CS 203: Software Tools & Techniques for AI
IIT Gandhinagar
Sem-II - 2024-25
LAB 11
Lead TA: Himanshu Beniwal

OM GUPTA -22110174
PRANAV SOMASE - 22110200

---

1. Dataset Preparation (10%)

- Load the training dataset and test data.
- Use 20% of the training dataset as the validation set.

ChatGPT

```python
import pandas as pd
from sklearn.model_selection import train_test_split

# Load the datasets
train_url = "https://raw.githubusercontent.com/clairett/pytorch-sentiment-classification/master/data/SST2/train.tsv"
test_url = "https://raw.githubusercontent.com/clairett/pytorch-sentiment-classification/master/data/SST2/test.tsv"

train_data = pd.read_csv(train_url, sep="\t")
test_data = pd.read_csv(test_url, sep="\t")

# Split train data into training and validation (80%/20%)
train_df, val_df = train_test_split(train_data, test_size=0.2, random_state=42)
```

2. Construct a **Multi-Layer Perceptron (MLP) model**. (10%)

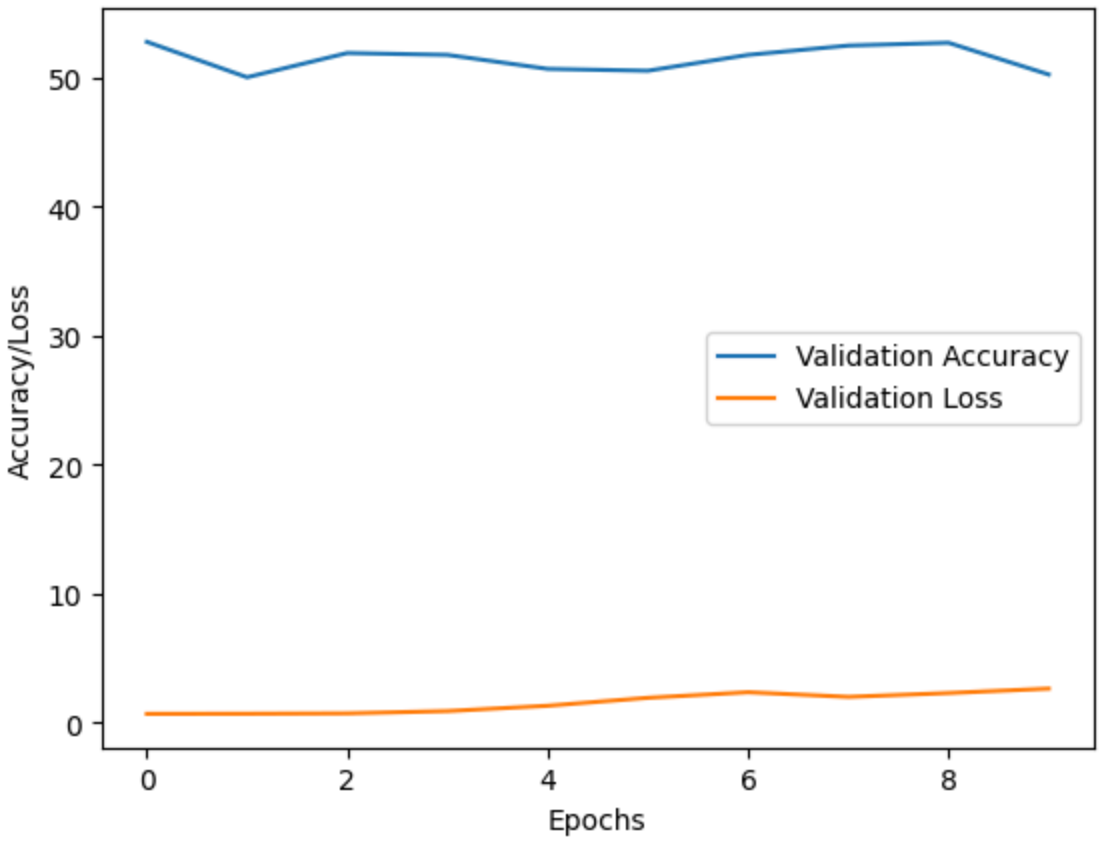
- The parameter should be with:
  - hidden_sizes=[512, 256, 128, 64]
  - Output should have two labels.
  - With the following architecture:

