CS 537: Assignment 1:

Behnam Saeedi
(Saeedib@oregonstate.edu)
Due Feb 1st 12:00pm

1 NOTE:

With the suggestion of the TA I put all of the image features into 1 file

2 PART 1

2.1 Types of key-points detected

OpenCV provides a wide range of different computer vision and image processing algorithms. For the purpose of this part of the assignment, Harris corners were used:

```
harris = cv2.cornerHarris(edges,2,3,0.04)
```

The image it self has gone through a gray scale, blur and edge detection in order to clean out the noise. Then I made sure to cherry pick top 200 highest thresholds and bundle them as keypoints for that image.

2.2 Link to OpenCV

- The resources I used to make this happen could be found here: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html
- my code could be found here (Link)

3 PART 2

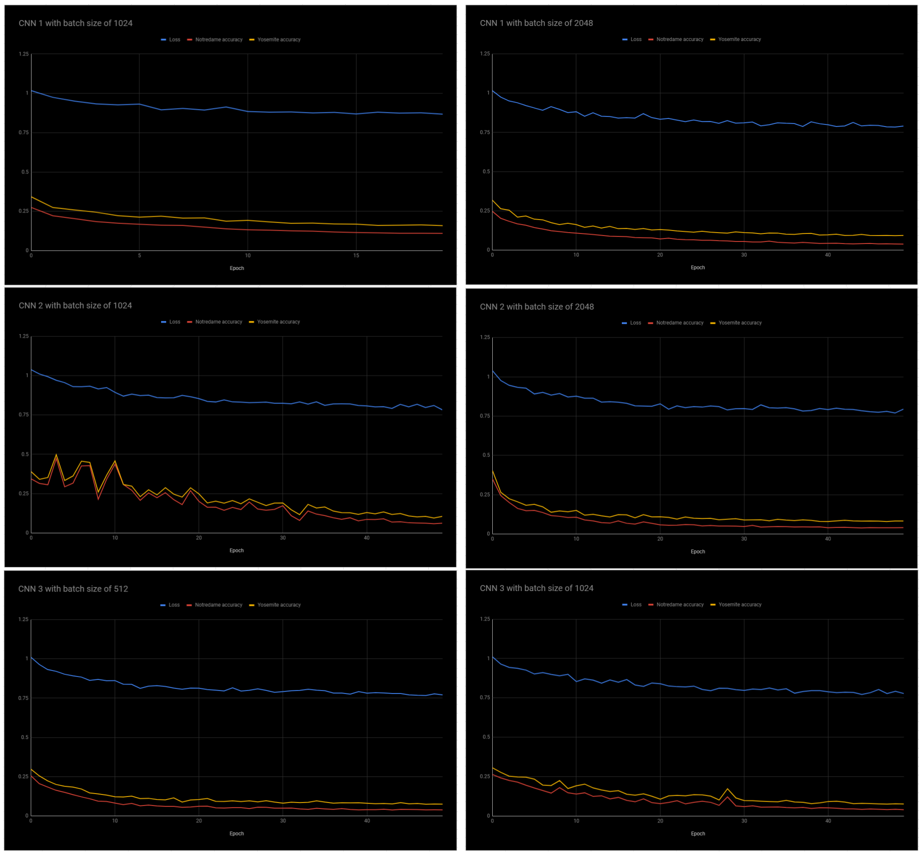
3.1 CNN - 2 and 3 Hyper parameters

- **Kernel size:** For all added layers, the kernel sizes were 3 to avoid ambiguous pyTorch errors.
- **Stride size:** Some of the layers had Stride size of 2 and others had Stride size of 1.
- **Padding:** All newly added layers had the padding size of 1 for best performance, by trials and error.
- **Bias:** All new layers had the Bias node added. This significantly improved the loss.

- **Activation function:** All activation functions used are ReLU.
- **Pooling:** The kernel size for pooling was also 3 and Stride-size of 1 and padding of 1 was imposed.
- **Batch normalization:** All added normalization layers are fully connected. (affine is set to True).

3.2 Optimal solution:

Please note that these values are subject to the hardware used and the conditions are unpredictable. My results were bound by the resources that were available to me. Used mini-batch sizes used were: 512, 1024 and 2048. Some of these values did poorly due to performance issues and others due to bad accuracy and high loss. Furthermore, after certain epochs there was not a significant improvement in performance for the training time. The models start to over fit at higher epochs where the loss continues going down but accuracy suffers. The only exception was CNN - 3. I ran out of CUDA memory before hitting those epochs.



- **CNN - 1:** Optimal batch size and epochs for CNN - 1 was batch size of 2048 at 50 epochs.
- **CNN - 2:** Optimal batch size and epochs for CNN - 2 was batch size of 2048 at 50 epochs.
- **CNN - 3:** Optimal batch size and epochs for CNN - 3 was batch size of 1024 at 50 epochs.