FCN con PyTorch

Tinoco Sergio.
Instituto Politécnico Nacional, Escuela Superior de Cómputo.
Redes neuronales y aprendizaje profundo

25 de abril de 2023

En el presente documento se va a presentar la implementación y entrenamiento de una red neuronal completamente conectada usando PyTorch.

Para ello se va a utilizar el dataset “MNIST” para el entrenamiento.

```
# Download and load the training data
trainset = datasets.MNIST('MNIST_data/', download=True, train=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64, shuffle=True)
```

Se definen las capas de la red neuronal.

```
# Hyperparameters for our network
input_size = 784
hidden_sizes = [128, 64]
output_size = 10

# Build a feed-forward network
model = nn.Sequential(OrderedDict([
                      ('fc1', nn.Linear(input_size, hidden_sizes[0])),
                      ('relu1', nn.ReLU()),
                      ('fc2', nn.Linear(hidden_sizes[0], hidden_sizes[1])),
                      ('relu2', nn.ReLU()),
                      ('logits', nn.Linear(hidden_sizes[1], output_size))]))
```

Se definen los hiperparámetros de la red.

```
epochs = 10
tasa_de_aprendizaje = 0.005
```

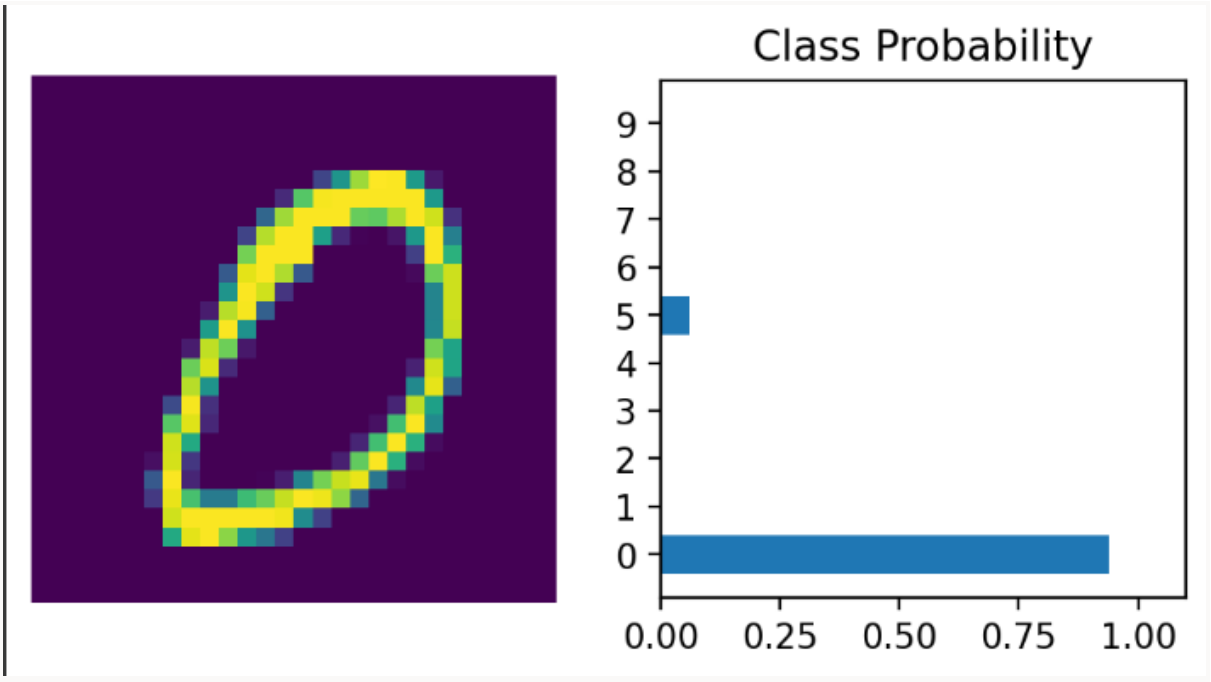
Resultados

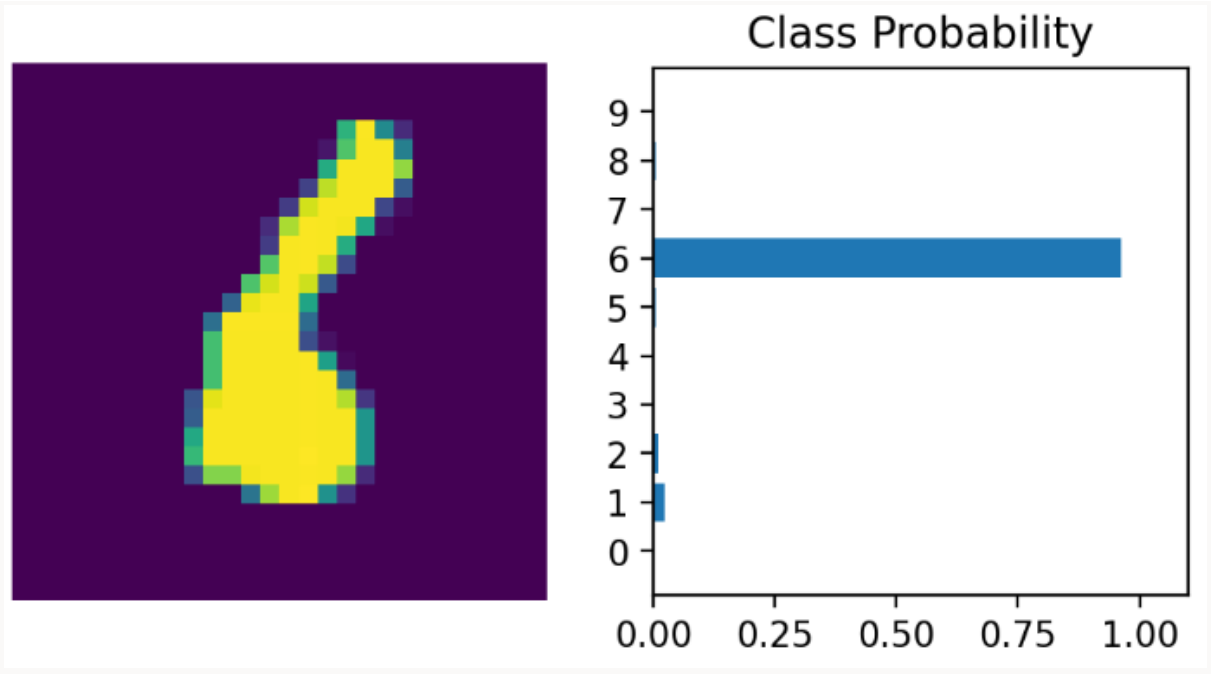
Se realiza el entrenamiento con 10 epocas.

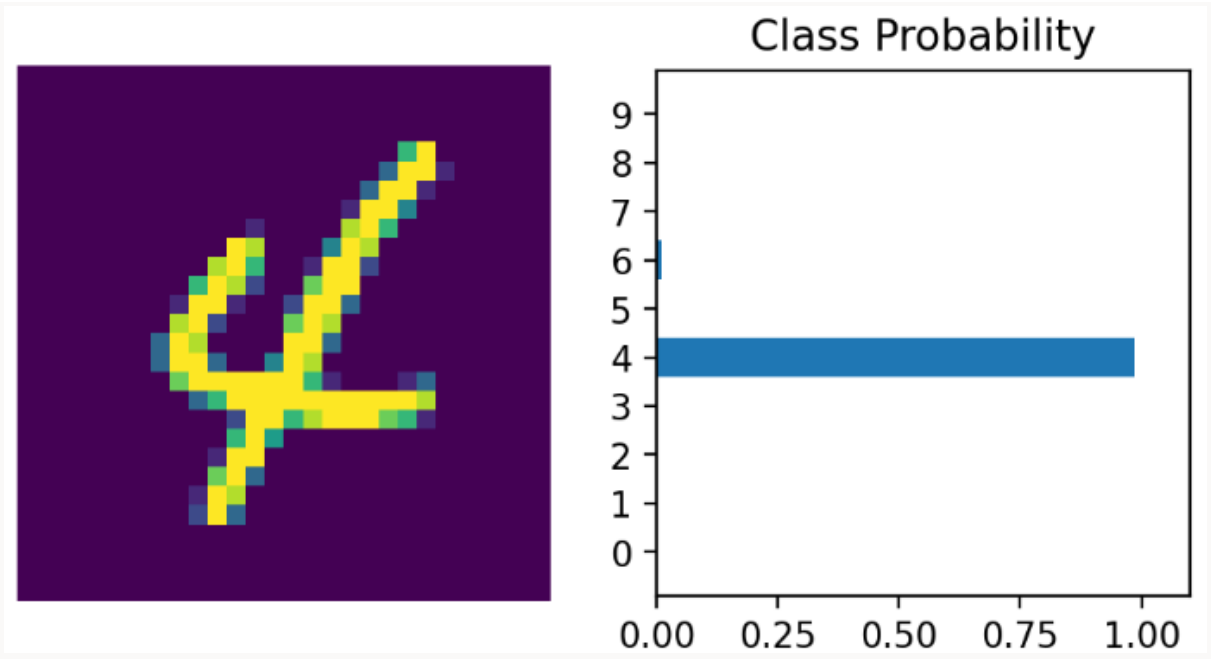
```
Epoch: 10/10...  Loss: 0.1488
Epoch: 10/10...  Loss: 0.1555
Epoch: 10/10...  Loss: 0.1802
Epoch: 10/10...  Loss: 0.1936
Epoch: 10/10...  Loss: 0.1895
Epoch: 10/10...  Loss: 0.1475
Epoch: 10/10...  Loss: 0.1603
Epoch: 10/10...  Loss: 0.1643
Epoch: 10/10...  Loss: 0.1853
Epoch: 10/10...  Loss: 0.1967
Epoch: 10/10...  Loss: 0.1949
Epoch: 10/10...  Loss: 0.1705
Epoch: 10/10...  Loss: 0.1402
```

Como se pudo observar, se alcanzó un “loss” de 0.14 el cual es menor que el loss objetivo (0.2).

Clasificación de las imágenes.







Conclusiones

Como se pudo observar en los resultados, la pérdida fue disminuyendo con cada época, alcanzando un valor menor a 0.2 que era el objetivo en esta práctica y para ello fue requerido aumentar la tasa de aprendizaje a un 0.005 y de igual forma se duplicó el número de épocas hasta 10, y gracias a las herramientas que PyTorch nos ofrece, se pudo realizar el backward pass sin necesidad de calcular las derivadas lo que lo hace más fácil y eficiente.