# Classification of Paintings

Naixin Zhu

Spring 2019 Machine Learning II Group 2

# High Renaissance

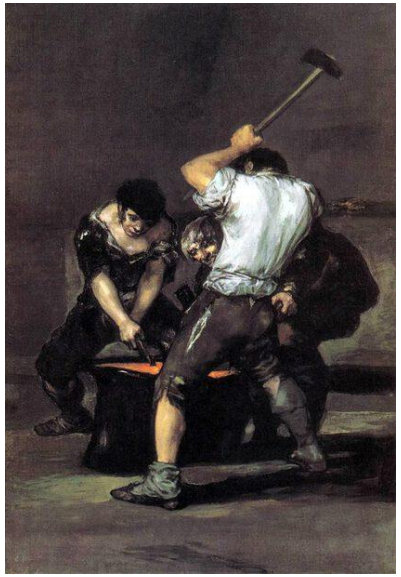
- **Leonardo da Vinci (1452 – 1519)**

# Baroque

- **Rembrandt (1606 – 1669)**

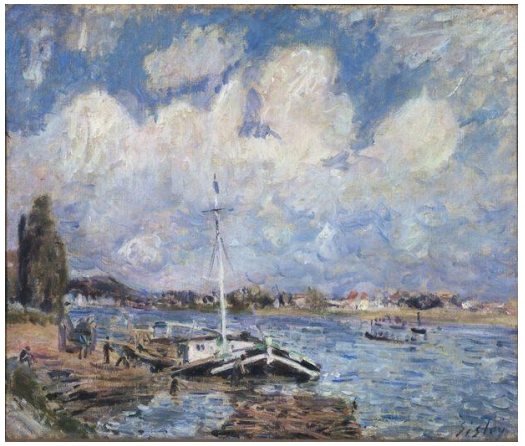
# Romanticism

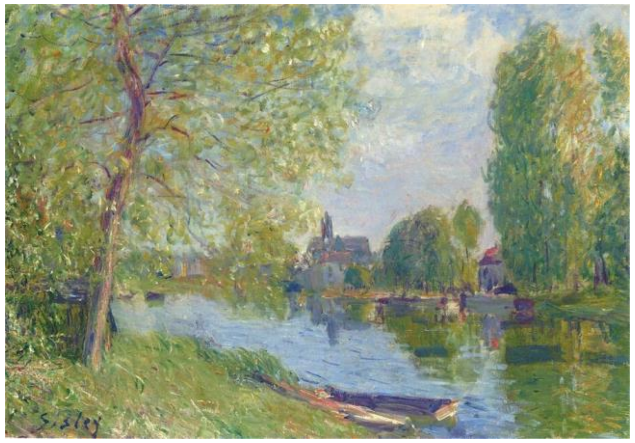
- **Francisco Goya (1746 – 1828)**

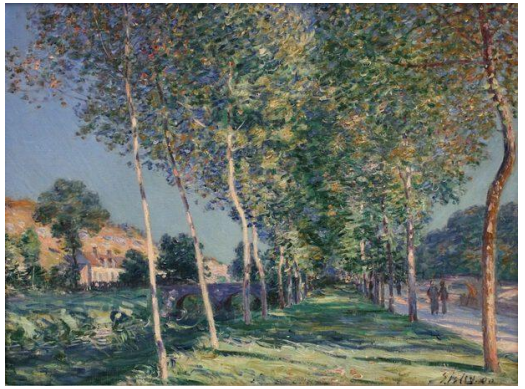


# Impressionism

- **Alfred Sisley (1839 – 1899)**
- Edgar Degas (1834 – 1917)

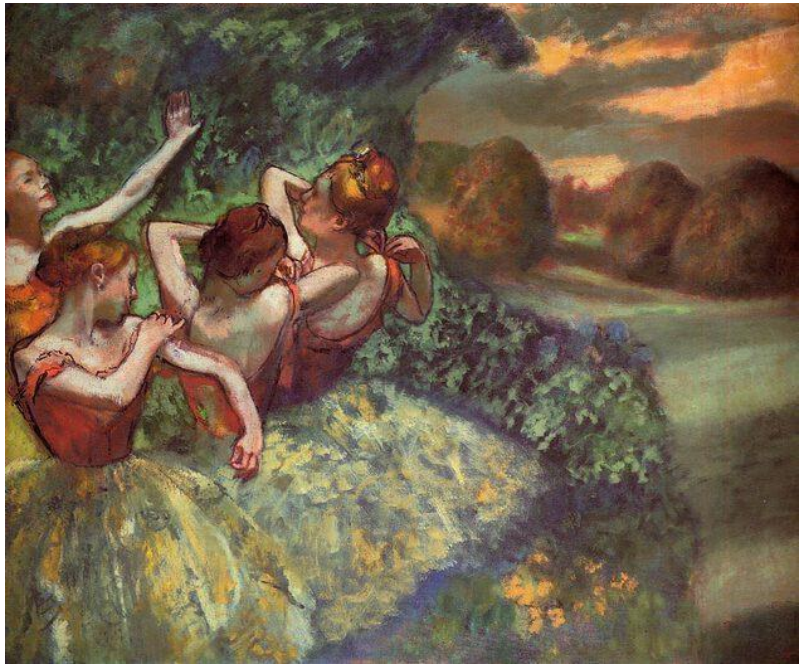


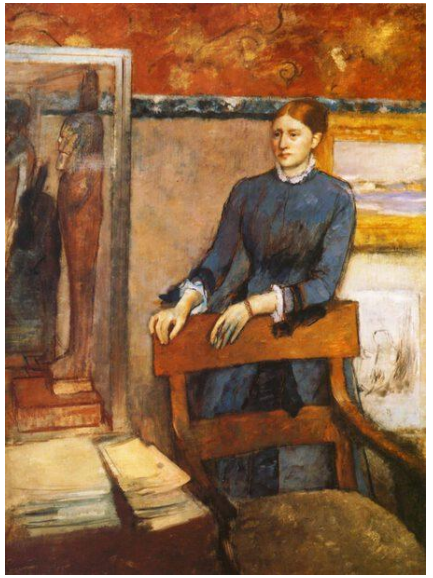


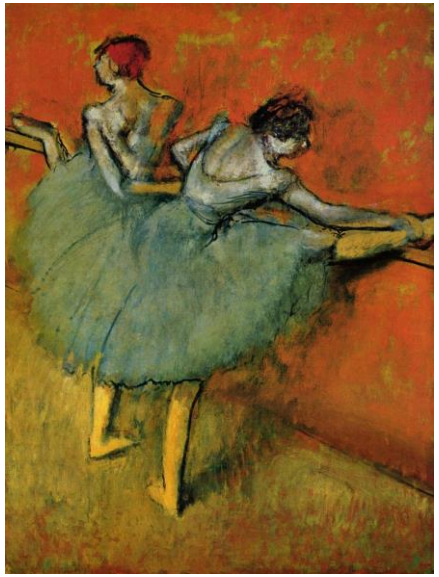


# Impressionism

- Alfred Sisley (1839 – 1899)
- **Edgar Degas (1834 – 1917)**

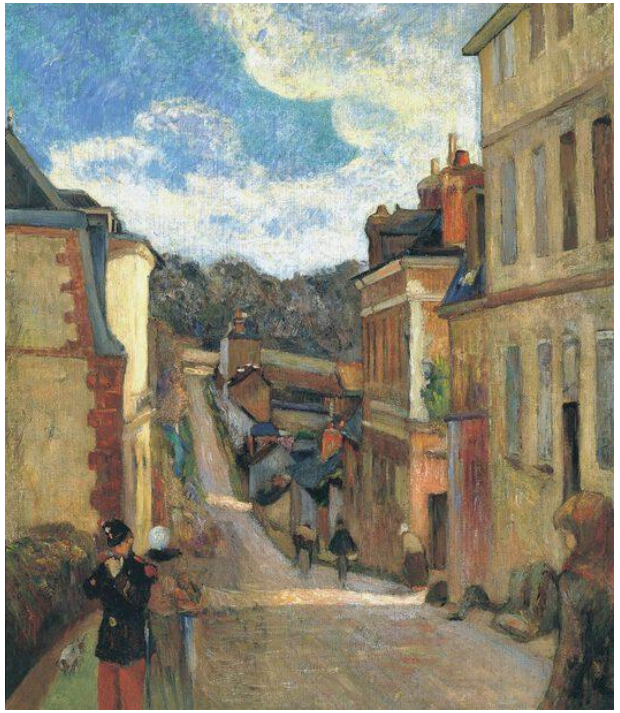


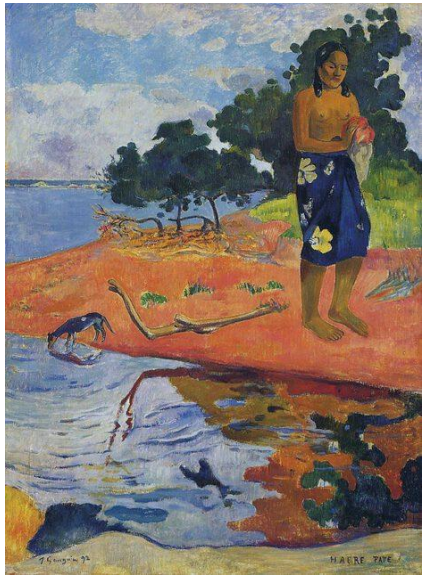


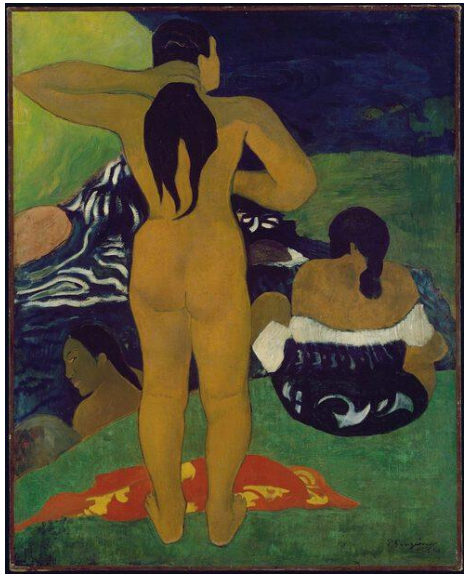


# Post-Impressionism

- **Paul Gauguin (1848 – 1903)**
- Vincent van Gogh (1853 – 1890)

