COL774 Assignment-4

- Pradyumna Meena (2016CS10375)

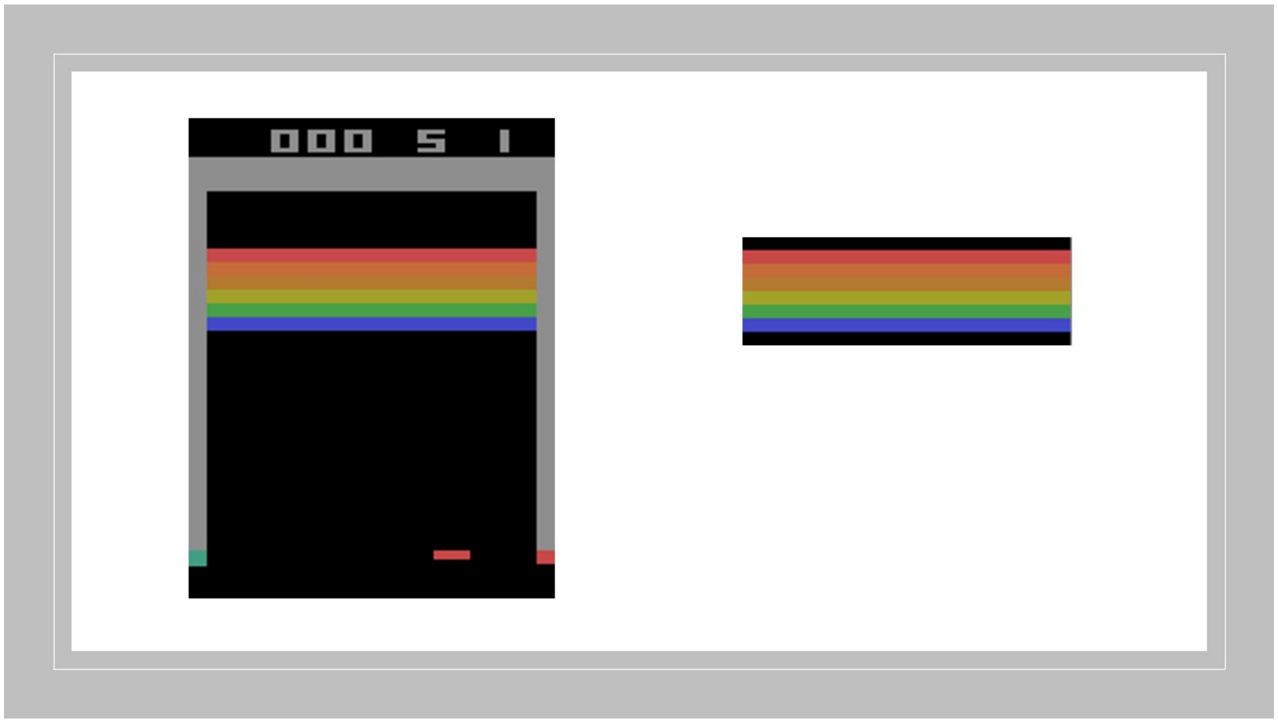
Libraries used

- Numpy
- Pandas
- OpenCV
- Scikit learn
- Pytorch
- Pickle (saving data to avoid re-processing)

