COL341 : Machine Learning
Assignment 3

Gaurav Jain - 2019CS10349 and T Abishek - 2019CS10407

1 Week 1 Submission

The following experiments were performed in Week 1:

- We started our work by implementing the baseline that is directly classifying the yoga poses into 1 of the 19 given classes using a CNN model.
- The train and test data loader are added by taking code snippets from COL341 Assignment 2.2 starter code.
- We were supposed to implemented some small CNN models with less than 3 million parameters in Assignment 2.2. These models are used with modifications to the output layer as CNN models.
- The first model is:

```
class Net(Module):
def __init__(self):
    super(Net, self).__init__()

    self.cnn_layers = Sequential(
        Conv2d(3, 32, kernel_size=3, stride=1),
        BatchNorm2d(32),
        ReLU(inplace=True),
        MaxPool2d(kernel_size=2, stride=2),

        Conv2d(32, 64, kernel_size=3, stride=1),
        BatchNorm2d(64),
        ReLU(inplace=True),
        MaxPool2d(kernel_size=2, stride=2),

        Conv2d(64, 512, kernel_size=3, stride=1),
        BatchNorm2d(512),
        ReLU(inplace=True),
        MaxPool2d(kernel_size=2, stride=2),

        Conv2d(512, 1024, kernel_size=2, stride=1),
        ReLU(inplace=True),
    )

    self.linear_layers = Sequential(
        Linear(1024 * 1 * 1 , 256),
        ReLU(inplace=True),
        Dropout(p = 0.2),
        Linear(256 * 1 * 1 , 19),
    )
```

  This model consists of 4 convolution layers and 2 linear layers. Batch Normalization is performed after every convolution layer and ReLU is used for activation function. MaxPool is used to reduce the image size. The output dimension of the last linear layer is 19 indicating the number of classes for classification.

  The test accuracy achieved using this model is **52%** on average.

- The second model is:

```
class Net_drop_1(Module):
def __init__(self):
    super(Net_drop_1, self).__init__()

    self.cnn_layers = Sequential(

        Conv2d(3, 32, kernel_size=3, stride=1,padding=1),
        BatchNorm2d(32),
        ReLU(inplace=True),
        Dropout(p = 0.2),

        Conv2d(32, 64, kernel_size=3, stride=1,padding=1),
        BatchNorm2d(64),
        ReLU(inplace=True),
        MaxPool2d(kernel_size=2, stride=2),
        Dropout(p = 0.2),

        Conv2d(64, 128, kernel_size=3, stride=1,padding=1),
        BatchNorm2d(128),
        ReLU(inplace=True),
        MaxPool2d(kernel_size=2, stride=2),
        Dropout(p = 0.2),

        Conv2d(128, 128, kernel_size=3, stride=1,padding=1),
        BatchNorm2d(128),
        ReLU(inplace=True),
        MaxPool2d(kernel_size=2, stride=2),
        Dropout(p = 0.2),

        Conv2d(128, 256, kernel_size=3, stride=1,padding=1),
        BatchNorm2d(256),
        ReLU(inplace=True),
        MaxPool2d(kernel_size=2, stride=2),
        Dropout(p = 0.2),

        Conv2d(256, 512, kernel_size=3, stride=1,padding=1),
        ReLU(inplace=True),
        Dropout(p = 0.2),
    )

    self.linear_layers = Sequential(
        Linear(512*4*4 , 512),
        ReLU(inplace=True),
        Dropout(p = 0.2),
        Linear(512, 64),
        ReLU(inplace=True),
        Dropout(p = 0.2),
        Linear(64 , 19),
    )
```

The main difference between the first and second model is the addition of dropout in convolution layers. Dropout with probability 0.2 is added to the model to deactivate some neurons while training. An additional convolution layer is also added to the model for better accuracy.

Dropout is also added to the linear layers as well. An additional linear layer is also present in the model. The test accuracy given by this model is around **56%** on average.

Actually, this accuracy is not achieved by using not only this model but image transforms are also added to the data loader. The size of the input is change to 64x64 and *random horizontal flip* is added to the training dataset.

- After implementing the above two models and seeing their accuracy scores, we modified a well known pretrained CNN - ResNet for the task at hand. We changed the output size of the only linear layer to 19 to fit our problem.

  We used *resnet18* present in *torchvision.models*. The accuracy achieved using this model is mere **48%**. This is lower than our own CNN models.

  Next, we tried another famous pretrained model, *googlenet* present in *torchvision.models*. The accuracy achieved is relatively good compared to *resnet18*. The accuracy is around **60%**.

- **Note.** We train for 20 epochs for all model implementations.

Thus, after completing the above experiments, we decided to submit the baseline implementation with *googlenet* CNN model for Week 1 submission.

2 Week 2 Submission

After achieving accuracy of **60%** using *googlenet*, we modified other well-known pretrained CNN models for our problem present in *torchvision.models*. We modified Inception v3 model (*models.inception_v3*) and changed image size to 299x299 to fit the model. We achieved an accuracy of around **67%** on the test dataset. We are training for 20 epochs to slightly modify the weights of the model. We didn't perform any more experiments in this week and submit this model for this week submission.

3 Week 3 Submission

There are no new changes in this week submission compared to last week submission and same files are submitted for this week submission.

4 Week 4 Submission

There is no change in the CNN model used in the problem. Inception v3 is used as our CNN model. We vary the number of training epochs while training and calculate accuracy from 1 to 20 epochs. We achieve best accuracy while training for 9 epochs. Thus, this version is submitted for this week submission.