A professional description of your model tuning process. Keep it to ~5 plots and ~1500 words of text. Remember cross-validation, bias / variance, roc curves, & generalization error.

Optimizer

After initial attempt to select a good learning rate, it became obvious that as I add more complexity to the model (e.g., convolution layers) and power (deeper networks), it would have taken too much effort to manually find a learning rate that would allow the gradients to converge as quickly as possible while preventing too much jitter around any local minima. Therefore, I used an adaptive Adam optimizer instead which instantly made significant improvements.

Convergence

Even with an adaptive optimizer, using the existing code that stopped the training the moment validation loss increased was not sufficient for stable convergence. Hence I added patience (~25 epochs to find a lower minima than current minima). This improved the convergence across multiple runs of the same model settings. It also stopped training once the model began to suffer from overfitting (variance).

Hidden Layers

Hidden Layer 1	Hidden Layer 2	Accuracy
5	5	0.6854 +/- 0.0431
5	10	0.8485 +/- 0.0333
5	20	0.9263 +/- 0.0243
10	5	0.7609 +/- 0.0396
10	10	0.8431 +/- 0.0338
10	20	0.9267 +/- 0.0242
20	5	0.6872 +/- 0.0431
20	**10**	**0.9411 +/- 0.0219**
20	20	0.9294 +/- 0.0238

I started with a simple model given the results of part 3 results where I tried a model with two hidden layers of 5 hidden nodes each and got pretty good results (~92% accuracy). I did a parameter sweep of these hidden layers, increasing the number of hidden nodes and found that an initial layer of 20 hidden nodes followed by 10 hidden nodes were the best. However it seems that most of the models are equivalent since they are within each other’s 95% error bounds. Each accuracy result is collected after 5 trials with the data split into a training set and a validation set (rather than cross-validation).

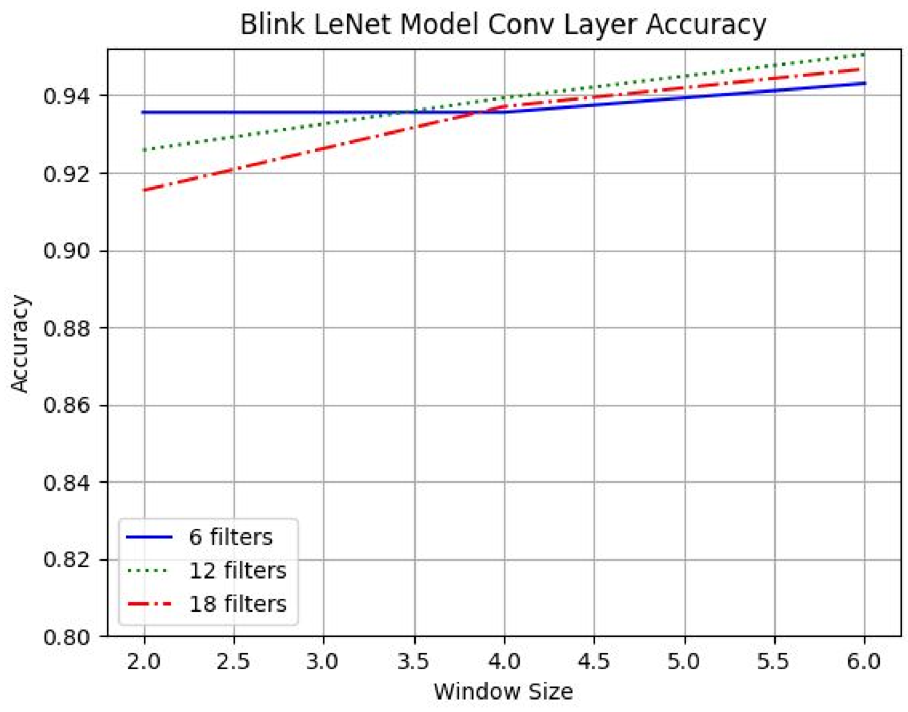
As this is a basic model, I tried next to increase the complexity of the model to reduce bias.

