Assignment 7 - Model Checkpointing

Team 25, Members:

Name	Roll Number
Romit Mohane	23110279
Rudra Pratap Singh	23110281

Introduction

This assignment aims to learn about text classification tasks for checkpoint creation.

Repo link here

Code in Notebook

Screenshots

Model architecture.

```
model = BinaryMLPClassifier(input_size=10000).to('cuda')
summary(model, input_size=(1, 10000))
```

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
BinaryMLPClassifier                      [1, 2]                    --
├─Linear: 1-1                            [1, 512]                  5,120,512
├─ReLU: 1-2                              [1, 512]                  --
├─Dropout: 1-3                           [1, 512]                  --
├─Linear: 1-4                            [1, 256]                  131,328
├─ReLU: 1-5                              [1, 256]                  --
├─Dropout: 1-6                           [1, 256]                  --
├─Linear: 1-7                            [1, 128]                  32,896
├─ReLU: 1-8                              [1, 128]                  --
├─Dropout: 1-9                           [1, 128]                  --
├─Linear: 1-10                           [1, 64]                   8,256
├─ReLU: 1-11                             [1, 64]                   --
├─Dropout: 1-12                          [1, 64]                   --
├─Linear: 1-13                           [1, 2]                    130
==========================================================================================
Total params: 5,293,122
Trainable params: 5,293,122
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 5.29
==========================================================================================
Input size (MB): 0.04
Forward/backward pass size (MB): 0.01
Params size (MB): 21.17
Estimated Total Size (MB): 21.22
==========================================================================================
```

```
params = model.parameters()
total_params = sum(p.numel() for p in params)
print(f"Total number of parameters: {total_params}")
```

```
Total number of parameters: 5293122
```

Hyperparameters

Case 1: BoW

```
def BoW(dataset_train, dataset_val, dataset_test):
  vectorizer = CountVectorizer(max_features=10000)
  # Fit the vectorizer on the training data
  vectorizer.fit(dataset_train['sentence'])
```

Number of max features for vectorizer

```
train_loader = DataLoader(list(zip(X_train_bow, y_train_bow)), batch_size=32)
val_loader = DataLoader(list(zip(X_val_bow, y_val_bow)), batch_size=32)
test_loader = DataLoader(list(zip(X_test_bow, y_test_bow)), batch_size=32)

train_losses, val_losses, val_accs, best_model_dict = train(model_bow, train_loader, val_loader, num_epochs=10, lr = 0.001)
```

Batch size and lr for dataset 1

```
train_loader = DataLoader(list(zip(X_train_bow, y_train_bow)), batch_size=32)
val_loader = DataLoader(list(zip(X_val_bow, y_val_bow)), batch_size=32)
test_loader = DataLoader(list(zip(X_test_bow, y_test_bow)), batch_size=32)

train_losses, val_losses, val_accs, best_model_dict = train(model_bow_pretrained, train_loader, val_loader, num_epochs=10, lr=0.0001)
```

Batch size and lr with previous best model's checkpoint On IMDB (dataset 2)

> **Case 2: BERT**

```
def generate_bert_embeddings(data, batch_size=16):
    """
    Generate BERT embeddings for the input dataset.
    Uses [CLS] token embeddings for sentence representation.
    """
```

Batch Size for the tokenizer

```
train_loader_embeddings = DataLoader(TensorDataset(X_train_embeddings, y_train_embeddings), batch_size=32, shuffle=True)
val_loader_embeddings = DataLoader(TensorDataset(X_val_embeddings, y_val_embeddings), batch_size=32)
test_loader_embeddings = DataLoader(TensorDataset(X_test_embeddings, y_test_embeddings), batch_size=32)

model_bert = BinaryMLPClassifier(input_size=768).to('cuda')
```

```
# ------------------- Train the Model -------------------
train_losses, val_losses, trained_model = train(model, train_loader, val_loader, num_epochs=10, lr=0.001, input_type=input_type)
```

Batch size and lr for dataset 1

```
# Create DataLoaders for Dataset 2
train_loader_embeddings = DataLoader(TensorDataset(X_train_embeddings, y_train_embeddings), batch_size=32, shuffle=True)
val_loader_embeddings = DataLoader(TensorDataset(X_val_embeddings, y_val_embeddings), batch_size=32)
test_loader_embeddings = DataLoader(TensorDataset(X_test_embeddings, y_test_embeddings), batch_size=32)
```

```
# ------------------- Fine-tune BERT Model on IMDB -------------------
train_losses, val_losses, final_model_bert = train(model_bert_pretrained, train_loader_embeddings, val_loader_embeddings, num_epochs=10, lr=0.0001, input_type="BERT_IMDB")
```

Batch size and lr for dataset 2

Logged Metrics **Case 1: BoW**