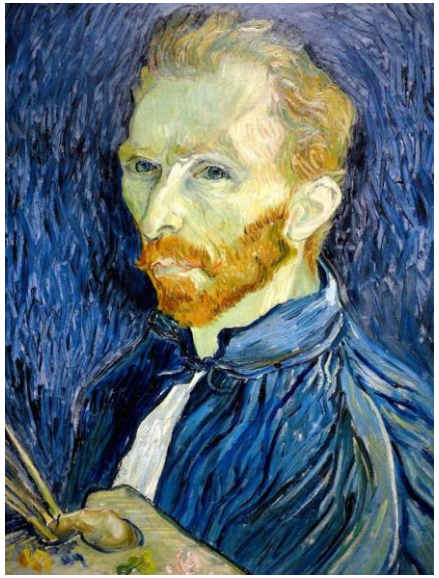


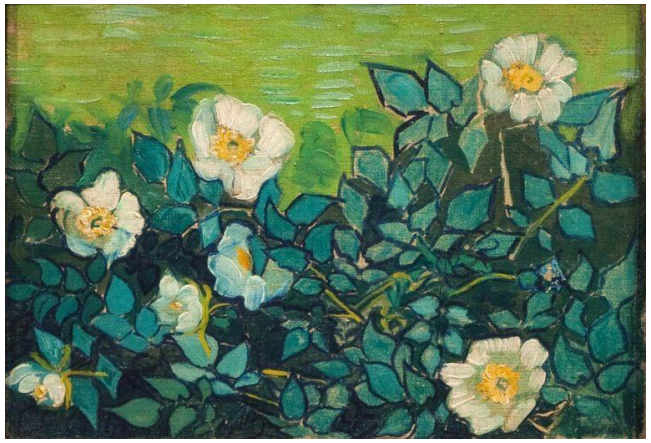


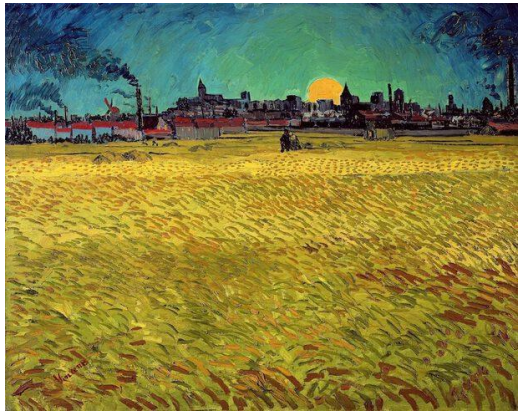


# Post-Impressionism

- Paul Gauguin (1848 – 1903)
- **Vincent van Gogh (1853 – 1890)**

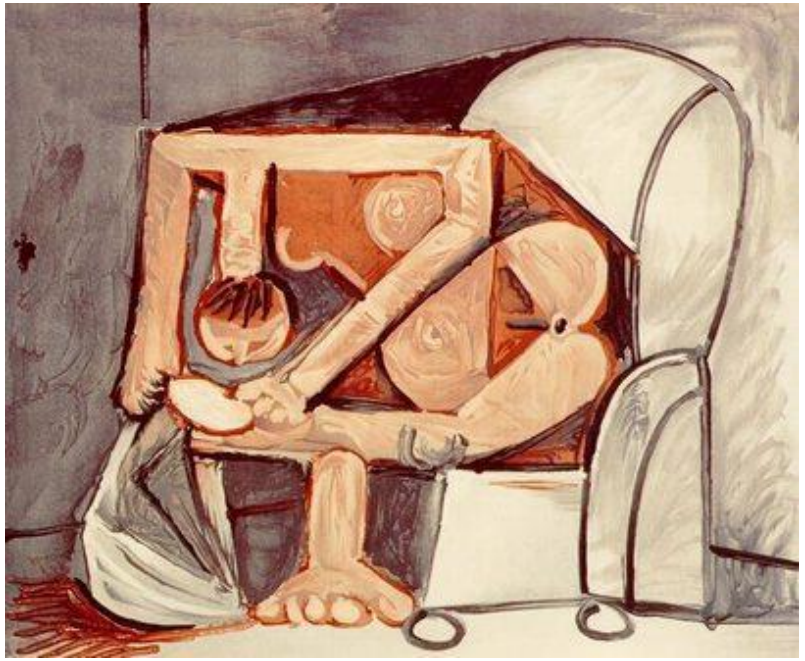


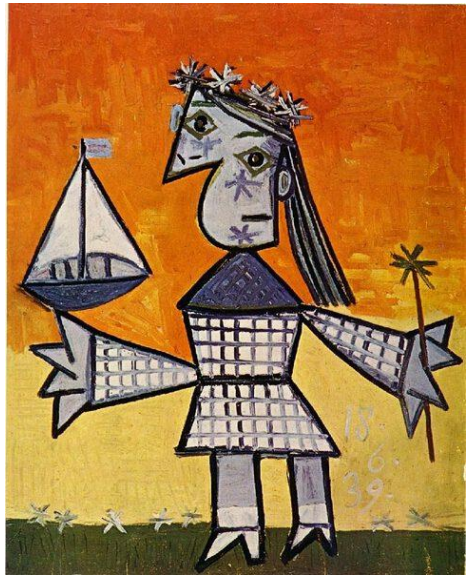


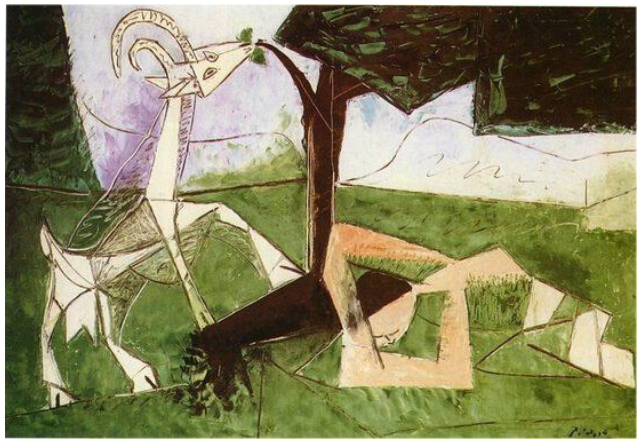


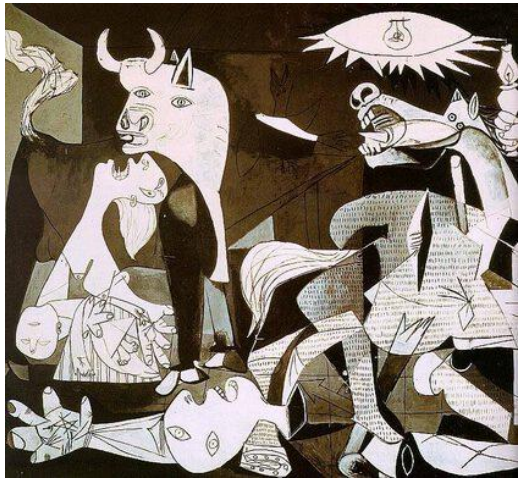
# Cubism

- **Pablo Picasso (1881 – 1973)**



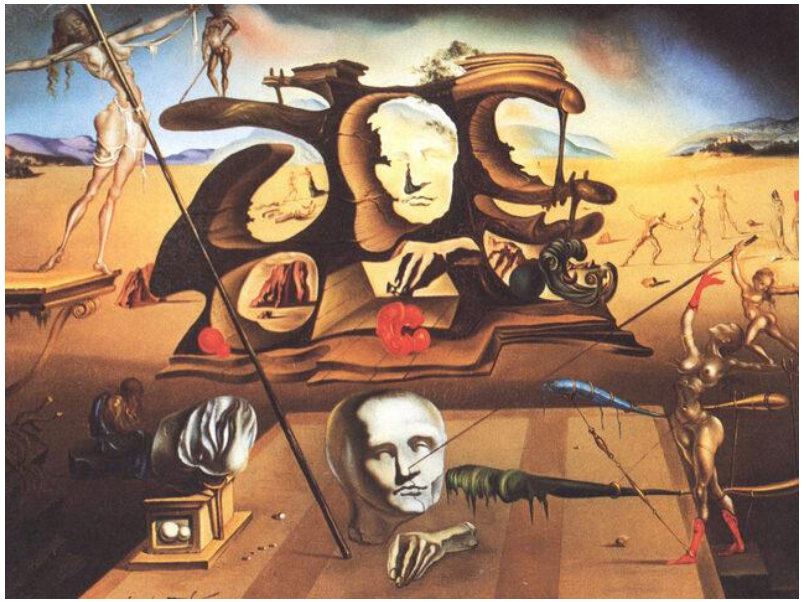


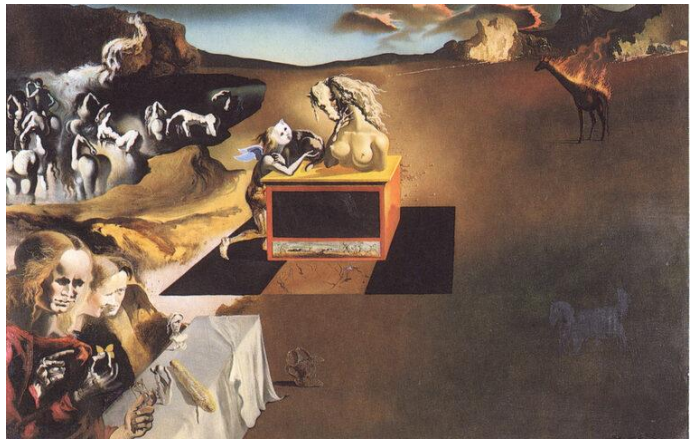




# Surrealism

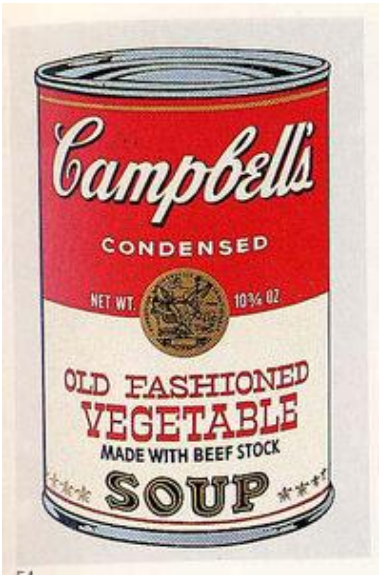
- **Salvador Dali (1904 – 1989)**

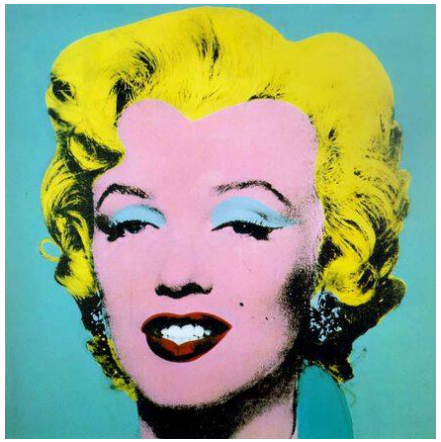


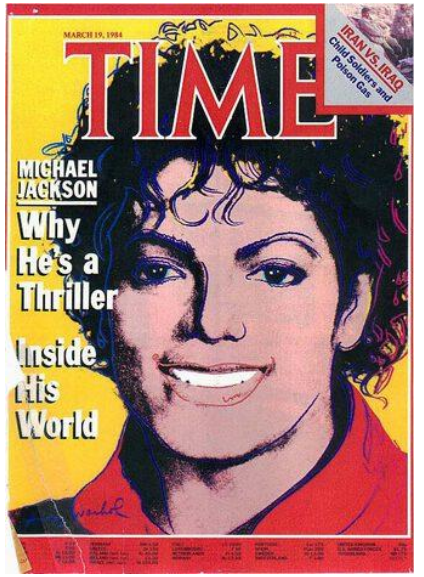


# Pop Art

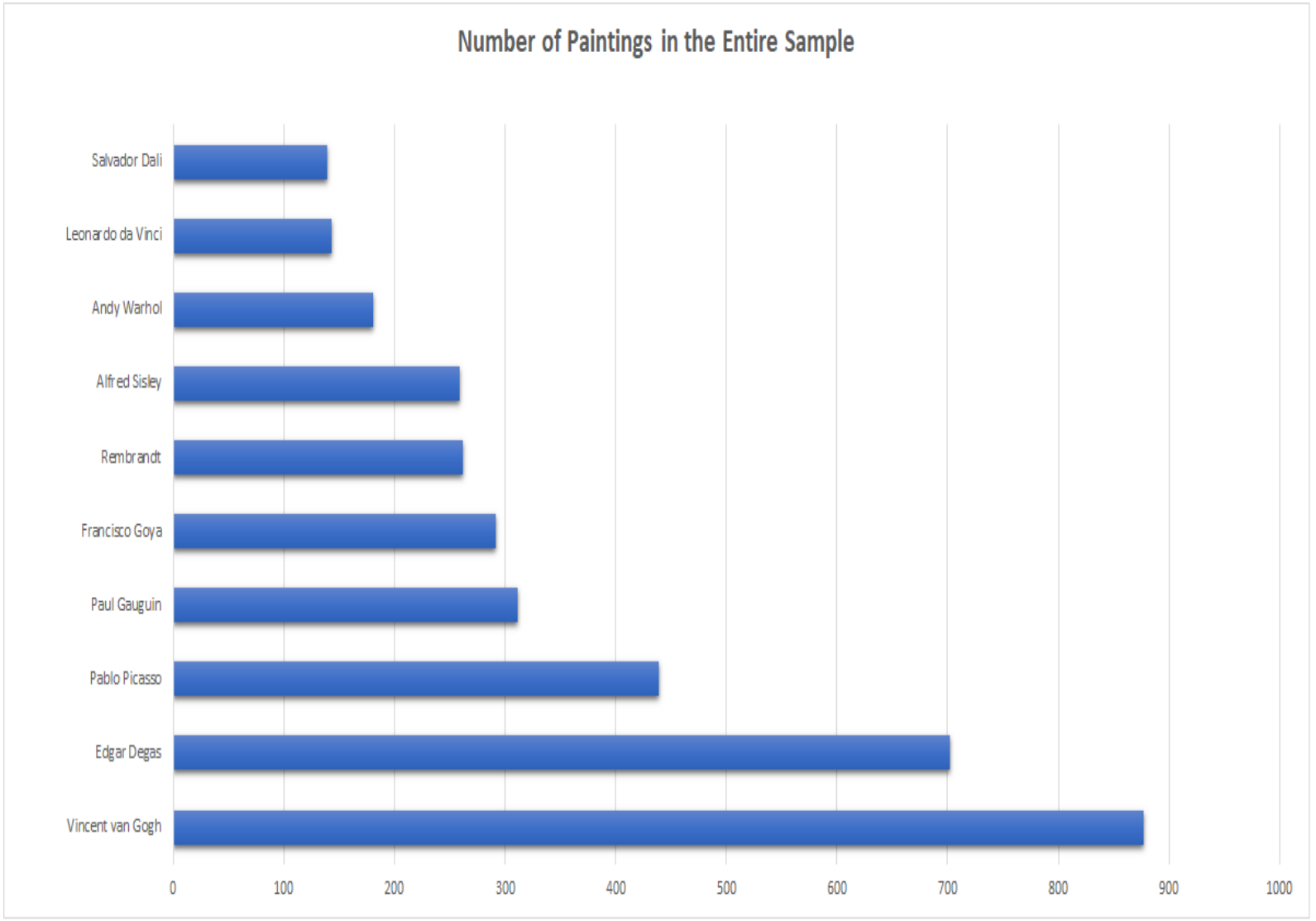
- **Andy Warhol (1928 – 1987)**





Number of Paintings per Class in Entire Dataset (3,604)
Training: 80%
Testing: 20%



