ENPM703- Assignment-2

Part5: Pytorch Implementation

Pytorch

PyTorch is a free, open-source machine learning library created by Facebook's AI Research team. It is based on tensors, which are similar to NumPy arrays, but with extra features for using GPUs, making it ideal for deep learning projects. PyTorch provides a strong framework for building deep learning applications, known for being user-friendly, having dynamic computation graphs, and focusing on performance and flexibility. Its API offers both low-level and high-level tools, making it suitable for a variety of uses.

Sample Functions

Functions in torch.nn.functional:

1. **relu():** Applies the Rectified Linear Unit (ReLU) activation function to each element, adding non-linearity to the model.
2. **cross_entropy():** Calculates the cross-entropy loss between predicted outputs and true labels, often used for classification tasks.
3. **conv2d():** Performs a 2D convolution on input data, applying the weights of a convolutional layer to the input tensor.

Functions in NN Module API:

1. **Linear(in_features, out_features):** Creates a fully connected layer that transforms the input data using a linear function, taking the number of input and output features.
2. **Conv2d(in_channels, out_channels, kernel_size):** Sets up a 2D convolutional layer used in image processing, requiring the number of input channels, output channels, and kernel size.
3. **BatchNorm2d(num_features):** Applies batch normalization to the output from a convolutional layer, helping to stabilize and speed up training by normalizing activations.

Advantages of Pytorch

1. **Flexibility and Ease of Use:**
   - PyTorch has dynamic computation graphs that allow for easy model building and debugging. It features a user-friendly interface and straightforward syntax, making it accessible for beginners and efficient for experienced users. Its flexible APIs support both low-level adjustments and high-level quick development.
2. **Performance and Efficiency:**
   - With built-in GPU support, PyTorch allows for faster training of deep learning models. It also makes it easy to save and load models, simplifying the process of deploying and sharing them.

NN Module API and Sequential API

The **NN Module API** is part of PyTorch that offers classes and functions for creating neural networks. It includes a variety of pre-defined layers, such as convolutional, recurrent, and linear layers, as well as tools for setting up loss functions and optimizers.

The **Sequential API** provides an easy way to build models by stacking layers in order, making it simple to construct models without needing to create a custom forward method unless necessary.

Open Ended Model

- Model includes two convolutional layers, each followed by batch normalization, ReLU activation, and max pooling, which help to extract and downsample features from the input images.
- The first convolutional layer has 64 channels, with a kernel size of 5, and includes padding of 2. The second layer has 32 channels, a kernel size of 3, and padding of 1.
- After extracting features, the model flattens the output and feeds it through three fully connected layers with 256, 128, and 10 output units, respectively, using ReLU activation and dropout to help prevent overfitting.
- The model is trained using the Adam optimizer with a learning rate of 1e-3 and a weight decay of 1e-4 to enhance generalization.

Tunning the Model

- **Lr:** 1e-3 was the sweet spot of getting optimal results in optimal time. increasing the value made the accuracy to dip down.
- **Weight Decay:** Values of range 1e-4 to 5e-4 produced optimal accuracies. Having this parameter will prevent overfitting of networks.
- **Filter Size:** Values of channel_1 = 64, channel_2 = 32 means more channels which help the model capture a wider range of features, prevent vanishing gradients, and retain more information during backpropagation, improving generalization on unseen data.
- **No. of Filters:** Both size (3,1) and (5,3) produced similar results but combination (5,1) performed poorly. In later, large filter size and a small filter size might not effectively extract useful details.
- **Fcc Dimension:** Fcc output dimensions 128->64->10 and 256->64->10 produced similar results but dimension 256->128->10 performed great. The added layer might help the model discover more useful patterns and enhance its ability to generalize to new data.