```
Initial Validation Accuracy: 51.517341040462426%
Epoch 1/10, Train Loss: 0.5813, Val Loss: 0.4433, Val Accuracy: 79.91329479768787%
Epoch 2/10, Train Loss: 0.2412, Val Loss: 0.5269, Val Accuracy: 80.92485549132948%
Epoch 3/10, Train Loss: 0.0728, Val Loss: 0.9206, Val Accuracy: 80.27456647398844%
Epoch 4/10, Train Loss: 0.0195, Val Loss: 1.4704, Val Accuracy: 76.80635838150289%
Epoch 5/10, Train Loss: 0.0057, Val Loss: 1.8001, Val Accuracy: 79.26300578034682%
Epoch 6/10, Train Loss: 0.0100, Val Loss: 1.5029, Val Accuracy: 78.46820809248555%
Epoch 7/10, Train Loss: 0.0059, Val Loss: 1.5850, Val Accuracy: 79.91329479768787%
Epoch 8/10, Train Loss: 0.0027, Val Loss: 1.7730, Val Accuracy: 79.11849710982659%
Epoch 9/10, Train Loss: 0.0008, Val Loss: 2.1455, Val Accuracy: 78.97398843930635%
Epoch 10/10, Train Loss: 0.0001, Val Loss: 2.3628, Val Accuracy: 79.1907514450867%
Final Validation Accuracy: 79.1907514450867%, Best Validation Accuracy: 80.92485549132948% after 2 epochs.
```

Metrics while training on dataset 1

```
Initial Validation Accuracy: 51.7125%
Epoch 1/10, Train Loss: 6.2075, Val Loss: 0.6909, Val Accuracy: 65.275%
Epoch 2/10, Train Loss: 0.6484, Val Loss: 0.5436, Val Accuracy: 77.6375%
Epoch 3/10, Train Loss: 0.4251, Val Loss: 0.3073, Val Accuracy: 88.7%
Epoch 4/10, Train Loss: 0.2813, Val Loss: 0.2840, Val Accuracy: 89.4%
Epoch 5/10, Train Loss: 0.2201, Val Loss: 0.2924, Val Accuracy: 89.525%
Epoch 6/10, Train Loss: 0.1655, Val Loss: 0.3245, Val Accuracy: 89.55%
Epoch 7/10, Train Loss: 0.1141, Val Loss: 0.4183, Val Accuracy: 89.1375%
Epoch 8/10, Train Loss: 0.0685, Val Loss: 0.5048, Val Accuracy: 88.8375%
Epoch 9/10, Train Loss: 0.0392, Val Loss: 0.5839, Val Accuracy: 88.5625%
Epoch 10/10, Train Loss: 0.0294, Val Loss: 0.6456, Val Accuracy: 88.61250000000001%
Final Validation Accuracy: 88.61250000000001%, Best Validation Accuracy: 89.55% after 6 epochs.
```

Metrics while training (using best trained model checkpoint) on dataset 2

Case 2: BERT