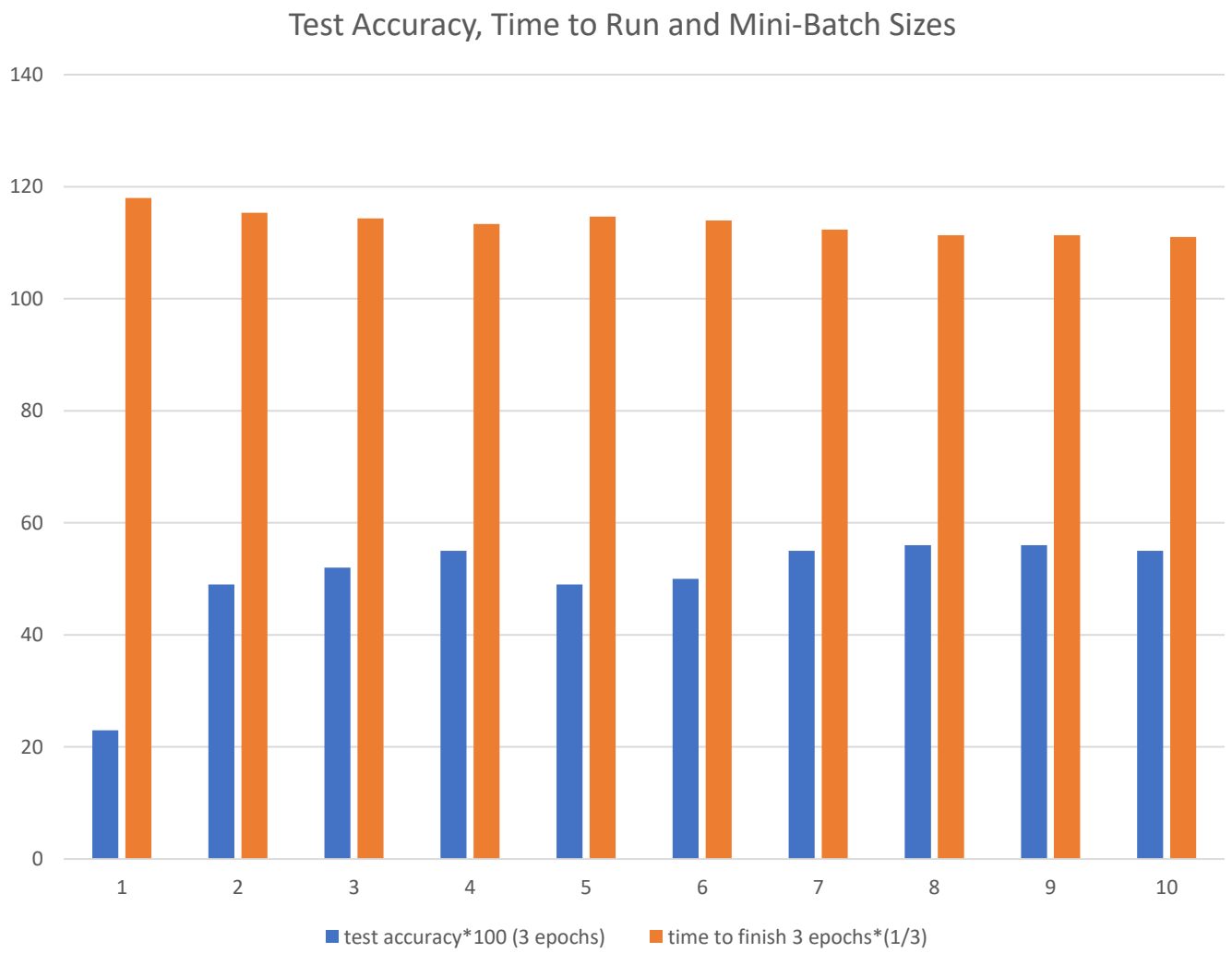
# PyTorch CNN

# Baseline Model (model 1)

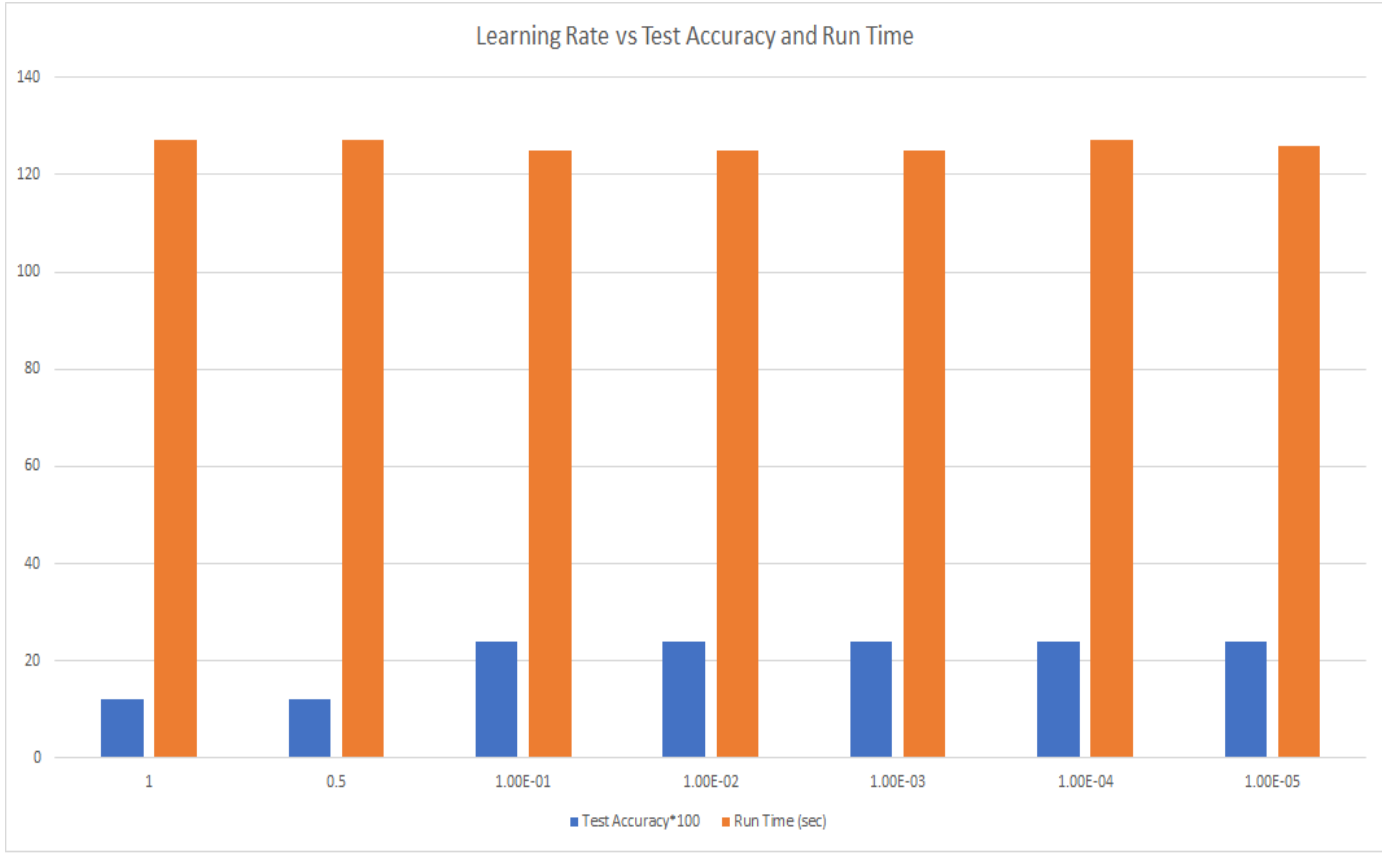
- Fixed for all models:
  - Batch Size = 4, Learning Rate = 1e-3
  - criterion = CrossEntrophyLoss()
  - Epoch = 20
  - Resize(800), Normalize(mean = [0.5, 0.5, 0.5], std = [0.5, 0.5, 0.5])
- Features:
  - 2 convolution layers
  - optimizer = torch.optim.Adam

