vector machine in [4]. Antonio De Perio in [6] has tried to solve the same problem using ResNet50 transfer learning model. He used fast.ai to implement his model and achieved 98.6% of accuracy on validation dataset.

III. Dataset and Data Augmentation

Our approach utilizes the PatchCamelyon dataset available on Kaggle[3]. It contains 220,025 histopathologic scans of lymph node sections, each image (96×96px) annotated with a binary label indicating presence of cancer tissue. Of this, 130,908 samples have a '0' label which imply absence of metastatic tissue, and 89,117 samples have a '1' label, expressing that metastatic tissue has been detected. This distribution, although slightly skewed, is acceptable for the models we have experimented with. Further, attempting to balance the dataset by reducing the number of '0' images leads to a reduction in the size of the dataset, causing a reduction in model efficiency and accuracy. We split the data as 70/20/10 for purposes of train, validation and test respectively.

Data augmentation is an effective strategy adopted in image classification problems to introduce regularization in the model. Keras is equipped with an 'ImageDataGenerator' function, that performs real-time data augmentation. This function returns a generator that gives a batch of augmented data in every iteration. One limitation of this function is that if rotation is enabled as one of the augmentation techniques, the image is randomly rotated by an angle in a specified range, which is undesirable for our purposes as it can destroy the scan properties which lead to accurate categorization. To counter this issue, we wrote a custom function that only rotates the image by 90°, 180°, or 270°. This function also injects 2% noise in the image, which is expected to introduce further regularization, and improve model performance. These are the image alterations we enabled in our models: **horizontal flip, vertical flip, rotation, and noise addition**.

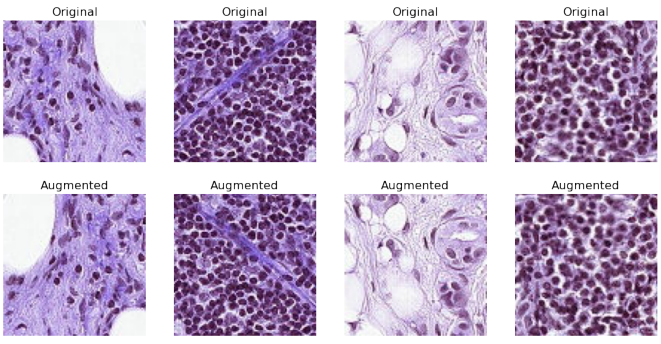


Fig. 1: Samples of data samples and augmented counterpart.

Another technique we adopted to improve model efficiency was **test-time augmentation**. During testing, we predict on five different augmentations of each test sample, and follow a voting process to ascertain the final classification of the image.

IV. Methods

A. Logistic Regression

Before moving forward with deep neural network models, the first natural step was to implement the simplest classification model: logistic regression. This would provide a reference to compare our complex models against. Further, it also justifies venturing into the domain of neural networks, which would not make sense if a basic model could achieve accurate results. The input to the model is a 1D array containing the RGB values of each pixel ($96 \times 96 \times 3 \equiv 27{,}648$ features). The model was optimized using a Stochastic Gradient Descent optimizer, with L2 regularization and binary cross entropy loss function. This model is intended to act as a baseline, and since the results of the initial model were highly unsatisfactory, no hyper-parameter tuning was attempted.

B. Convolutional Neural Network

We implemented a CNN model[2] similar to AlexNet[5]. AlexNet is one of the earliest and most influential papers published in computer vision that employs CNNs and GPUs for accelerated deep learning. The model we implemented uses the same basic building blocks as AlexNet (convolution, maxpooling, and fully connected layers). However, it is deeper than AlexNet, and uses smaller kernel and filter sizes. This enables the model to work with a smaller input size, eventually resulting in fewer parameters (∼61,000,000 in AlexNet vs 1,661,186 in CNN). The use of ReLU activation in each layer is a direct implication of AlexNet's inferences, which shows that ReLU performs considerably better than sigmoid and tanh. Our experiments with different activation functions were in conjunction with this observation. We also experimented with different dropout ratios before converging on 0.3 as the ideal value. We reported this model's metrics with different learning rate approaches.

