COMPUTER VISION
Assignment 5 Report (Tournament Arc)

Samartha S M

2018101094

Introduction

Deep Learning (DL) (also known as **Deep Structured Learning**) is part of broader **Machine Learning (ML)** based on **Artificial Neural Networks (ANNs)** with representation learning. **Convolutional Neural Networks (CNNs)** are a class of **Deep Learning Neural Networks**. They're very useful for image classification and recognition. We are using **Convolutional Neural Networks (CNNs)** for the purpose of the Image Classification problem as part of assignment 5. The framework used for this purpose is **PyTorch**. The main parts of the code were writing data loaders, model and training code. This report contains the study of various parametric choices of the **Deep Learning (DL)** model architecture design.

Transfer learning

I have used wide range of pre trained models to perform transfer learning, which are as follows,

- **resnet50**
- **resnet152**
- **wide_resnet_101_2**
- **vgg19_bn**

But however, with **resnet152**, I was able to obtain an F-score of **0.477** in the AICrowd challenge.

But later, I used **resnet50** with data augmentation through which I obtained an F-score of **0.51** in the AICrowd challenge.

Data augmentation types I used are as follows,

- Rotation by 90 deg
- Rotation by 180 deg
- Horizontal flipping
- Vertical flipping

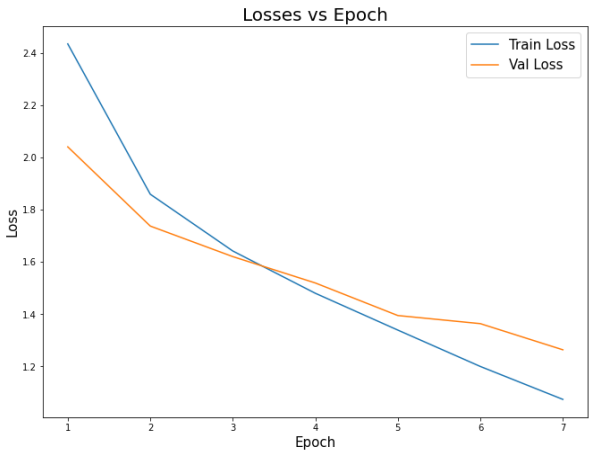
Due to this, I increased the training and validation dataset by **5** times (including original images). Hence, the increased amount of data also helped the pre trained model to learn better.

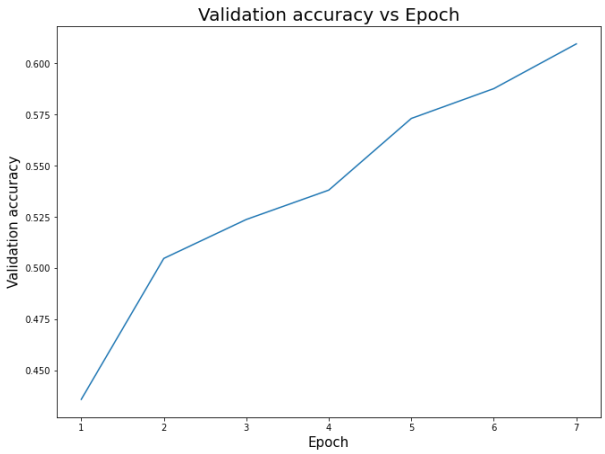
Model Architecture

```
model = models.resnet50(pretrained=True)

for param in model.parameters():
    param.requires_grad = False

num_features = model.fc.in_features

model.fc = nn.Sequential(
    nn.Linear(in_features, 1024),
    nn.ReLU(),
    nn.Linear(1024, 512),
    nn.ReLU(),
    nn.Linear(512, 256),
    nn.ReLU(),
    nn.Linear(256, 128),
    nn.ReLU(),
    nn.Linear(128, 62)
)
```

We can see that I have imported the **resnet50** model from torchvision. I have set requires_grad attribute of all parameters in the model to false so that the model's parameters (weights) won't change during training, because we only want the last fully connected linear layers to be trained. Hence, I have replaced the fully connected linear layers with the above shown sequential layer. This is necessary because the pre-trained model is designed for a different number of classes. We need to change it to accommodate our data which has a different number of classes.

Graph Plots





We can see from the graphs that the model was still not overfitting, because the validation loss is still not increasing. Hence, there was still further scope of learning. But as I had already instantiated the model with just 7 epochs, these are the plots I got. The minimum validation loss in these 7 epochs is **1.2636**. The maximum validation accuracy attained in these 7 epochs is **60.95%**.

With this transfer learned model, I got an F-score of **0.51** in AICrowd challenge.