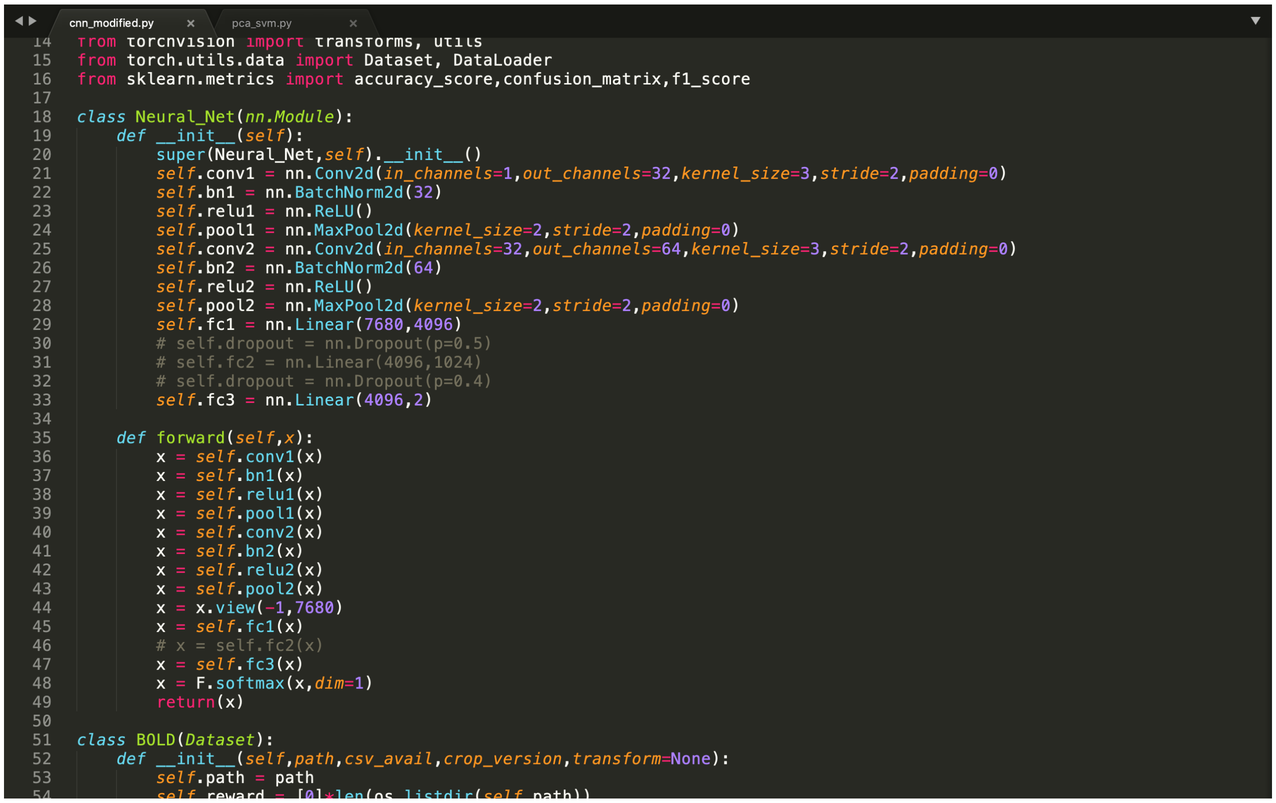
Competitive Part

Architecture used was the same as was provided in the assignment pdf. There were only a few modifications added on top:

- Dataset Modification
    - Instead of taking the whole image I took only the bricks part of the image since all the events are happening only in that region.



- Architecture Details



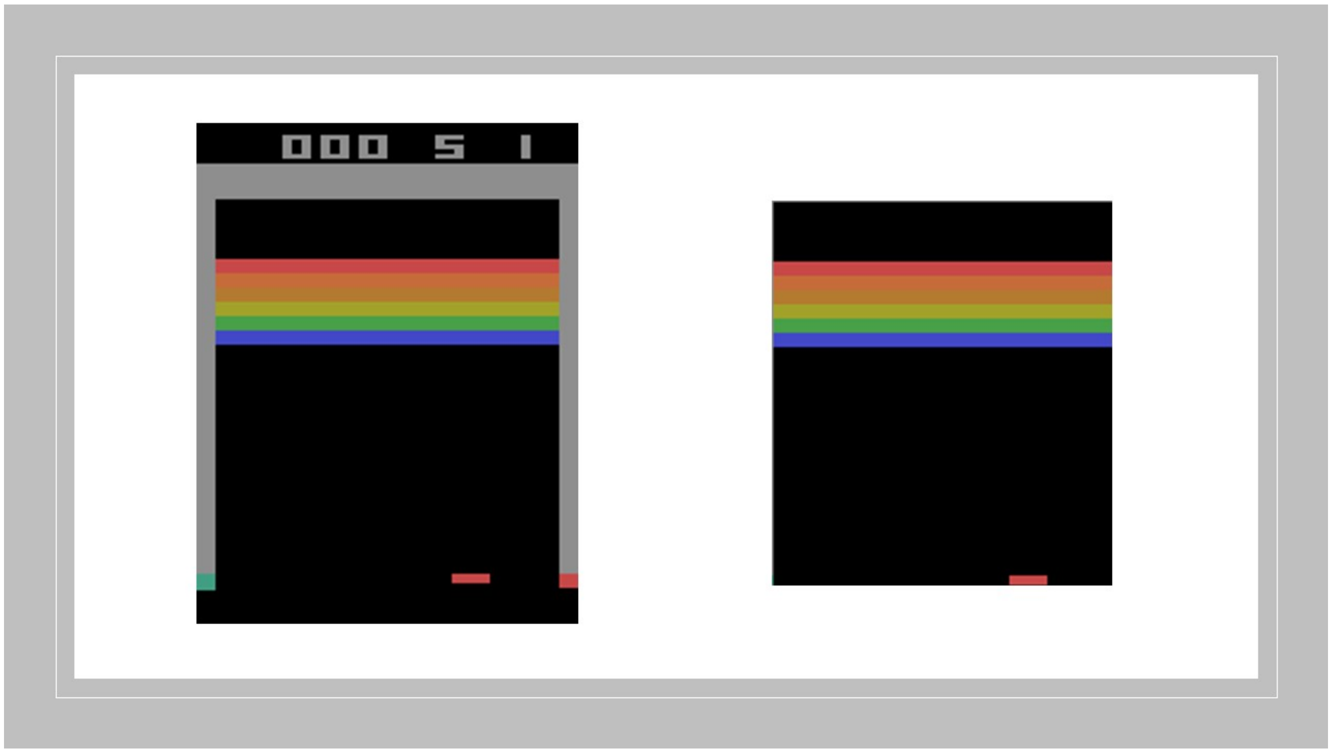
```python
from torchvision import transforms, utils
from torch.utils.data import Dataset, DataLoader
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score

class Neural_Net(nn.Module):
    def __init__(self):
        super(Neural_Net,self).__init__()
        self.conv1 = nn.Conv2d(in_channels=1,out_channels=32,kernel_size=3,stride=2,padding=0)
        self.bn1 = nn.BatchNorm2d(32)
        self.relu1 = nn.ReLU()
        self.pool1 = nn.MaxPool2d(kernel_size=2,stride=2,padding=0)
        self.conv2 = nn.Conv2d(in_channels=32,out_channels=64,kernel_size=3,stride=2,padding=0)
        self.bn2 = nn.BatchNorm2d(64)
        self.relu2 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(kernel_size=2,stride=2,padding=0)
        self.fc1 = nn.Linear(7680,4096)
        # self.dropout = nn.Dropout(p=0.5)
        # self.fc2 = nn.Linear(4096,1024)
        # self.dropout = nn.Dropout(p=0.4)
        self.fc3 = nn.Linear(4096,2)

    def forward(self,x):
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu1(x)
        x = self.pool1(x)
        x = self.conv2(x)
        x = self.bn2(x)
        x = self.relu2(x)
        x = self.pool2(x)
        x = x.view(-1,7680)
        x = self.fc1(x)
        # x = self.fc2(x)
        x = self.fc3(x)
        x = F.softmax(x,dim=1)
        return(x)

class BOLD(Dataset):
    def __init__(self,path,csv_avail,crop_version,transform=None):
        self.path = path
        self.reward = [0]*len(os.listdir(self.path))
```

