Tales Araujo Leonidas
Professor Patricia McManus
ITAI 2376: Deep Learning
February 15, 2024

Introducing Convolution Neural Networks (CNNs): Reflective Journal

Introduction

This reflective journal describes and illustrates my experience performing the Lab 4 of the first module of AWS MLU Application of Deep Learning to Text and Image Data. The goal is to share hands-on experience in convolution neural networks by building, training, and testing the performance of a CNN on the MNIST dataset, a large dataset of handwritten digits.

Activities Performed

In this lab, after installing and importing the necessary libraries and its dependencies, such as boto3, pandas, numpy, matplotlib, torch and torchvision, I loaded and printed the dimensions of the train and test datasets, which were included in the torchvision library. the images from the MNIST dataset have been size-normalized and centered with fixed dimensions, so I did not need to perform this task. The entire dataset used consisted in 70000 images of 28 x 28 pixels, as seen in the image below.

```
Training data shape: [60000, 28, 28].
Test data shape: [10000, 28, 28]
```

Then, I defined the hyperparameters, the step size of the learning rate, the use of GPU, and loaded the train and test datasets in batches before creating a neural network with the requested attributes.

Results

With ChatGPT4's assistance, I was able to overcome the biggest challenge in this lab after a few tries, which was creating a neural network with a 2D convolutional layer with these attributes:

- Conv2D layer with `in_channel=1`, `out_channel=32`, and `kernel_size=3`
- Flatten the layer to squash the data into a one-dimensional tensor
- Linear layer with 128 units
- One output layer
- Softmax activation function for the output layer

The code used to fulfill these requirements is pictured below:

```
############### CODE HERE ###############

class CustomCNN(nn.Module):
    def __init__(self):
        super(CustomCNN, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3, padding=0)
        self.fc1 = nn.Linear(26 * 26 * 32, 128)  # Flattened dimension for the linear layer
        self.fc2 = nn.Linear(128, num_classes)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = x.view(-1, 26 * 26 * 32)  # Flatten the output of Conv2D
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Initialize the network
net = CustomCNN().to(device)

############## END OF CODE ##############
```

The final result is CNN's impressive 98% accuracy on the MNIST dataset.

```
     accuracy                           0.98     10000
    macro avg       0.98      0.98      0.98     10000
 weighted avg       0.98      0.98      0.98     10000
```

Key Learnings

- The Conv2D layer serves to extract features from input images through convolution operations, effectively capturing spatial hierarchies in data.
- The MaxPooling2D layer reduces dimensionality and computational complexity by downsampling the feature maps (2D arrays that contain information about the spatial patterns in the image, such as edges, textures, or specific shapes).
- One-hot encoding is employed for categorical data representation, ensuring a clear distinction between classes by converting labels into a binary matrix (zeros and ones), crucial for classification tasks.
- The Flatten layer takes the 2D feature maps and "flattens" them into a long 1D vector. This process doesn't change the data itself but restructures it from a 2D format to a 1D format, making it compatible with the dense layers that follow the network.
- I utilized categorical cross-entropy loss function and the Adam optimizer, respectively, for loss measurement and optimization due to their efficacy in handling sparse gradients and multi-class classification problems.

Works cited

AWS Academy. (2023). Application of Deep Learning to Text and Image Data: Module 1 Lab 4. Retrieved from https://awsacademy.instructure.com/login/canvas