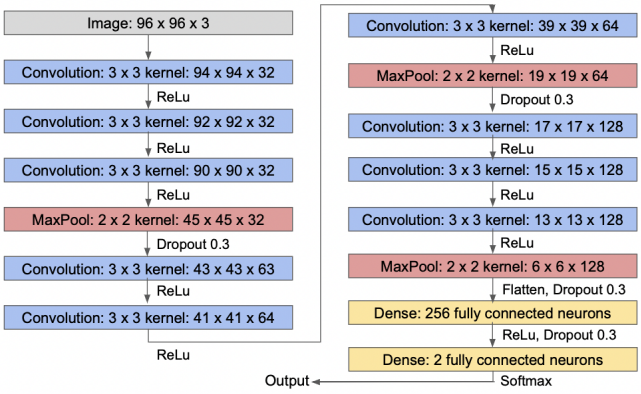
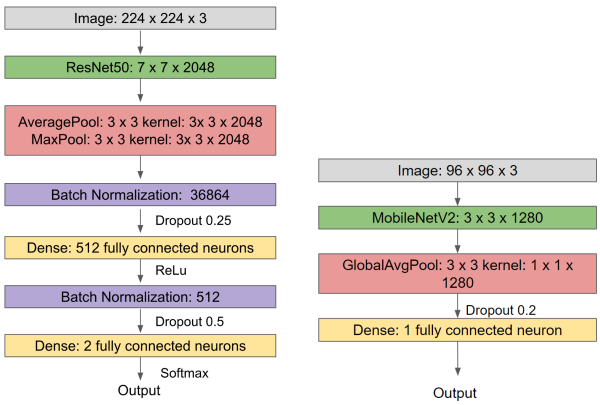


Fig. 2: CNN Model Architecture



(a) ResNet50 Architecture (b) MobileNetV2 Architecture

Fig. 3: Transfer learning models

C. Transfer Learning with MobileNetV2

In the pursuit of improved performance of convolutional neural networks on mobile devices, Google proposed MobileNetV2[7], [10] in 2018. The depthwise separable convolution is introduced which dramatically reduced the complexity cost and model size of the network allowing latency reduction in mobile devices or any other device with low computational power. The depthwise convolution, performs lightweight filtering by applying a single convolutional filter per input channel followed by a pointwise convolution layer without any non-linearity which is responsible for building new features through computing linear combinations of the input channels. In our model we prepend a preprocessing layer before the base mobileNet model and append a global average pooling layer, a dropout layer and a fully connected layer generating a single output. While compiling the model we use Adam optimizer and binary cross entropy loss. The model is then trained for 30 epochs initially while having all the layers of the base mobileNet model frozen[11]. Then we fine-tune the model by unfreezing some top layers and training the model for another 30 epochs. The initial 100 layers of the base model are still kept frozen in this stage of training.

D. Transfer Learning with ResNet50

We used residual network as a backbone for our transfer learning model. Microsoft's ResNet[1] was the winner of 2015 Imagenet challenge. For deep neural network inputs become very small for deeper layers. So during back propagation weight parameter gradient becomes zero which prevents the model from training. To mitigate this, He et al. suggested to add bypass path for the input to the next stage along with the output. This prevented the inputs from becoming very small in deep layers. This enabled the deep neural network to train. Fig. 3 shows the top layers that we have added on the ResNet base model. The original ResNet50 input image size is $224 \times 224 \times 3$, so we upsize our image size from $96 \times 96 \times 3$. In top layers we use dropout layer and batch normalization layers to prevent overfitting. We use ImageNet weight parameters for ResNet50 structure for initial training. We optimize the model on Adam optimizer with categorical cross entropy loss

function. Initially we train the model for 30 epochs, then we unfreeze the last 124 layers of ResNet50 for fine-tuning (30 epochs). Also we disable the batch normalization layers from training. Here also we use different strategies e.g. one cycle policy and learning rate reduction on plateau, for learning rate selection.