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(3, 50, kernel_size=10, stride=5, padding=2),
            nn.BatchNorm2d(50),
            nn.ReLU(),
            nn.MaxPool2d(4))
        self.layer2 = nn.Sequential(
            nn.Conv2d(50, 100, kernel_size=10, stride=5, padding=2),
            nn.BatchNorm2d(100),
            nn.ReLU(),
            nn.MaxPool2d(4))
        self.fc = nn.Linear(100, 10)
```

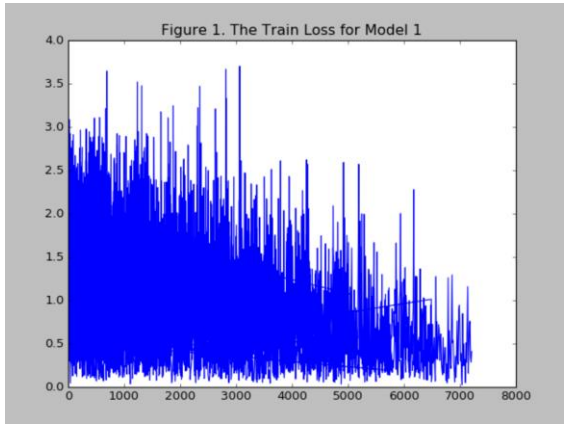
