DeepLearning

LeNet-5 FMNIST Training and Evaluation

This script implements and trains the LeNet-5 network using PyTorch and the FashionMNIST dataset. The purpose of this task is to compare different regularization techniques and analyze their impact on model performance.

Prerequisites

- Python 3.10+
- PyTorch
- NumPy
- Matplotlib
- Google Colab (Optional: to run the script)

Training

We have a model configuration instance that contains different configurations of the LeNet-5 model. Each configuration applies a different regularization technique. The `models_config` dictionary stores these models and their respective parameters for training.

For example, the following dictionary contains four different configurations of the LeNet-5 model:

```
models_config = {
    "model_no_reg": {"model": LeNet5().to(device), "params": {}},
    "model_with_dropout": {"model": LeNet5(dropout=True).to(device), "params":
{}},
    "model_with_batchnorm": {"model": LeNet5(batch_norm=True).to(device),
"params": {}},
    "model_with_weight_decay": {"model": LeNet5().to(device), "params":
{"weight_decay": 1e-4}}
}
```

Then, we train each model in the models_config dictionary with different learning rates(0.01, 0.001, 0.0001). Refer to **Training** section in the script for more details.

At the end of the training process, the best model for each regularization technique will be saved in result dictionary along with all the meta info:

- name
- model
- model parameters
- learning rate
- test accuracy(final)
- train accuracies(epoch-wise)
- validation accuracies(epoch-wise)

- test accuracies(epoch-wise)

Testing

As mentioned above, the best model for each regularization technique is saved in the result dictionary. To test the model with the saved weights for each case, you can load the model using the following code:

```
model_to_test = result["MODEL_NAME"]["model"]
model_params = result["MODEL_NAME"]["parameters"]
model_to_test.load_state_dict(model_params)

# Evaluate the model
test_accuracy = evaluate_model(model_to_test, test_loader)
```

MODEL_NAME can be ["model_no_reg", "model_with_dropout", "model_with_batchnorm", "model_with_weight_decay"]. Note that we also store the final test accuracy for each model in the result dictionary, so the output of test accuracy should be the same as the final test accuracy.