V. Experiments and Results

A. Logistic Regression

We train the logistic regression model for 30 epochs using Stochastic Gradient Descent optimization with a learning rate of 0.01. As we can see from the training loss plot in Fig. 4, the loss drops to 0 after 15 epochs, stagnating the model accuracy to roughly 70%. Further, as we can see from the confusion matrix (Fig. 5; X-Axis: Predicted label values, Y-Axis: True label values), the model predicts almost 80% of the data as having no metastatic tissue, which is why it is correct 70% of the time. The model metrics are as published in Table I.

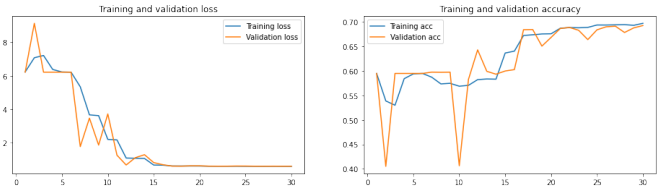


Fig. 4: Logistic Regression Loss and Accuracy Plots

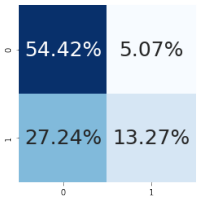


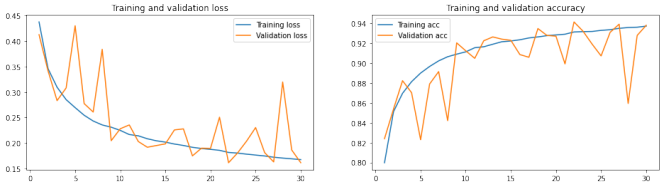
Fig. 5: Logistic Regression Confusion Matrix

TABLE I: Logistic Regression Metrics

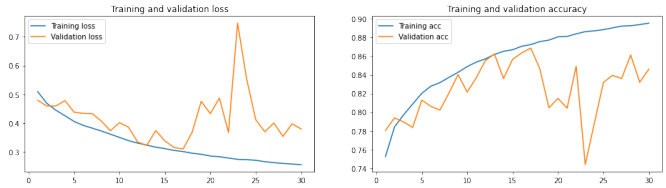
F1 Score	AUC	Recall	Precision	Acc.
0.452	0.622	0.329	0.723	0.677

B. Convolutional Neural Network

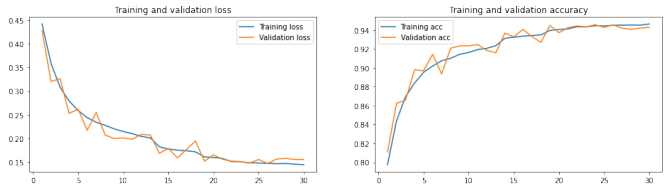
We train three CNN models with different learning rate approaches: **fixed learning rate, reducing learning rate on plateau, and varying learning rate using one cycle policy**. The loss and accuracy plots, and confusion matrices for these are as shown in Fig. 6 and Fig. 7. The reported metrics in Table II show that CNN performs much better than logistic regression, and reducing learning rate on a plateau gives optimal results. Further, we can see from the accuracy and loss plots for One Cycle Policy (Fig. 6b), the model starts overfitting after 20 epochs.



(a) Fixed LR

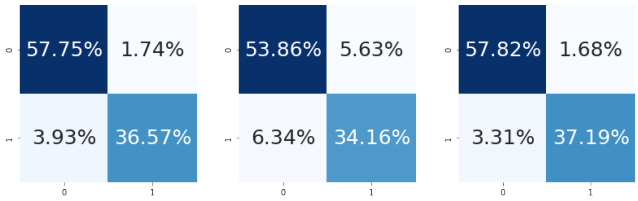


(b) One Cycle Policy



(c) Reduce LR on Plateau

Fig. 6: CNN Loss and Accuracy Plots



(a) Fixed LR (b) One Cycle Policy (c) Reduce LR on Plateau

Fig. 7: CNN Confusion Matrices

C. Transfer Learning with MobileNetV2

Three experiments were performed, using constant learning rate, changing learning rate using a one cycle policy and learning rate reduction on plateau. Initial learning rate was kept 0.0001 in all the cases. One disclaimer to note here is that due to a training interruption and restarting there is a kink in the one cycle policy plot.

