Deep Learning Assignment

Part - 2
Vyom Goyal (2021101099)

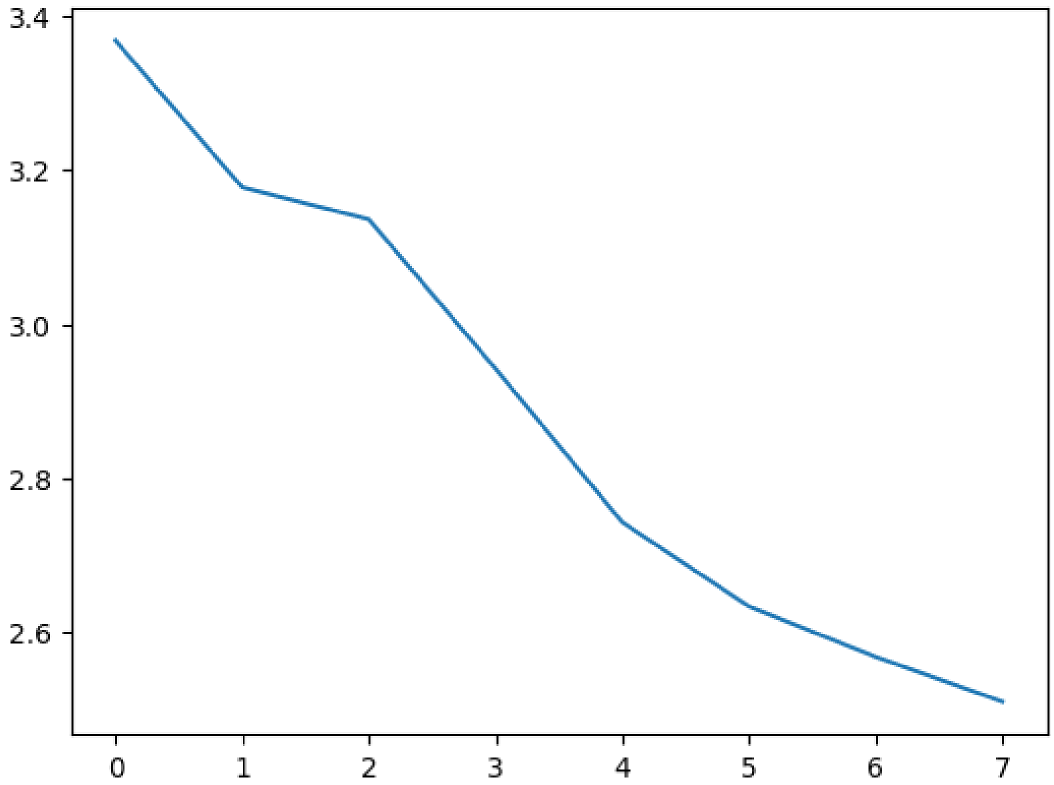
- **Task :** Predict the sum of all the digits present in the image
- **Dataset :** The given dataset contains images with multiple digits and the sum of these digits as labels.
- **Baseline Model :**
    - We can use a simple CNN model and train it for this classification problem with possible classes being 0-36 as the numbers are 4-digit numbers.
    - The Loss function used is Cross entropy loss and the optimizer used is Adam.

```
class CNNModel(nn.Module):
    def __init__(self, num_classes=37):
        super(CNNModel, self).__init__()

        self.conv1 = nn.Conv2d(in_channels=1, out_channels=32
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=6
        self.conv3 = nn.Conv2d(in_channels=64, out_channels=1

        self.fc1 = nn.Linear(128 * 5 * 21, 256)
        self.fc2 = nn.Linear(256, num_classes)

        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.dropout = nn.Dropout(0.7)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))

        x = x.view(x.size(0), -1)

        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.fc2(x)

        return x
```

