Iris & Pupil Detection L2 by Akhil

Dataset:

The given dataset consists of:
image: 224x160 RGB
mask:

- 224x160
- Greyscale
- background: 0
- pupil: 1
- iris: 2

numbers: cx, cy, rx, ry, angle

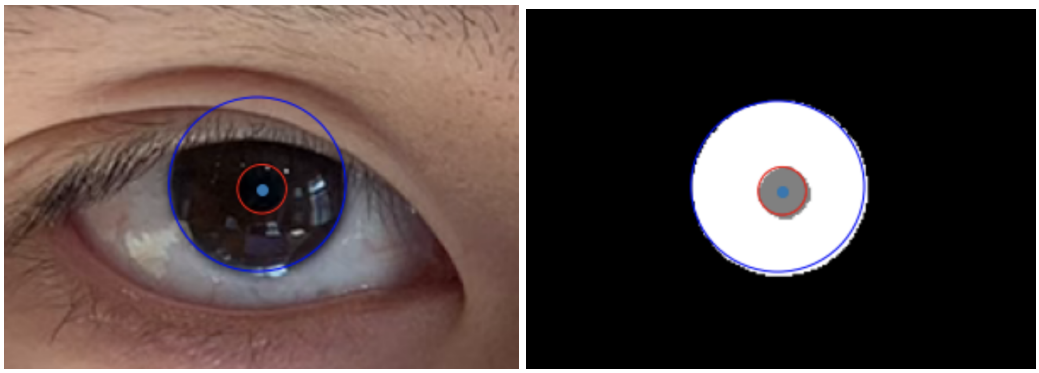

Figure 1: Sample image with labels. Code at dataset_viz.ipynb

Method:

The main ideas I worked with in this assignment are

- UNet architecture for segmentation mask generation [1]
- An auxiliary discriminator loss inspired from pix2pix [2]
- Use the predicted segmentation mask to find circles using hough circle transforms for final predictions.

UNet is an excellent architecture for segmentation. It has skip connections for every i and n-i layers, propagating high resolution features to the decoding layers

I was extremely impressed with the pix2pix paper, so I badly wanted to try it out for this task.

After producing initial results, it was immediately clear that segmentation mask is not enough by itself to produce rich width predictions. So I figured that this has to be used as a binary image and apply circular hough transforms with appropriate radius ranges to extract iris and pupil.

Losses:

1. Pixelwise cross entropy loss b/w predicted mask and the real mask. The class weights are as follows: [0.1770, 0.8455, 0.9775] for background, iris and pupil respectively.
2. Generator loss: binary cross entropy
3. Discriminator loss on real images: binary cross entropy
4. Discriminator loss on generated images: binary cross entropy

I used hydra [3] for hyper parameter configuration and optuna [4] for hyper parameter optimization.

Results:

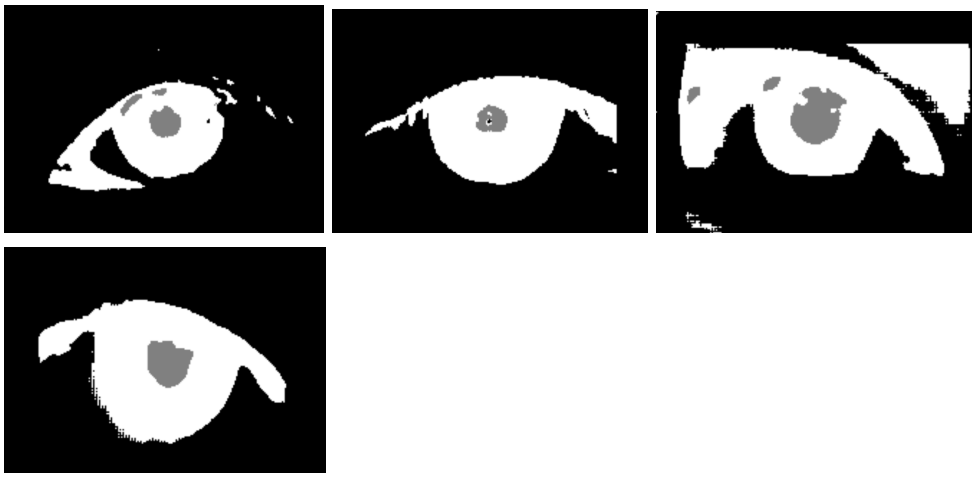

Figure 2: Sample predicted masks from training for 5 epochs

The initial results without discriminator loss looked promising as seen in Figure 5, I could spend my time in a couple of directions:

1. Improve the predictions by training longer, doing automated neural architecture search, hyper parameter optimization and possibly augmenting and combining with other datasets.
2. Use an exotic pix2pix like training scheme for fun.