# Determine Mini-batch size



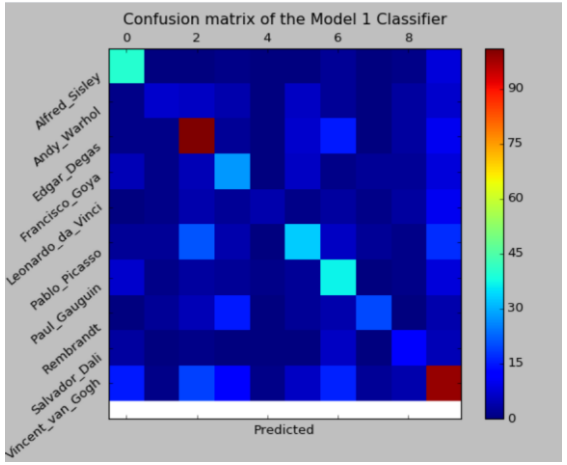
# Determine Learning Rate



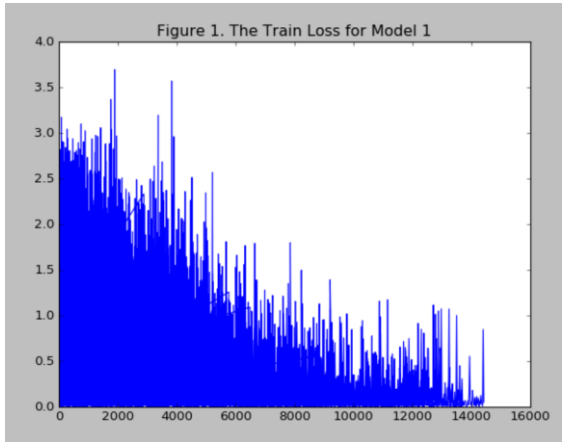
Epoch = 10



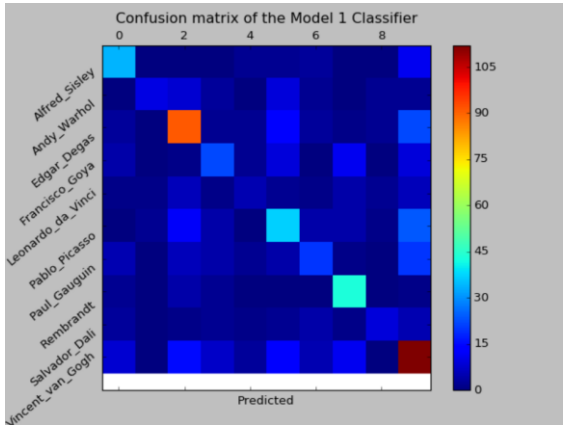

Epoch = 10




Epoch = 20



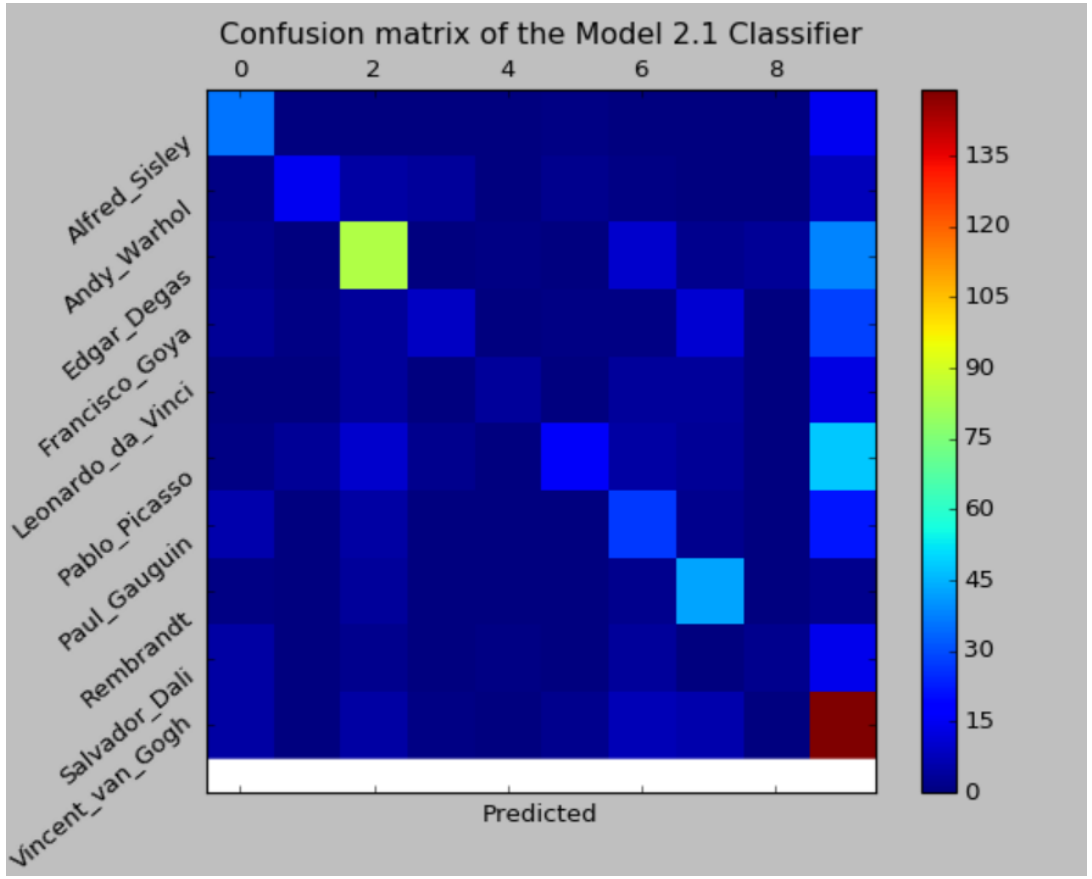

Epoch = 20

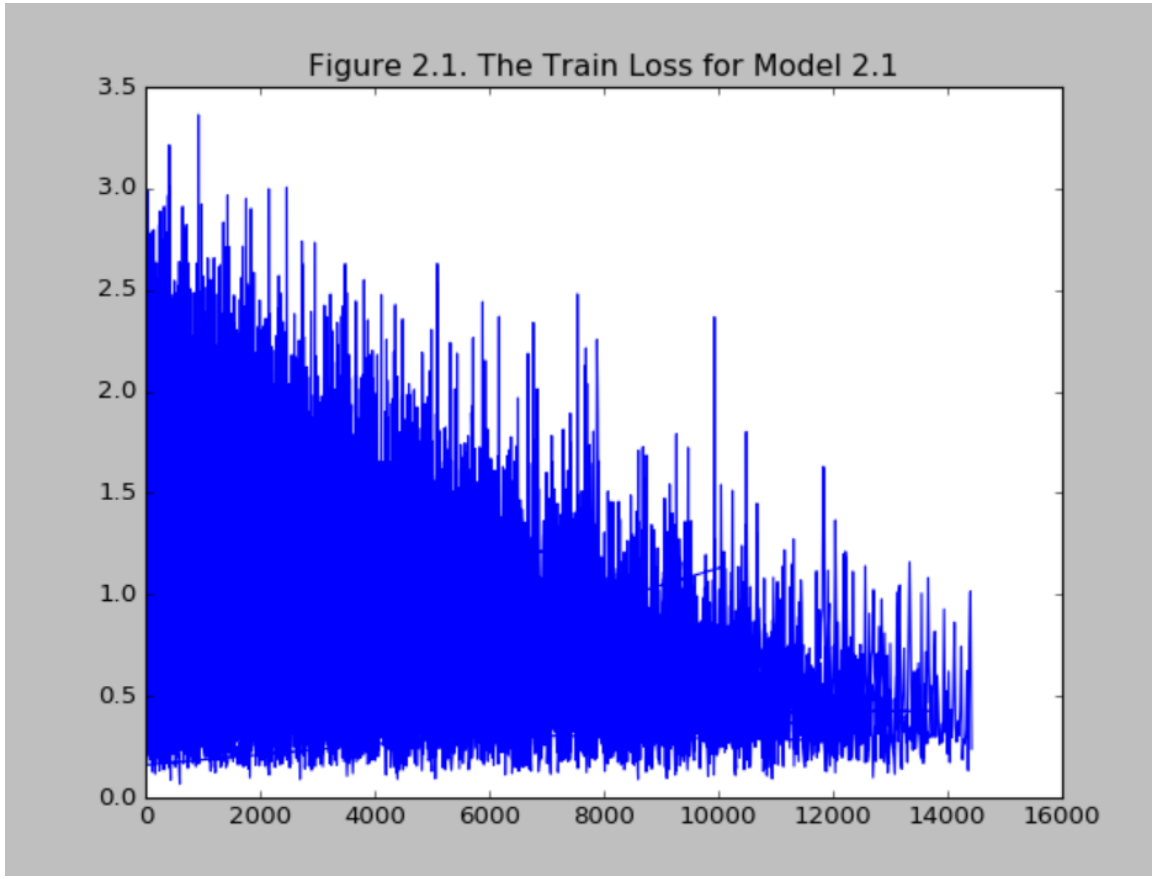



# Baseline Model (model 1)

# Model 2: Change nn.optimizer

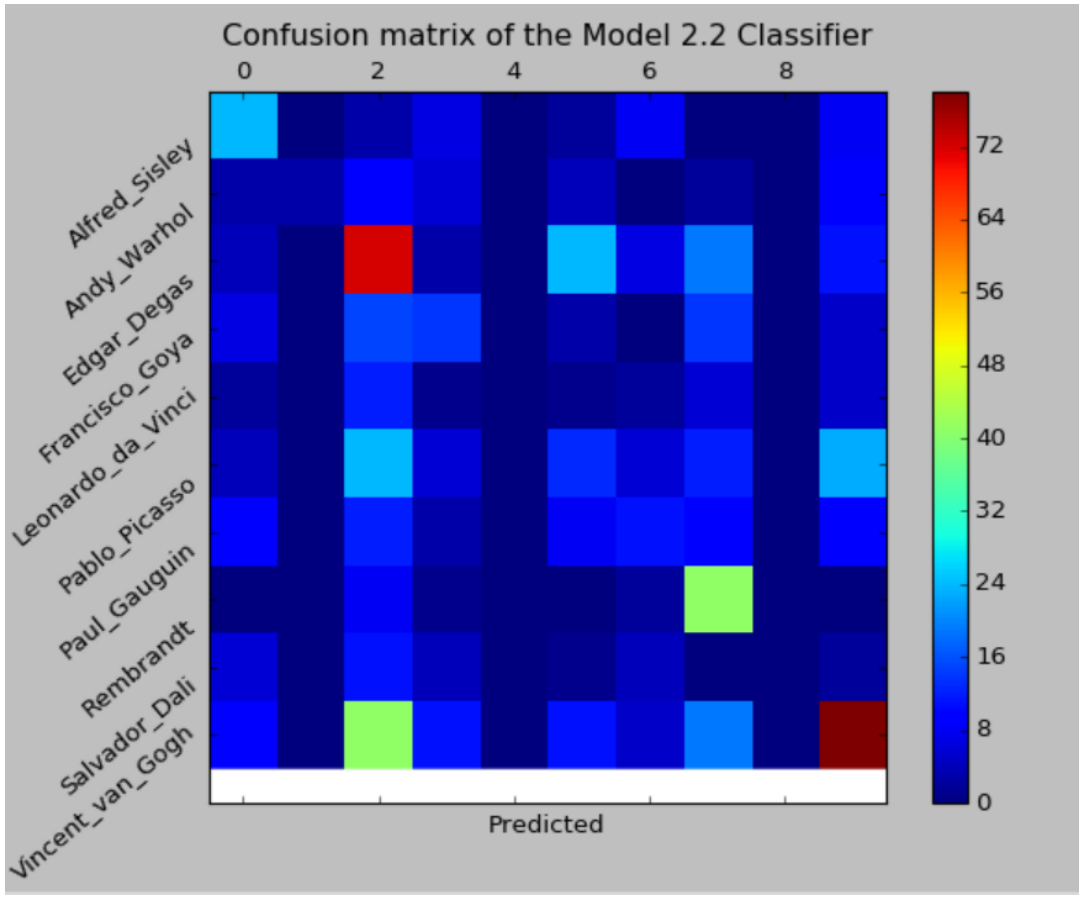
- Model 2.1
  - optimizer = torch.optim.SGD(cnn.parameters(), lr=learning_rate)
