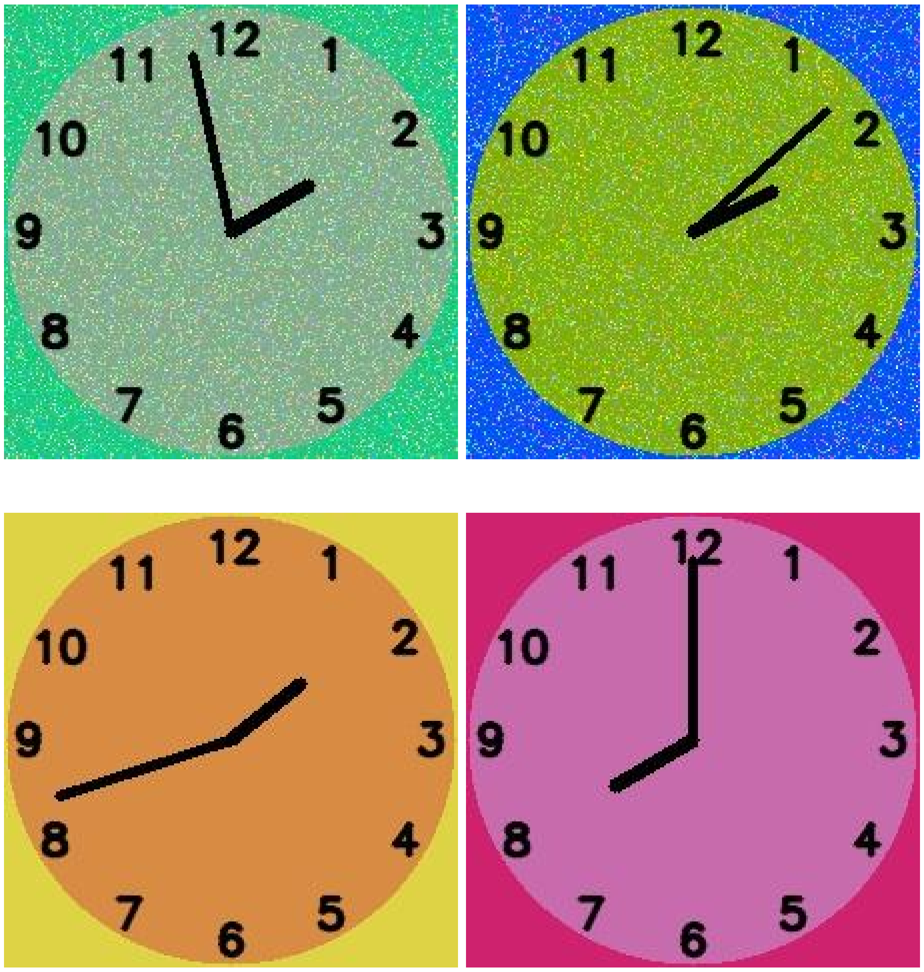
Assignment 1 Report

We have implemented a CNN model to train it for predicting hour and minute information from the data containing images of clocks. Some of the examples of our generated clocks are provided below.



In the **Prob1.py** file, we implemented two functions: one for creating the clock image in the jpg format and one for generating a certain number of clock images for training and testing our model. We generated 5000 images (3000 for training and 2000 for testing). We also saved corresponding hour and minute data in a json file.

Some of the images (approximately 50%) have no noise and the other images have noise in them. We also provided some code to draw a clock in that file. That code is at the end of the script and can be run by uncommenting it. An example hour and minute information is provided and can be changed.

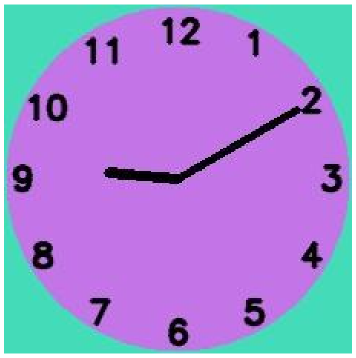
In the **Prob2.py**, we implemented our CNN model, which includes a pretrained ResNet18 model with two new feed-forward layers: one for mapping input features to hour class (12 categories) and one for mapping ro minute class (60 categories). So, we treat the problem of predicting hour and minute information as a classification problem. The training loss will be the summation of hour classification loss and minute classification loss.

In addition to training those two new feed-forward layers, we also train the backbone ResNet model in order to better adapt to our task.

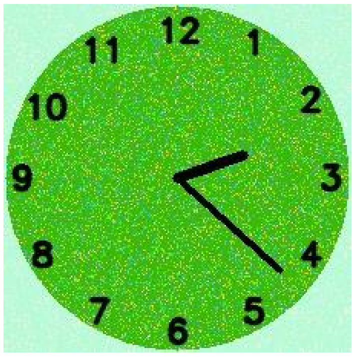
In the **Prob3.py**, we define our dataset class (ClockDataset) for loading and accessing our clock image data with its corresponding hour and minute labels. After splitting the dataset into train and test set and initializing the Pytorch data loaders (with a number of data transformations necessary for backbone ResNet model), we train our model for 10 epochs with batch_size equal to 32 and learning rate 0.001. After training our model, we evaluate it on the test set of 2000 clock images.

Only in a few epochs, we were able to reach full accuracy of 100% on the test set. The reason for such perfect performance can be that even though randomly added noise to the images is different and the color of clock / background also randomly varies across the dataset, the most important information such as digit location / color is the same across the whole dataset and hour / minute combination seen in the test set was probably seen in train set as there are only 12*60 = 720 hour and minute combinations and we have 3000 training images. In other words, the train set and test have the same distribution thus if we do well on the train set, then it is almost guaranteed that we will do well in the test set. Also, since we have a strong pretrained feature extractor such as ResNet, it helped us achieve such performance only in a few training epochs.

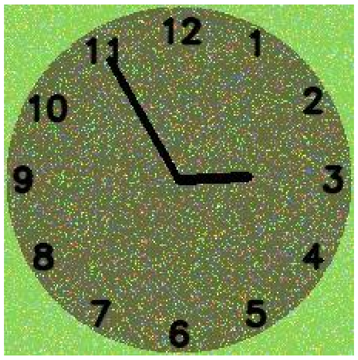
Here, we provided the prediction results for 3 images chosen from our dataset:



Predicted Hour: 9
Predicted Minute: 10



Predicted Hour: 2
Predicted Minute: 22



Predicted Hour: 2
Predicted Minute: 55

In **Prob4.py,** we implemented code to get hour / minute predictions for a single image. An example image path is provided and then we print the predicted hour and minute.

Appendix

We used the following libraries to implement and run our code:
– os
– json
– tqdm
– math
– numpy
– opencv-contrib-python
– torch
– torchvision