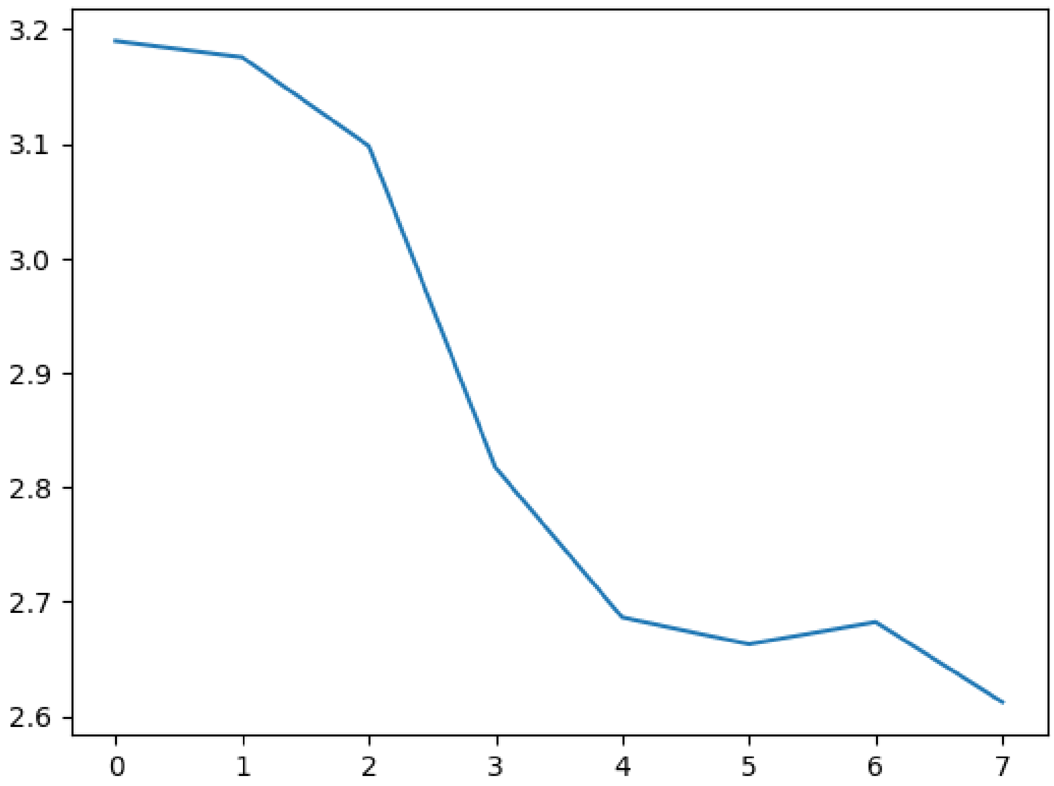
- Final Model: I have used Resnet50 as the base model and then modified it to use it for our task.
    - **Base Model:**
      The model uses ResNet-50, a pre-trained deep convolutional neural network trained on the ImageNet dataset (`weights=models.ResNet50_Weights.IMAGENET1K_V1`). ResNet-50 is well-suited for feature extraction due to its residual architecture, which enables training very deep networks by addressing vanishing gradient issues.
    - **Modified Classification Head:**
      The fully connected layer of ResNet-50 is replaced with a custom head:
        - A linear layer reduces the dimensionality from `in_features` (the number of features from the original ResNet's global average pooling layer) to 128.
        - A ReLU activation introduces non-linearity, followed by a dropout layer with a rate of 0.3 to prevent overfitting.
        - Another linear layer maps the output to `num_classes` for classification.
    - The design allows easy adaptation for any classification task by specifying `num_classes`, enabling fine-tuning on new datasets while leveraging pre-trained ResNet-50's powerful feature extraction capabilities.
    - The Loss function used is Cross entropy loss and the optimizer used is Adam.

```
class ResNetForClassification(nn.Module):
    def __init__(self, num_classes):
        super(ResNetForClassification, self).__init__()
        self.resnet = models.resnet50(weights=models.ResNet5
        in_features = self.resnet.fc.in_features
        self.resnet.fc = nn.Sequential(
            nn.Linear(in_features, 128),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(128, num_classes)
        )

    def forward(self, x):
        return self.resnet(x)
```

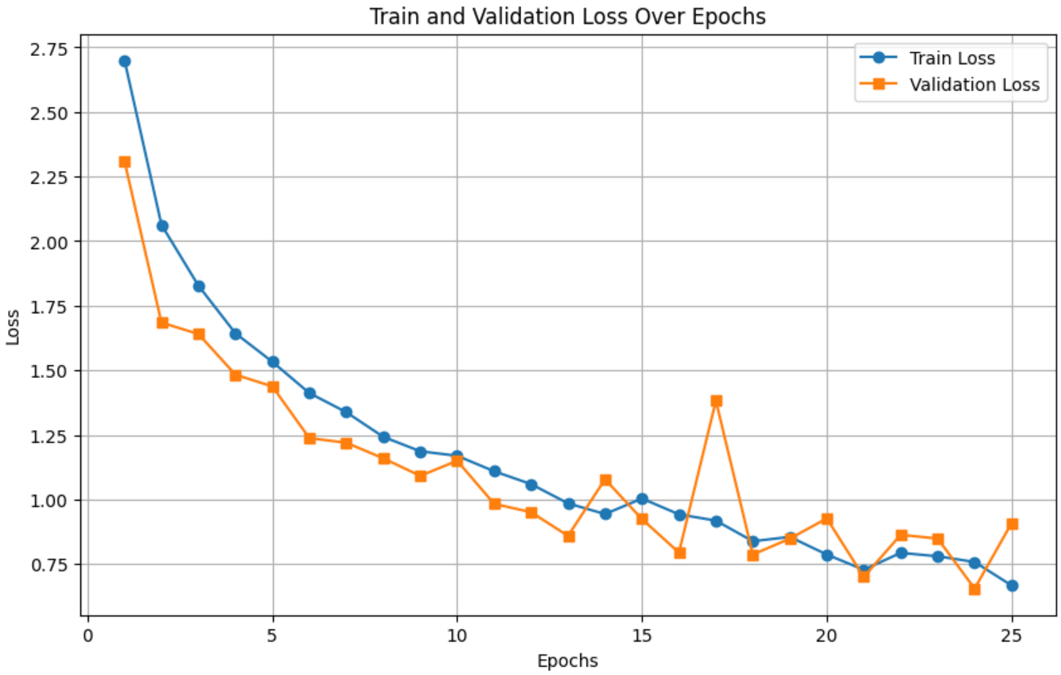
- **Results Obtained:**
  - **Base Model:** The accuracy obtained on the validation dataset is ~ 13.5% . The results obtained are not very good as the simple CNN model cannot learn the task accurately and requires more sophisticated techniques.
    - Training Loss:



- Val Loss:



- Final Res-net model: The accuracy obtained on the test dataset is around 84-85% while the accuracy on the entire dataset is around 90-91%.
    - Train and Val Loss:



- **Hyper parameters used:**
  - Baseline Model:
    - batch size: `64`
    - learning rate: `1e-3`
    - num_epochs: `8`
    - dropout rate: `0.7`
  - Res-net Model:
    - batch_size: `64`
    - dropout rate: `0.3`
    - num_epochs: `25`
    - learning rate: `5e-4`
- **File Structure:**
  - `training.ipynb`: Loads, data and trains our CNN model. The trained model is then saved.
  - `inference.ipynb` : Loads the model and performs testing.

    - `model.pth`: Saved CNN model