Conv1 Layer

Num Filters	Window Size	Accuracy
6	2	0.9356 +/- 0.0228
6	4	0.9356 +/- 0.0228
6	6	0.9431 +/- 0.0215
12	2	0.9258 +/- 0.0243
12	4	0.9393 +/- 0.0222
12	**6**	**0.9506 +/- 0.0201**
18	2	0.9154 +/- 0.0259
18	4	0.9371 +/- 0.0226
18	6	0.9468 +/- 0.0208

Next I did a parameter sweep of the first convolution layer. It seems like results improved with a larger window size and medium number of filters. Each accuracy result is collected after 3 trials with data split into a training set and a validation set (rather than cross validation). It seems again, most of the models are within each other’s 95% error bounds.

Again here, the validation loss dropped smoothly with epoch, leading us to believe that improvements can still be made by adding power to the model.

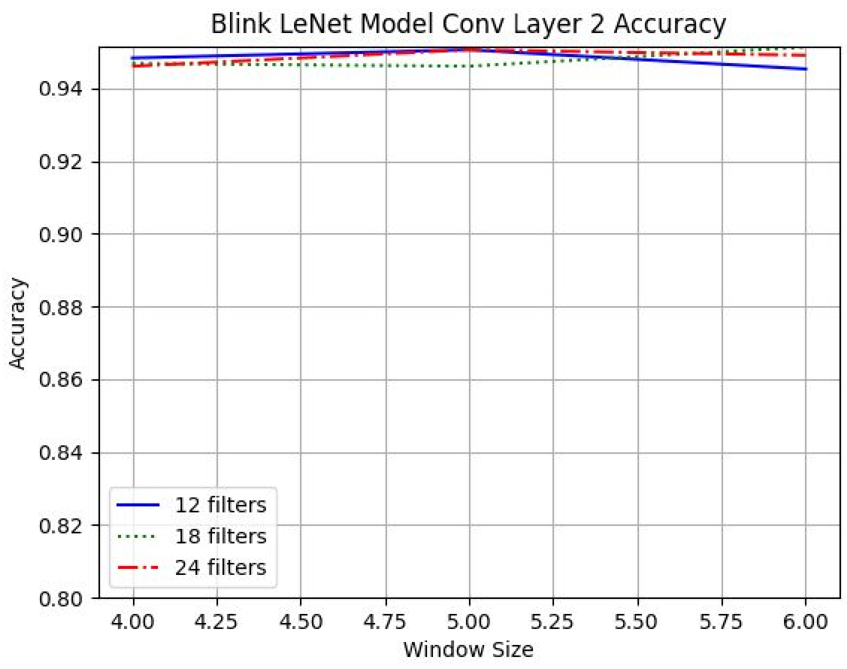


Conv2 Layer

Num Filters	Window Size	Accuracy
12	4	0.9483 +/- 0.0206
12	4	0.9506 +/- 0.0201
12	4	0.9453 +/- 0.0211
18	5	0.9468 +/- 0.0208
18	5	0.9461 +/- 0.0210
18	5	0.9513 +/- 0.0200
24	6	0.9461 +/- 0.0210
24	6	0.9506 +/- 0.0201
24	6	0.9491 +/- 0.0204

From prior results of the first convolution layer, it seems that a window size between 4-6 and a number of filters >12 could be a good starting point in the parameter sweep of the second convolution layer. Here again we conduct 3 trials at each setting with different initialization with training set and validation set (no cross-validation).

At this point, we felt the model should be sufficiently complex architecturally to capture the concept. What remains can perhaps be resolved with greater training data or model ensembles as the return from adding another layer was very negligible.