D. Transfer Learning with ResNet50

For training the model we first need to find a suitable learning rate. For that we have used learning rate finder from [9]. Based on this we see $10^{-4} - 10^{-2}$ is a good range of choice. So for reduced learning rate we use 10^{-2} as the initial learning rate and reduce it by factor of 2 when there is no improvement in validation accuracy for two consecutive epochs. During fine tuning we find preferable learning rate is $10^{-6} - 10^{-4}$. So for reduced learning rate we use 5×10^{-5} as the starting learning rate. Fig. 10 represents the learning curve

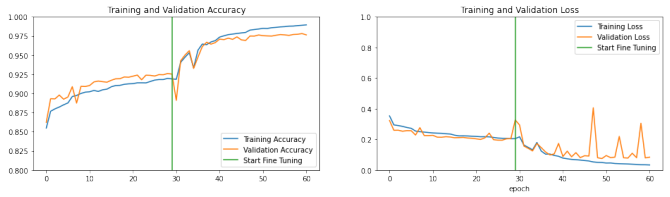
TABLE II: CNN Model Metrics

	F1 Score	AUC	Recall	Prec.	Acc.
Fixed	0.928	0.937	0.903	0.955	0.943
1 Cycle	0.852	0.875	0.845	0.858	0.881
Plateau	0.936	0.944	0.917	0.956	0.949

TABLE III: MobileNetV2 Model Metrics

	F1 Score	AUC	Recall	Prec.	Acc.
Fixed	0.916	0.925	0.876	0.959	0.935
1 Cycle	0.931	0.941	0.916	0.947	0.945
Plateau	0.922	0.933	0.913	0.932	0.938

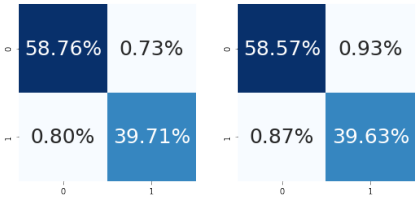
for two different strategies. It is clear during fine tuning there is a good amount of improvement. We can see some fluctuation in crossentropy loss function at the end of the training but it does not affect the accuracy. Confusion matrices in Fig. 11 and Table IV shows the performance of the model on test dataset and the final test accuracy is 98.5%.



(a) Reduce LR on Plateau

(b) One Cycle Policy

Fig. 10: ResNet50 Loss and Accuracy Plots



(a) One Cycle Policy (b) Reduce LR on Plateau

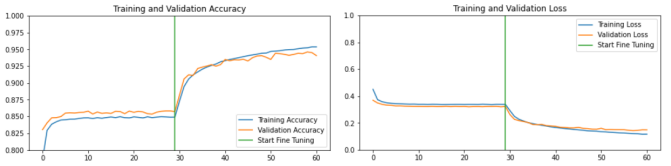
Fig. 11: ResNet50 Confusion Matrices

TABLE IV: Resnet50 Model Metrics

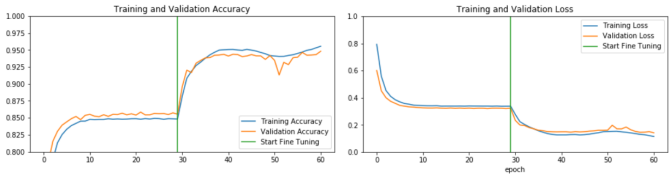
	F1 Score	AUC	Recall	Precision	Acc.
1 Cycle	0.981	0.984	0.98	0.982	0.985
Reduce	0.978	0.981	0.978	0.977	0.982

VI. Conclusion

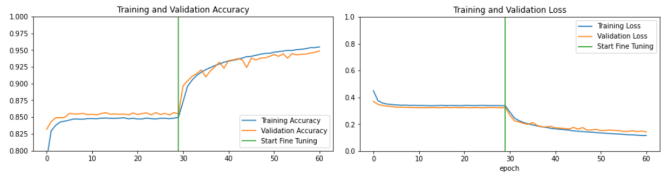
Table V summarizes the best results of each model. As we can see, logistic regression has the worst performance,