- Dataloader Class
    - This was used for faster loading of dataset using parallelization.
- Optimization details
    - Learning rate = 0.01
        - Loss was becoming saturated so I increased the learning rate from 0.001 to 0.01
    - Batch size = 32
    - Optimizer = Adam
        - Learning rate = 0.01
        - betas=(0.9,0.999) - Default
        - Epsilon = 1e-8 - Default
- Link to the model -> Drive Link

Non Competitive Part

PCA+SVM:

1. Preprocessing
    a. Conversion to grayscale
    b. Cropping was done to remove the borders from the images as shown below
    c. Normalization was also done (divided by 255.0) and then converted to float32 as float64 was giving memory error



2. Accuracies
    a. Linear SVM with penalty = 1
        i. Training Data Accuracy = 0.4637382749942805
        ii. Validation Data Accuracy = 0.3875
    b. Gaussian SVM with penalty = 1 and gamma = 0.05
        i. Training Data Accuracy = 0.978473634610097
        ii. Validation Data Accuracy = 0.9729310344827586
    c. Linear SVM with penalty = 5
        i. Training Data Accuracy = 0.5773813646657809
        ii. Validation Data Accuracy = 0.6420689655172414
    d. Gaussian SVM with penalty = 5 and gamma = auto
        i. Training Data Accuracy = 0.6304143867315648
        ii. Validation Data Accuracy = 0.6857758620689656
3. Observations
    a. Gaussian SVM is always performing better than the linear version because the given data is clearly not linearly separable and hence we need more complex curves to fit betterly. Though increasing penalty is also not helping for gaussian version.
    b. Reducing the number of components for PCA somehow helped in increasing the accuracy over the test data provided at kaggle.
    c. Another observation was that scikit learn SVM works remarkably faster than LIBSVM.

CNN

1. No preprocessing what so ever. Only change done was to stack up 5 images together.

2. Accuracies
    a. Validation Data Accuracy = 93% (most of them were zeros)
    b. Training Data Accuracy = Since all were zeros and a few ones the accuracy was over training data was very low.
3. Observations
    a. Initially the model predicts all zeros and then as it starts to learn it starts predicting 1.
    b. Adam optimizer saturated the loss very quickly in the second iteration only. So I used SGD for optimization purpose and it was working good.