And I chose the second option at the cost of better results, as we'll see, training a GAN is quite difficult.

When GANs work, they are outstanding, but more often than not, I struggle to maintain balanced training and this assignment was no exception.

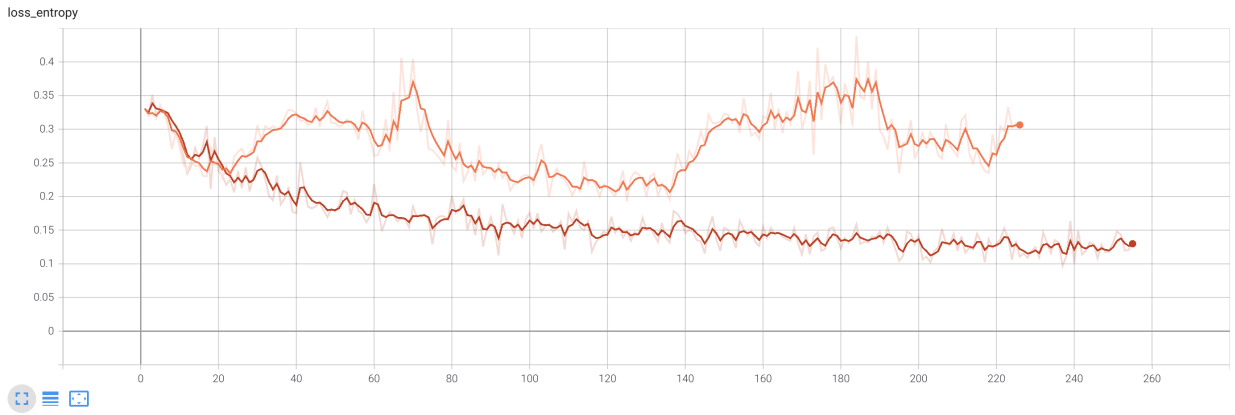

Figure 3: Training cross-entropy loss. Red is with no discriminator loss, orange is with discriminator loss.

In Figure 3, we can see that the cross-entropy loss without GAN (red) is much more monotonic than the brownian motion like curve of cross-entropy loss with GAN.

The instabilities can also be seen in Figure 4 for GAN losses.

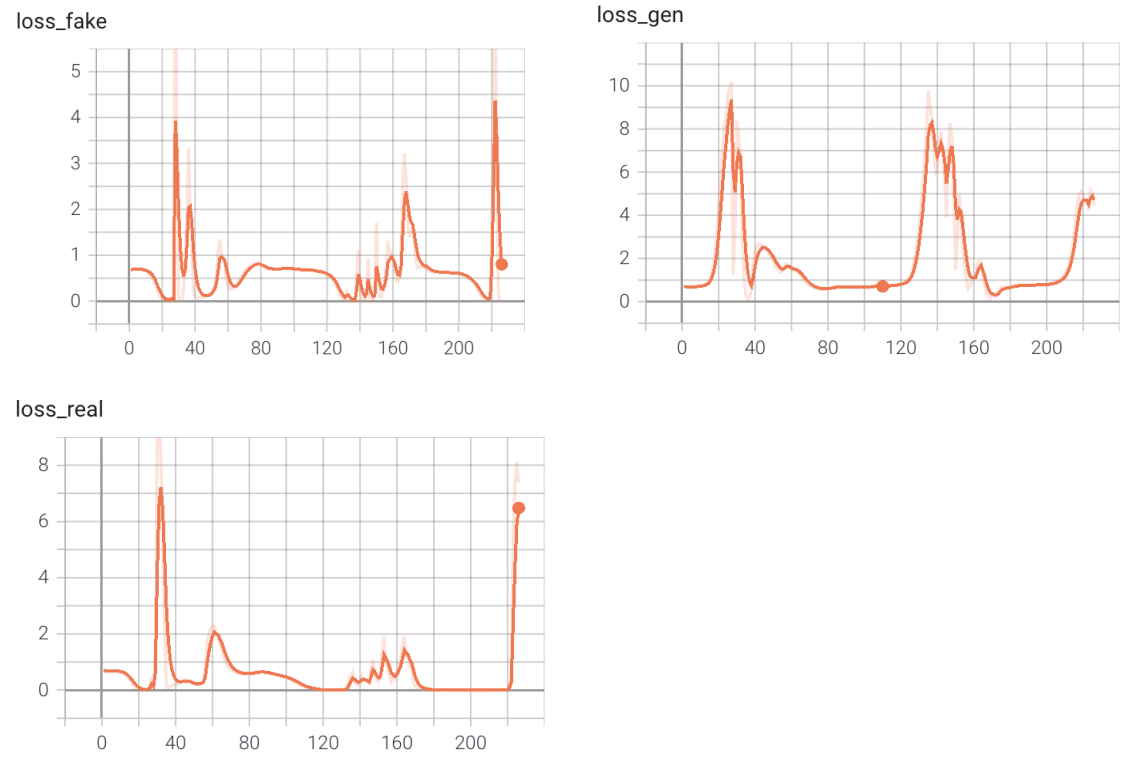

Figure 4: Training loss of discriminator and generator

Alternate training schemes:

I could design a model to predict:

- Ellipses
- Widths
- Segmentation masks

We clearly are looking for pixel or sub-pixel level accuracy. Regression based training in my experience is noisy and achieving a high precision is very difficult, so I decided to go with prediction of segmentation masks.

Difficulties I faced:

- Training a GAN is notoriously tricky
- I spent hours tuning hyperparameters, both manually and log-spaced random search only to realize that there was a bug in the generator loss. I didn't repeat the experiment after fixing the bug because I don't have another overnight of training time.

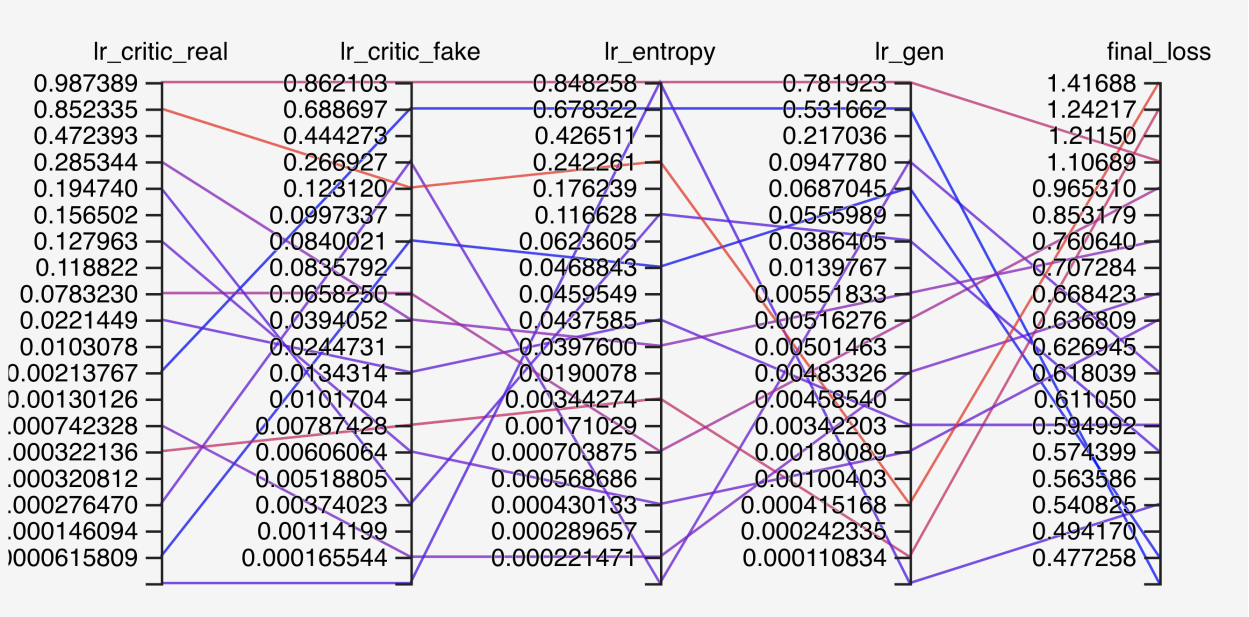

Figure 5: Parallel coordinates view of learning rates for each loss and final total loss. This is useful for visualizing hyper parameter tuning.

- My laptop is just too slow for the amount of training required, especially with automated hyperparam tuning. I should've probably used Google Colab instead

Further improvements if time permitted:

- Merging the UBIRIS dataset I had with this one for better predictions
- Use Wasserstein loss instead of cross entropy for discriminator/critic

- Add some randomness to generator either by latent vector addition or random dropouts
- More code tweaks for stable GAN training.

Conclusion:

This was a fascinating and a fun problem. I feel like I am somewhere in the ballpark of actually solving it. With additional time, I think I would bring an end-to-end working solution.

References:

[1] UNet paper https://arxiv.org/abs/1505.04597
[2] Pix2pix paper https://arxiv.org/abs/1611.07004
[3] Hydra https://hydra.cc/
[4] Optuna https://optuna.org/