MLNS Deep Learning Assignment

Part 1

- **Task :** Predicting sum of digits based on image containing multiple digits
- **Dataset :** Images containing multiple digits and the sum of all digits as label
- **Baseline :** We can directly use a CNN to extract features from the images and learn across the entire dataset. There can be 2 ways to treat this problem :
    1. As Regression :
        - Here, we predict real values for each image.
        - The loss is the MSE (mean squared error) between the actual sum label and the predicted value.
        - We can calculate a final MSE/MAE score for evaluation.
        - Accuracy will not make much sense in this case since we can predict any real number value, not just the integer sums.
    2. As Classification :
        - Here, we consider 37 distinct classes (Possible sum values : 0-36)
        - The prediction is done over one of the classes and model is trained as such using Cross Entropy loss.
        - Here, we can calculate an average MAE across all images and also define accuracy.
        - The accuracy is very low since it is directly trying to learn sums and not to first recognise digits and then add them.
- **Model Used**

```
# Baseline CNN Model Architecture
class BaselineCNN(nn.Module):
```

```
    def __init__(self, num_classes=1):
        super(BaselineCNN, self).__init__()
        self.conv_layers = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, stride=1, padding=1)
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
        )
        self.fc_layers = nn.Sequential(
            nn.Flatten(),
            nn.Linear(64*420, 128),
            nn.ReLU(),
            nn.Linear(128, num_classes)
        )

    def forward(self, x):
        x = self.conv_layers(x)
        x = self.fc_layers(x)
        return x
```

- **File Structure**
    1. models/ : contains the saved models after training
    2. training.ipynb : loads datasets, defines models, trains for both approaches, saves model
    3. inference.ipynb : loads saved models and tests over testing instances