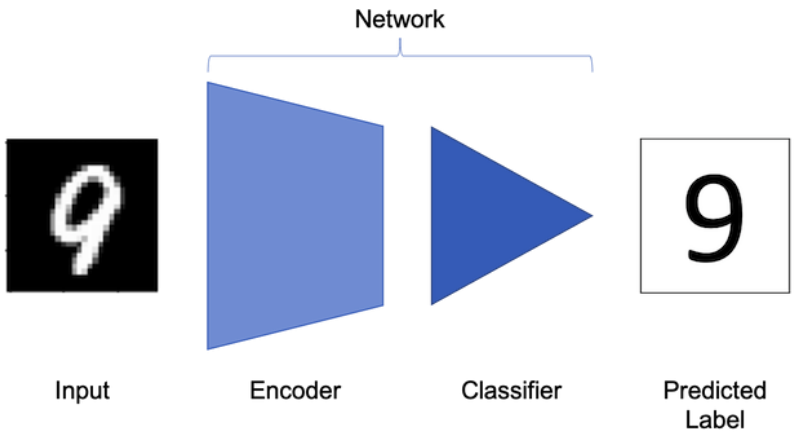
HW8 Autoencoder for MNIST in PyTorch Lightning

B083040029 邱品諺

MNIST 資料集為手寫辨識的資料集，具有 60000 筆訓練資料及 10000 筆測試資料，且每筆資料為 28x28、數字 0-9 的圖片。由於本次作業的資料集中，標注資料僅有 300 筆（train、val、test 各 100 筆），除了可利用之前提過的 Data Augmentation 增加資料外，本作業將運用 Autoencoder 解決此問題。

A Simple Classifier

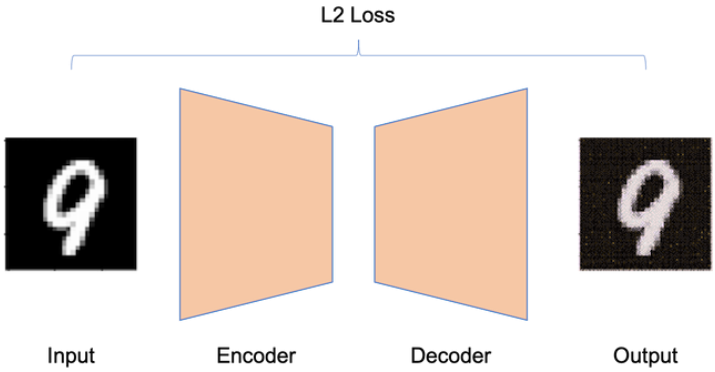


首先，Encoder 的作用主要為擷取 Input 中有用的資訊，再將這些萃取出的資訊送入 Classifier 進行分類，下圖分別為 Encoder、Classifier 的網路架構，及 hyperparameters。

```
self.encoder = nn.Sequential(
    nn.Linear(input_size, self.hparams["n_hidden_en"]),
    nn.ReLU(),
    nn.Linear(self.hparams["n_hidden_en"], self.hparams["n_hidden_en"]),
    nn.ReLU(),
    nn.Linear(self.hparams["n_hidden_en"], self.hparams["latent_dim"])
)
self.model = nn.Sequential(
    nn.Linear(self.hparams["latent_dim"], self.hparams["n_hidden_cl"]),
    nn.ReLU(),
    nn.Linear(self.hparams["n_hidden_cl"], 10)
)
```

```
hparams = {
    "learning_rate": 0.001,
    "n_hidden_en": 512,
    "n_hidden_cl": 64 ,
    "batch_size": 8,
    "latent_dim": 128
}
```

Autoencoder



Autoencoder 有兩部分：Encoder 及 Decoder，Encoder 擷取出 Input 的重要特徵，Decoder 利用 Input 擷取出的特徵進行生成還原，為了使輸出近似於 Input，因此將會計算 Input 及 Output 之間的 loss，使 Output 盡量地接近 Input，而 Autoencoder 會有 low dimension bottleneck 的問題。利用未標注資料訓練完 Autoencoder 後，Encoder 其擷取資訊的能力更強，因此保留 Encoder 的 model parameters 並將 Decoder 改成 Classifier 再對標注資料進行訓練測試，此方式稱為 Transfer learning，可增加模型的準確率。

下圖為 Decoder 的網路架構及 Autoencoder 的 hyperparameters：

```
self.decoder = nn.Sequential(
    nn.Linear(self.hparams["latent_dim"], self.hparams["n_hidden_en"]),
    nn.ReLU(),
    nn.Linear(self.hparams["n_hidden_en"], self.hparams["n_hidden_en"]),
    nn.ReLU(),
    nn.Linear(self.hparams["n_hidden_en"], output_size)
)
```

```
hparams = {
    "learning_rate": 0.001,
    "n_hidden_en": 512,
    "batch_size": 8,
    "latent_dim": 128
}
```

最後，由 validation 準確率可得知利用 Autoencoder pretrain 後，由於利用 Unlabeled data 進行 training 使得 Encoder 擷取資訊的能力更強，因此得出比未使用 autoencoder pretrain 還高的準確率。

```
Validation accuracy when training from scratch: 53.0%
Validation accuracy with pretraining: 65.0%
Test accuracy when training from scratch: 52.0%

Now to the pretrained classifier:
Validation-Accuracy: 65.0%
FYI: Your model has 0.739 mio. params.
Great! Your model size is less than 20 MB and will be accepted :)
Your model has been saved and is ready to be submitted. NOW, let's check the test-accuracy.
Test-Accuracy: 71.0%
```