(a) Fixed LR

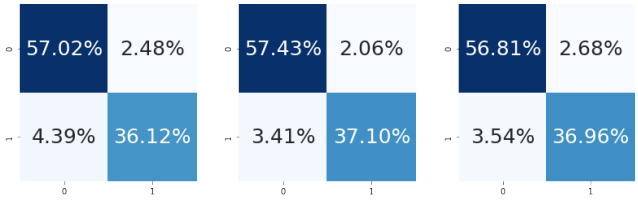


(b) One Cycle Policy



(c) Reduce LR on Plateau

Fig. 8: MobileNetV2 Loss and Accuracy Plots



(a) Fixed LR (b) One Cycle Policy (c) Reduce LR on Plateau

Fig. 9: MobileNetV2 Confusion Matrices

classifying almost all the samples as 'no cancer tissue found'. CNN, MobileNetV2, and ResNet50 have considerably improved performance. CNN and MobileNetV2 have comparable metrics. However, MobileNetV2 being a light-weight model, has reasonably smaller run-time during training. ResNet50 exhibits the highest accuracy, at a cost of substantially large runtime. The high complexity of the ResNet50 model justifies its significantly better performance.

Over the past few sections, we have discussed four different models with different hyper-parameter tuning and learning rate approaches. Regularization techniques such as dropout and data augmentation have pivotal contribution to improving model performances. Further, test time augmentation also enhances the model metrics and makes it more robust.

TABLE V: Model Comparisons

	F1 Score	AUC	Recall	Prec.	Acc.
Regression	0.452	0.622	0.329	0.723	0.677
CNN	0.936	0.944	0.917	0.956	0.949
MobileNet	0.931	0.941	0.916	0.947	0.945
ResNet50	0.981	0.984	0.98	0.982	0.985

VII. Contributions

- **Sayak Kundu:**
  Sayak worked on the ResNet50 model. Conducted multiple trials to find optimum hyper-parameters for the model, and experimented with top layer training and model fine-tuning to converge on the best model.
- **Vaibhav Tiwari:**
  In charge of MobileNetV2 model. Carried out multiple experiments pertaining to the model, including using different types of datasets (original, augmented, original + augmented).
- **Krish Vadodaria:**
  Worked on data augmentation, and logistic regression and CNN models. Responsible for all experiments related to CNN, including hyper-parameter tuning, learning rate variations, etc.

The team held regular meetings to discuss project progress, difficulties faced, and worked together to complete all deliverables on time.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Kaggle. Cnn - how to use 160,000 images without crashing. https://bit.ly/3izQDqA.

[3] Kaggle. Histopathologic cancer detection. https://bit.ly/3ixTaSr.

[4] Sonal Kothari, John H Phan, Andrew N Young, and May D Wang. Histological image classification using biologically interpretable shape-based features. *BMC medical imaging*, 13(1):1–17, 2013.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[6] Antonio De Perio. Histopathological Cancer Detection with Deep Neural Networks. https://bit.ly/2SogiYE.

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[8] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M. Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, 2016.

[9] Pavel Surmenok. Estimating an optimal learning rate for a deep neural network. https://bit.ly/3gl8miW.

[10] TensorFlow. Transfer learning and fine-tuning. https://bit.ly/3xejq8w.

[11] Sik-Ho Tsang. Review: Mobilenetv2 - light weight model (image classification). https://bit.ly/3zix105.

[12] Sulaiman Vesal, Nishant Ravikumar, AmirAbbas Davari, Stephan Ellmann, and Andreas Maier. Classification of breast cancer histology images using transfer learning. In *International conference image analysis and recognition*, pages 812–819. Springer, 2018.

VIII. Peer Review Response

We would like to thank Groups 4, 5 and 11 for their critical review of our work. We have absorbed their suggestions as best as possible in the limited time available. Please find our responses to the queries raised:

- **Q.** The goal is to detect metastatic cancer, but the traditional method requires lots of work and time. For all network model results such that validation and training accuracy, it looks like that all of the results didn't converge with your limited training epoch. And some results still showed the overfitting effect with such a limited training epoch. Perhaps you could increase the training epoch to real demonstrate faster, reducing overfitting and accuracy.
  A. Thank you for the suggestion. As discussed in the report, we have increased the number of epochs to 30 for CNN and logistic regression, and to 30 + 30 for the transfer learning models. The results are roughly in the same ballpark, which was our original assumption and intention behind fewer epochs, as it would conserve resource utilization.