  - scheduler = StepLR(optimizer, step_size=2, gamma=0.95)

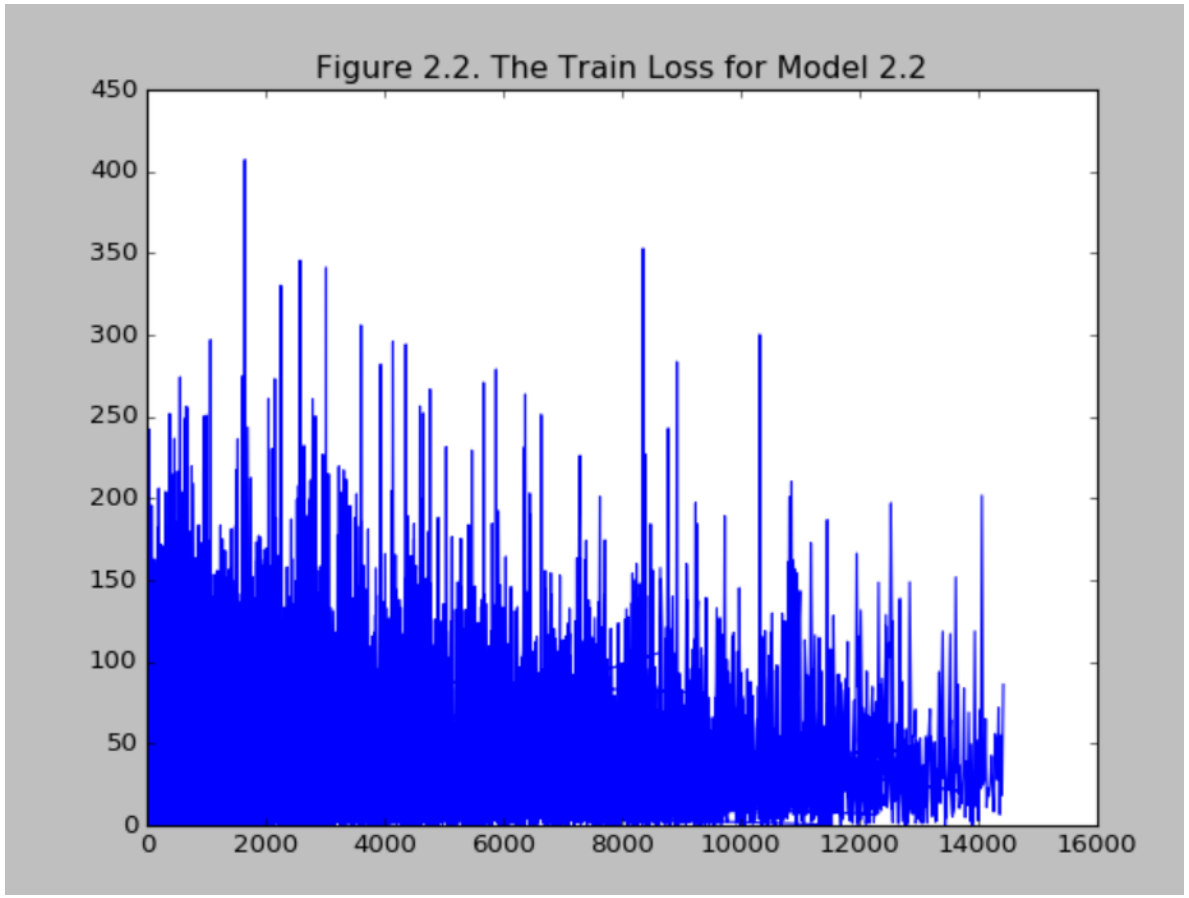


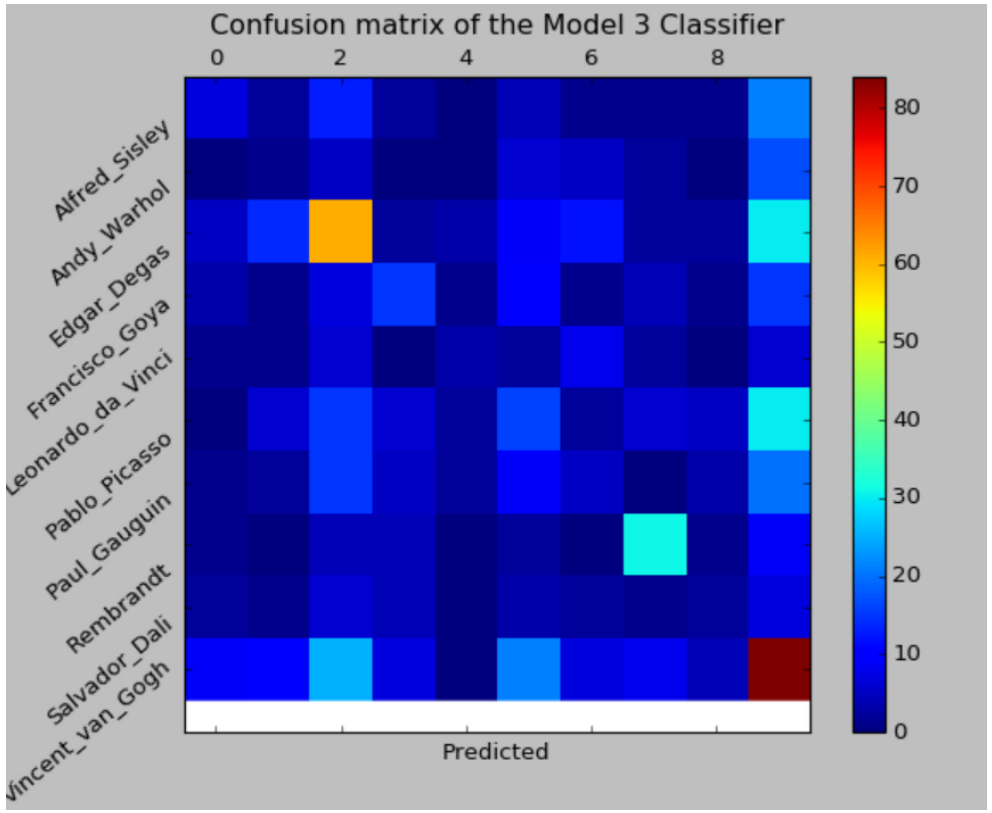



# Model 2: Change nn.optimizer

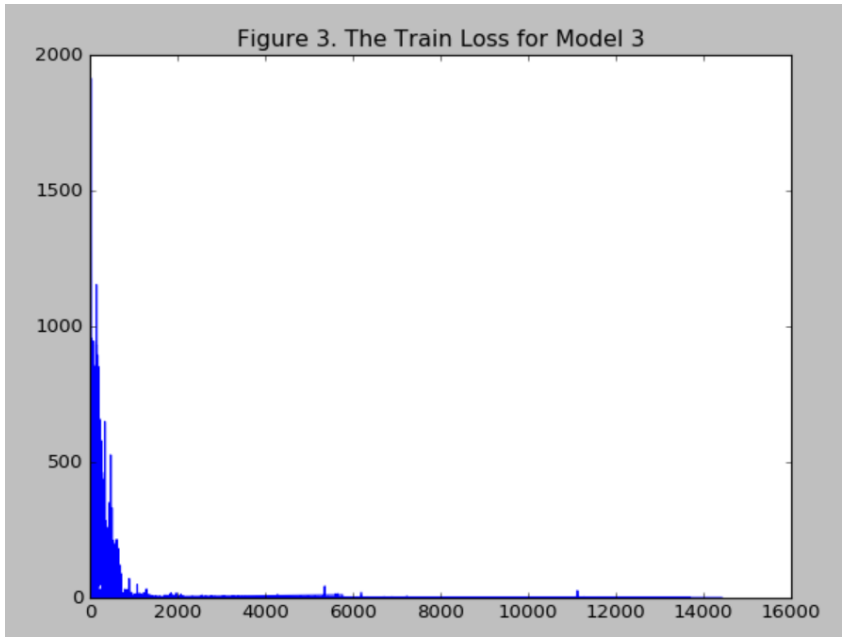
- Model 2.2
  - optimizer = torch.optim.Rprop(cnn.parameters(), lr=learning_rate)
  - scheduler = StepLR(optimizer, step_size=2, gamma=0.95)



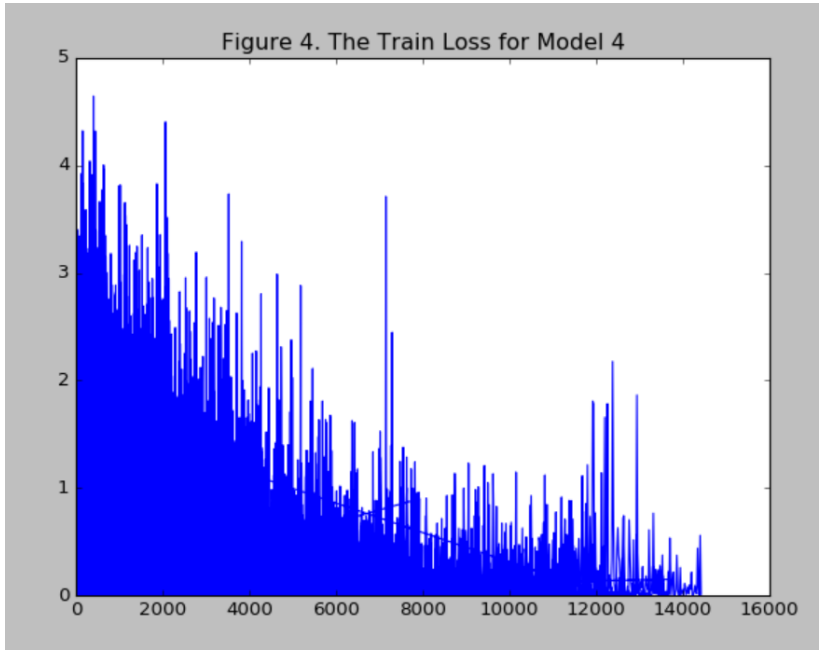



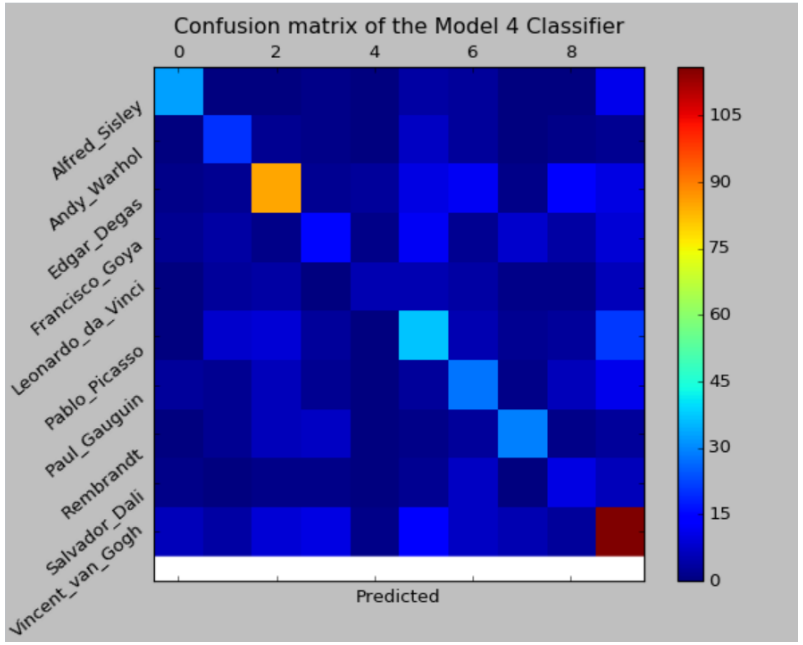