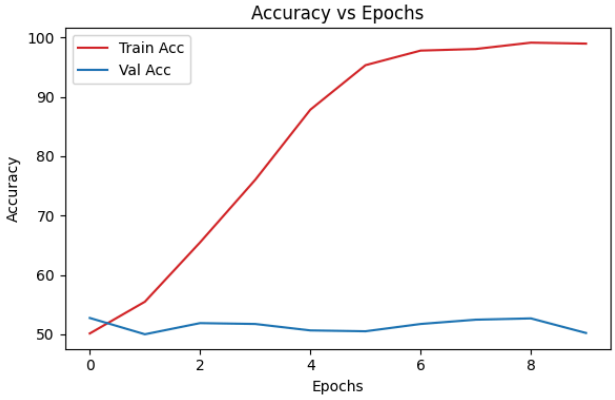
```
print("========== Model Summary ==========")
print(model)
print("\nTotal Trainable Parameters:", count_parameters(model))
```

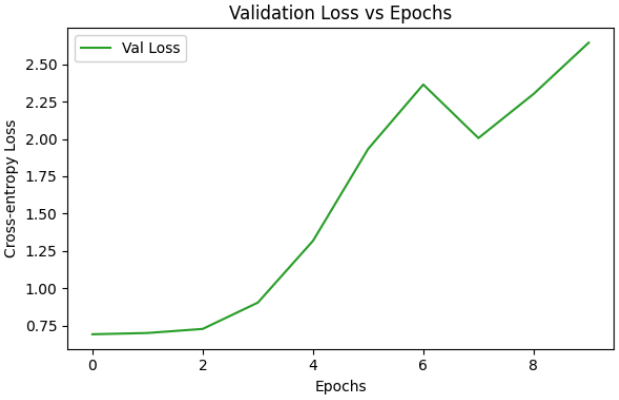
```
========== Model Summary ==========
MLPModel(
  (fc1): Linear(in_features=768, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=256, bias=True)
  (fc3): Linear(in_features=256, out_features=128, bias=True)
  (fc4): Linear(in_features=128, out_features=64, bias=True)
  (fc5): Linear(in_features=64, out_features=2, bias=True)
  (relu): ReLU()
)

Total Trainable Parameters: 566338
```

3. Train the model with 10 epochs and create the best-performing model (checkpoint.pt) (10%)

- Plot the validation accuracy + loss (epochs vs accuracy-loss).


Accuracy/Loss
Epochs
Validation Accuracy
Validation Loss


Accuracy vs Epochs
Train Acc
Val Acc
Accuracy
Epochs


Validation Loss vs Epochs
Val Loss
Cross-entropy Loss
Epochs

```python
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))

# accuracy
ax1.plot(train_accuracies, label="Train Acc",  color="tab:red")
ax1.plot(val_accuracies,   label="Val Acc",    color="tab:blue")
ax1.set_xlabel("Epochs")
ax1.set_ylabel("Accuracy")
ax1.set_title("Accuracy vs Epochs")
ax1.legend()

# validation loss
ax2.plot(val_losses, label="Val Loss", color="tab:green")
ax2.set_xlabel("Epochs")
ax2.set_ylabel("Cross-entropy Loss")
ax2.set_title("Validation Loss vs Epochs")
ax2.legend()

plt.tight_layout()
plt.show()
```

4. Now use:

1. **Dynamic Quantization with INT4 or INT8 (Link: here) (20%)**
    a. Use the torch.quantization.quantize_dynamic()
2. **Half precision (20%)**
    a. Use the .half() function. (Reference: here)

Quantisation

```python
[5] model_dynamic = torch.quantization.quantize_dynamic(
        model, {nn.Linear}, dtype=torch.qint8
    )

    # Save the quantized model
    torch.save(model_dynamic.state_dict(), 'checkpoint_dynamic.pt')
```

```python
model_half = model.half()

# Save the half-precision model
torch.save(model_half.state_dict(), 'checkpoint_half.pt')
```

5. Fill the table for different quantization techniques. (30%)

S.I.	Model Name	Accuracy (Out of 100)	Storage (In MB)	Inference time (In ms)
1.	Original			
2.	Dynamic			
3.	Half			

```python
import pandas as pd

# Create result table
results = {
    "Model Name": ["Original", "Dynamic", "Half"],
    "Accuracy (%)": [best_val_accuracy, val_accuracy_dynamic, val_accuracy_half],
    "Storage (MB)": [original_size, dynamic_size, half_size],
    "Inference Time (ms)": [original_time, dynamic_time, half_time]
}

df_results = pd.DataFrame(results)
print(df_results.to_string(index=False))
```

```
Model Name  Accuracy (%)  Storage (MB)  Inference Time (ms)
  Original     52.745665      2.163973             8.432811
   Dynamic     50.216763      0.549654             1.986309
      Half     50.216763      1.083839            12.067134
```