- **Q.** Are you planning to address/reduce class imbalance?
  A. Discussed in Section 3. We had attempted a CNN model by balancing the data so there were equal number of '0' and '1' samples. However, we noticed a slight reduction in accuracy due to reduction in the training dataset (we would have to ignore almost 40,000 image samples to balance the data).

- **Q.** Do you have any plans on using feature extraction techniques?
  A. No. The deep neural network models have several layers, which adjust weights and do their own 'feature extraction' of sorts in the hidden layers. The logistic regression model was simply used as a baseline, and since accuracy was very poor, no degree of modification would augment its performance to the neural network models, and was thus unwarranted.

- **Q.** What made you choose the models that you did?
  A. The problem statement calls for a binary classification machine learning model. Our literature survey for the project showed that several models have been implemented with varying metrics. Since the team was relatively new to TensorFlow and Keras, we were very intrigued by a lot of features they offered, including but not restricted to learning rate modifications, data augmentation techniques, hyper-parameter tuning options, transfer learning, etc. We decided that it would be interesting to try models with different complexities, and explore pros and cons of each. The CNN architecture implemented was chosen due to its similarities to AlexNet, one of the earliest successful image classification models. MobileNetV2 seemed like a lucrative option due to its lightweight architecture, which makes it a potential game-changer in the upcoming years, with ML gradually moving to mobile devices with smaller computational capacity. ResNet50 is considered a gold standard, and the most popular and widely used image classification model, which made it an obvious choice for experimenting.

- **Q.** Why did you only train the CNN model for 10 epochs? Based on the graph, it did not converge and higher accuracy could potentially be achieved if it was trained for more epochs.
  A. Answered above.

- **Q.** Isn't the percentage of false negatives a better deciding factor between the CNN, MobileNet, and ResNet models (as opposed to runtime)?
  A. It certainly is and we just provided runtime as another parameter for comparison between models in the presentation using which (in addition to accuracy/precision of the models), users can compare the performance gained per unit time of training time of each model. This threshold doesn't hold much value when all the models are run for larger epochs and thus we have dropped this metric from the final report.

- **Q.** What can you infer from the Logistic Regression results? Why do you think the other models outperformed it?
  A. The results of the Logistic Regression model show that the problem at hand is similar to the XOR situation, *i.e.*, the distinction between the two classes cannot be expressed as a simple linear categorization. The neural network models further extract different features from the data through their hidden layers, and are thus able to outperform Logistic Regression.

- **Q.** Do you think a ResNet model with more layers (e.g. ResNet152) could produce better results?
  A. It certainly will, given enough epochs to converge. However, the decision for model choice lies in the tradeoffs to be made with respect to model accuracy, computation cost, and memory overhead. While a ResNet152 model could produce better results, it would be a significant expense to the compute cost as well.

- **Q.** Although the authors have provided interesting strategies for the learning rate scheduling, they do not state what optimizer they use for training. Why not use the optimizer which can adjust the learning itself, like Adam?
  A. Adam optimizer was used during training of all models, but while optimizers modulate learning rate after each batch of images, we aimed to change the learning rate after each epoch.

- **Q.** I understand that it's a common way to add noise and adjust the brightness when performing data augmentation, but have the authors compared the results w/o them? I have done experiments like instance segmentation for detecting the retina, and if not using a proper noise or brightness, it might harm the model performance.
  A. The reviewer is right in pointing out the possibility of reduction in model performance with heavily augmented data because of which trials were run to decide which augmentations provided good results. For example, the amount of noise to be added, amount of rotations to be performed, etc. Furthermore, experiments with original, augmented and original + augmented datasets were also performed and published in the presentation for MobileNetV2 model, and we found best results for original + augmented data suggesting added regularization benefited model performance for unseen data. Also, the ResNet50 model was run with and without noise. The model with 2% noise added outperformed the model with noise-free data.