```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(3, 50, kernel_size=10, stride=1, padding=0),
            nn.BatchNorm2d(50),
            nn.ReLU(),
            nn.MaxPool2d(2))
        self.layer2 = nn.Sequential(
            nn.Conv2d(50, 100, kernel_size=10, stride=1, padding=2),
            nn.BatchNorm2d(100),
            nn.ReLU(),
            nn.MaxPool2d(2))
        self.layer3 = nn.Sequential(
            nn.Conv2d(100, 200, kernel_size=10, stride=1, padding=0),
            nn.BatchNorm2d(200),
            nn.ReLU(),
            nn.MaxPool2d(2))
        self.layer4 = nn.Sequential(
            nn.Conv2d(200, 400, kernel_size=10, stride=1, padding=0),
            nn.BatchNorm2d(400),
            nn.ReLU(),
            nn.MaxPool2d(2))
        self.layer5 = nn.Sequential(
            nn.Conv2d(400, 150, kernel_size=10, stride=1, padding=0),
            nn.BatchNorm2d(150),
            nn.ReLU(),
            nn.MaxPool2d(2))
        self.layer6 = nn.Sequential(
            nn.Conv2d(150, 50, kernel_size=10, stride=1, padding=0),
            nn.BatchNorm2d(50),
            nn.ReLU(),
            nn.MaxPool2d(2))
        self.fc = nn.Linear(3802500, 10)
```




# Model 3: Increase Number of Layers



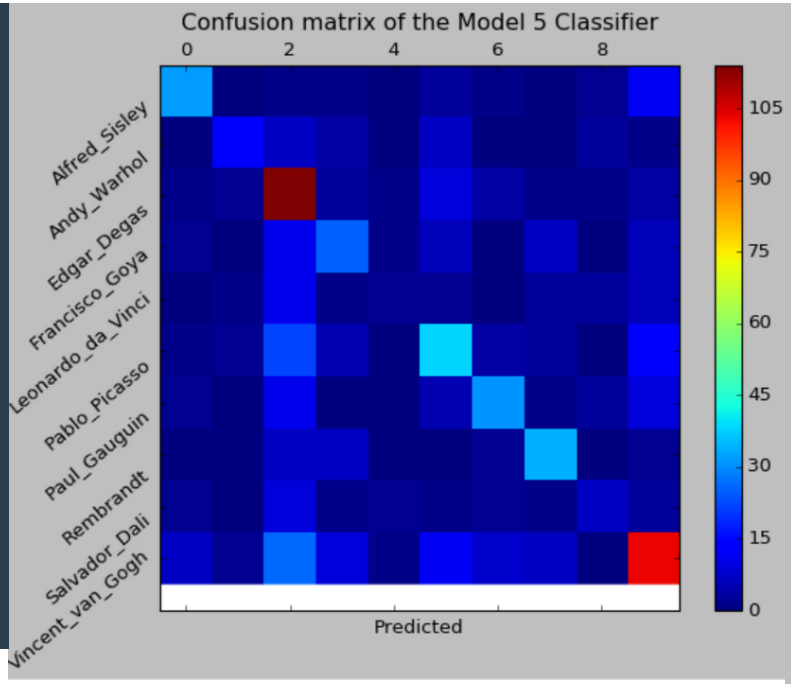



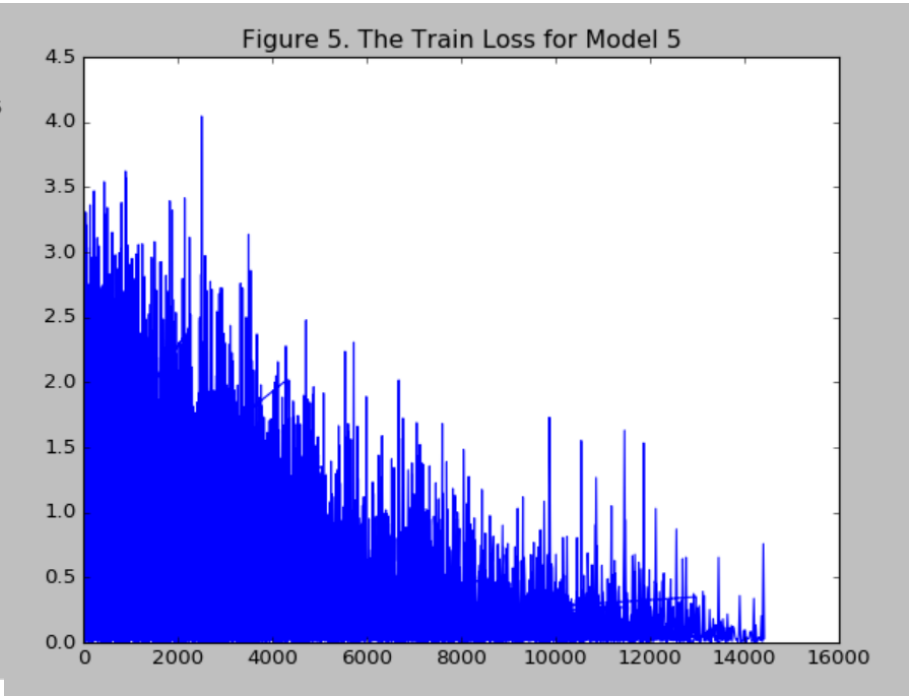

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(3, 500, kernel_size=10, stride=5, padding=2),
            nn.BatchNorm2d(500),
            nn.ReLU(),
            nn.MaxPool2d(4))
        self.layer2 = nn.Sequential(
            nn.Conv2d(500, 1000, kernel_size=10, stride=5, padding=2),
            nn.BatchNorm2d(1000),
            nn.ReLU(),
            nn.MaxPool2d(4))
        self.fc = nn.Linear(1000, 10)
```