Test Dataset Results

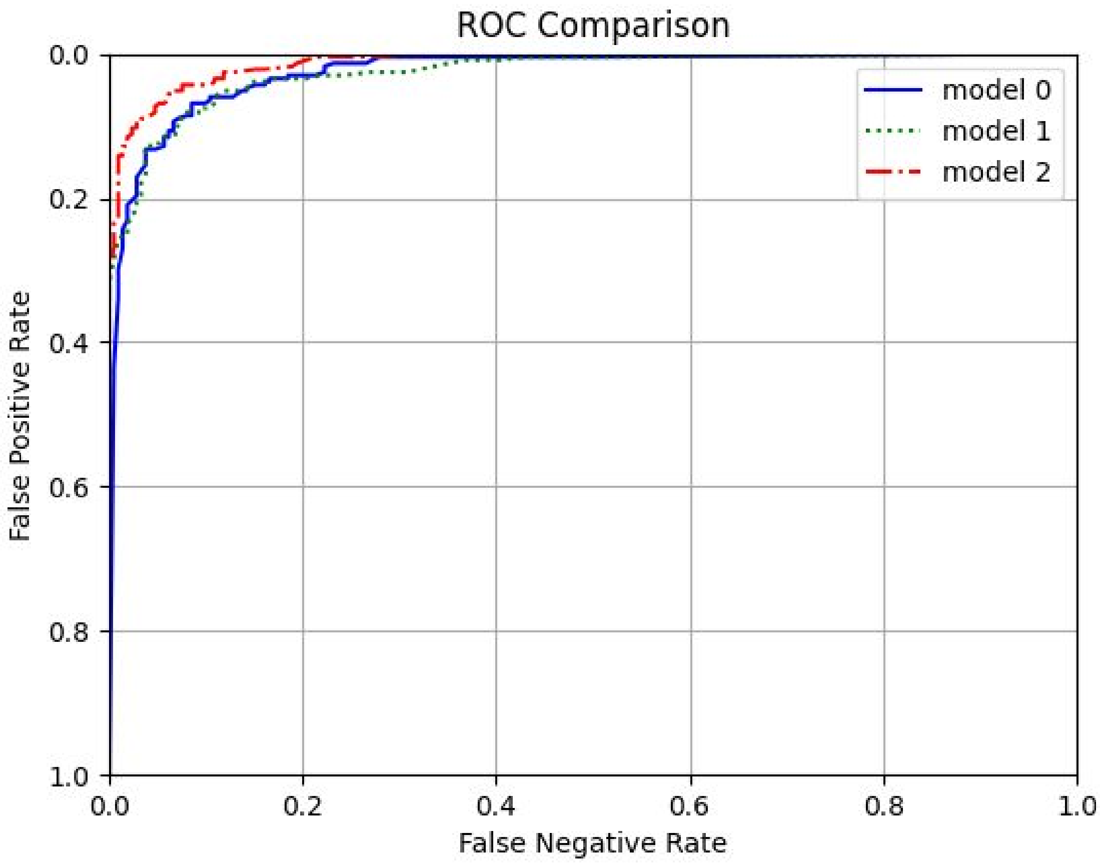
We create the final model with our best parameters:

Conv1layer: (12, 6)
Conv2layer: (18, 5)
Hiddenlayer: (20, 10)

We trained the model 3 times and tested it on the validation set data and test set data (one-shot). We choose the test data result of the model with the highest **validation set accuracy** as our final reported test data set accuracy. All error bounds are at 95% confidence level. Further we produce an ROC curve of the 3 models trained with different initialization.

Model #	Validation Accuracy	Test Accuracy
0	0.9438 +/- 0.02139	0.9146 +/- 0.02597
1	0.9326 +/- 0.02330	0.9169 +/- 0.02565

2	0.9685 +/- 0.01622	0.9393 +/- 0.02218



We learn two things here. First validation set performance is a good proxy for performance on an unseen test set. And that the gap in performance between test set and validation set could be solved by better initialization and more data (e.g., data transformations, which we didn't do because the data generator was very slow when we tried to implement it).

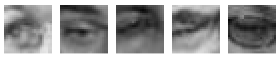
The best model is model #2, with 96.85% accuracy and noticeably better FNR and FPR across all thresholds.

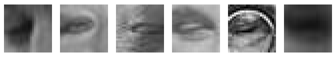
Wrong Images

In the end, we return to take a look at the data and look at some of the images that we got wrong for model #2.

Open Left Eyes:

Closed Left Eyes:

Open Right Eyes: 

Closed Right Eyes: 

We notice a few things:

1) The error rate on open eyes are not symmetric, perhaps data transformations can help here where we random pick half of the images to horizontally flip on each epoch (have to get data generator to not stall)
2) Glasses seem to interfere with the model, perhaps more data on that or finer windows on convolution filters to generate features that do not include the frame
3) Eyes at different angles seem to get misclassified
4) There are some that are just bad pictures and a human would not even know what they are looking at. That is about ¼ of these photos.