# Model 4: Increase Number of Neurons

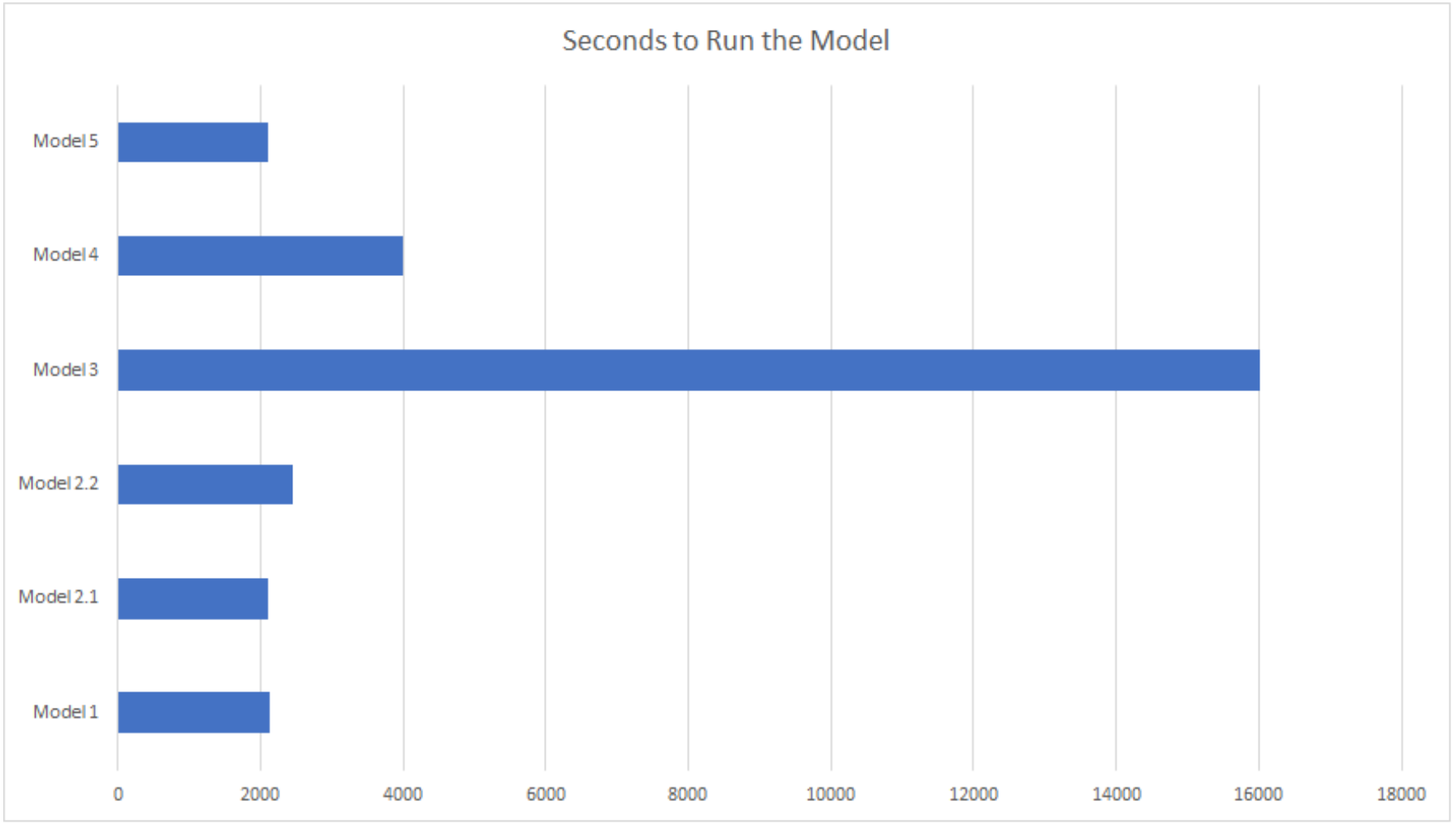
# Model 5: Add Dropout Layers

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(3, 50, kernel_size=10, stride=5, padding=2),
            nn.BatchNorm2d(50),
            nn.ReLU(),
            nn.MaxPool2d(4))
        self.dropout = nn.Dropout2d(p=0.5)
        self.layer2 = nn.Sequential(
            nn.Conv2d(50, 100, kernel_size=10, stride=5, padding=2),
            nn.BatchNorm2d(100),
            nn.ReLU(),
            nn.MaxPool2d(4))
        self.fc = nn.Linear(100, 10)
```

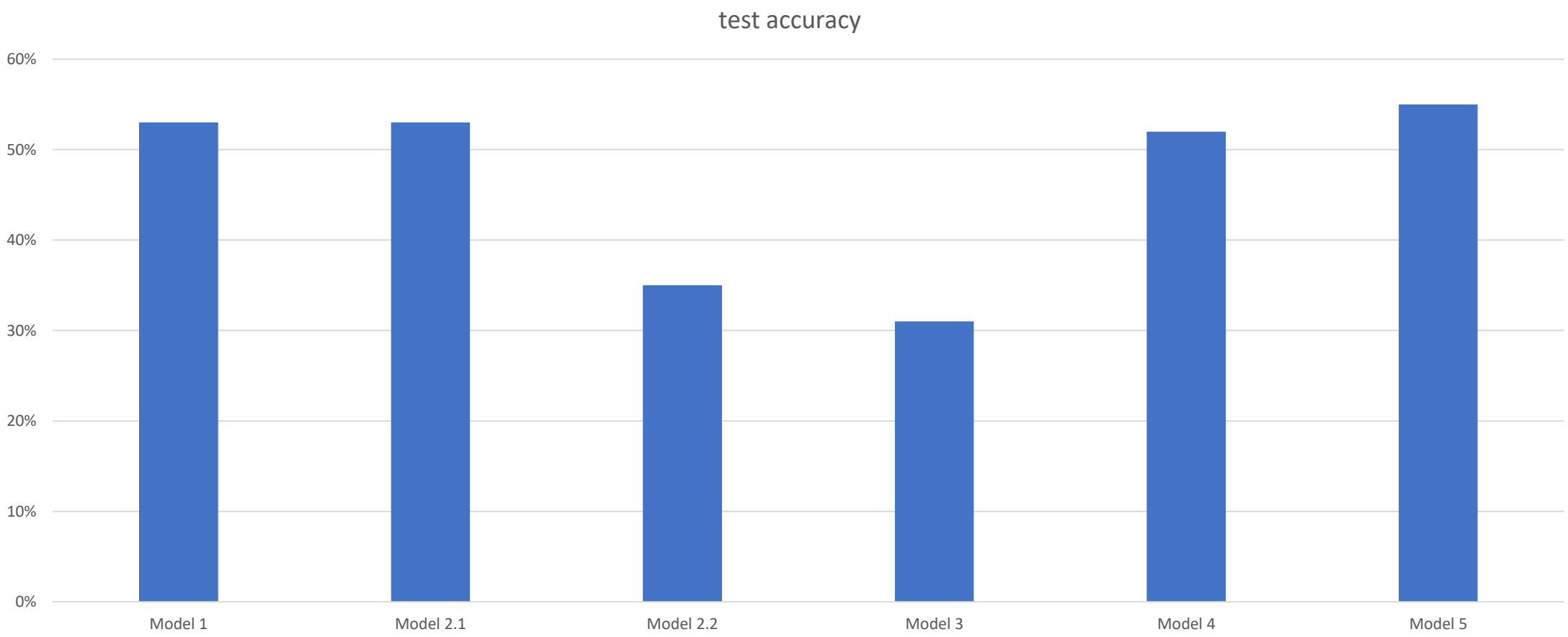


# Time taken to run the models (epoch = 20)



# Test Accuracy



# Future Research

- The data set is not balanced, could experiment with more balanced data in two ways:
  - Sample with replacement in classes that are under-represented
  - Reduce the number of samples in over-represented classes
- The relatively low accuracy score overall is probably due to limited number of images in dataset, need to increase data size.
- Experiment with different convolution layers design.