```
Initial Validation Accuracy: 48.48%
Epoch 1/10, Train Loss: 0.4633, Val Loss: 0.3487, Val Accuracy: 85.77%
Validation loss improved from inf to 0.3487. Saving model...
Epoch 2/10, Train Loss: 0.3689, Val Loss: 0.3147, Val Accuracy: 86.92%
Validation loss improved from 0.3487 to 0.3147. Saving model...
Epoch 3/10, Train Loss: 0.3415, Val Loss: 0.3020, Val Accuracy: 86.92%
Validation loss improved from 0.3147 to 0.3020. Saving model...
Epoch 4/10, Train Loss: 0.3272, Val Loss: 0.3090, Val Accuracy: 87.36%
Epoch 5/10, Train Loss: 0.3236, Val Loss: 0.3011, Val Accuracy: 87.43%
Validation loss improved from 0.3020 to 0.3011. Saving model...
Epoch 6/10, Train Loss: 0.3094, Val Loss: 0.3108, Val Accuracy: 87.14%
Epoch 7/10, Train Loss: 0.2989, Val Loss: 0.3074, Val Accuracy: 86.27%
Epoch 8/10, Train Loss: 0.3012, Val Loss: 0.3162, Val Accuracy: 86.20%
Epoch 9/10, Train Loss: 0.2764, Val Loss: 0.3042, Val Accuracy: 86.92%
Epoch 10/10, Train Loss: 0.2743, Val Loss: 0.3097, Val Accuracy: 87.07%
Best Validation Accuracy: 87.43% at Epoch 5
Final Validation Loss (Best Model): 0.3011
```

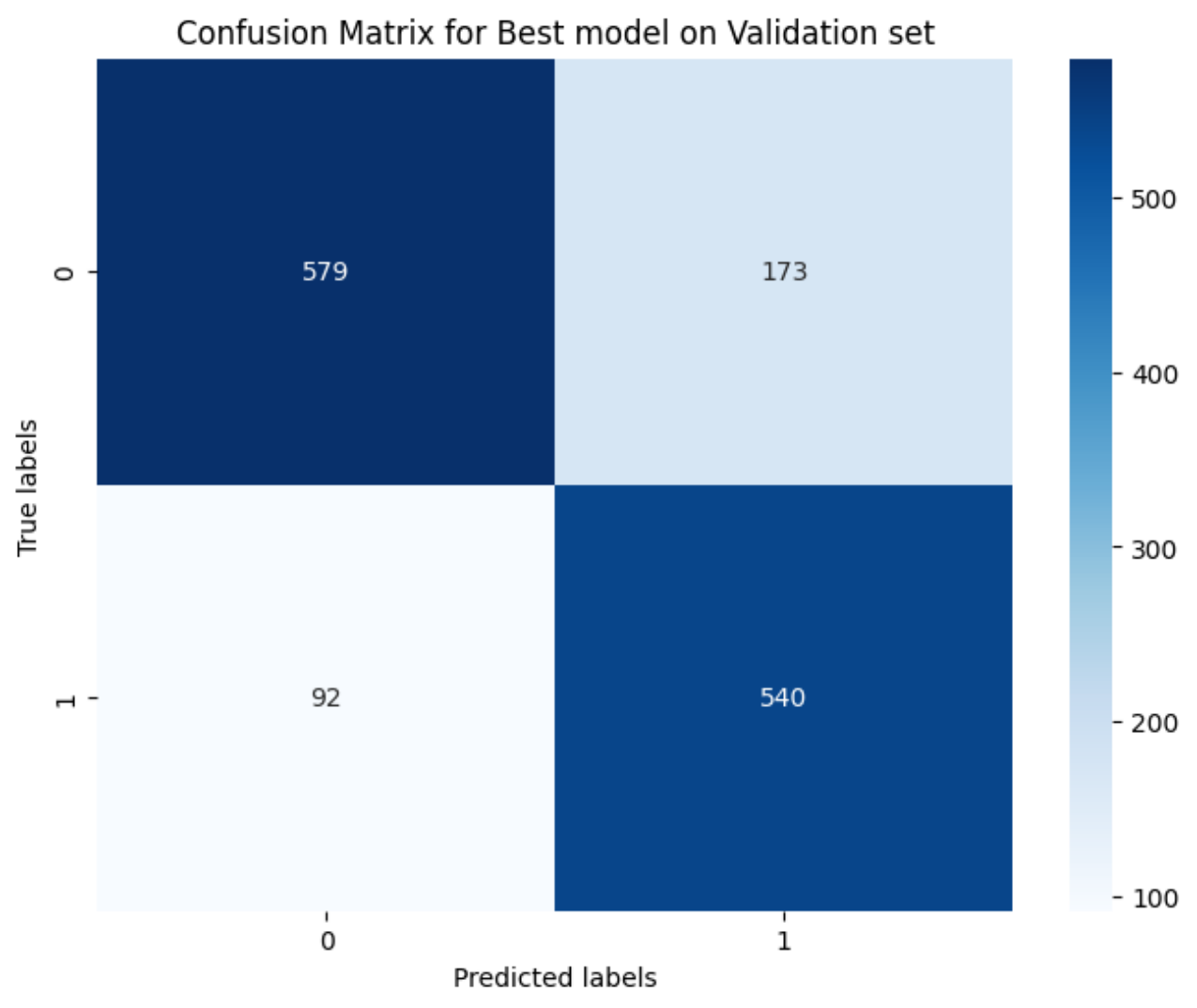
Metrics while training on dataset 1

```
Initial Validation Accuracy: 80.50%
Epoch 1/10, Train Loss: 0.3882, Val Loss: 0.3415, Val Accuracy: 85.52%
Validation loss improved from inf to 0.3415. Saving model...
Epoch 2/10, Train Loss: 0.3536, Val Loss: 0.3255, Val Accuracy: 86.11%
Validation loss improved from 0.3415 to 0.3255. Saving model...
Epoch 3/10, Train Loss: 0.3406, Val Loss: 0.3192, Val Accuracy: 86.41%
Validation loss improved from 0.3255 to 0.3192. Saving model...
Epoch 4/10, Train Loss: 0.3317, Val Loss: 0.3131, Val Accuracy: 86.75%
Validation loss improved from 0.3192 to 0.3131. Saving model...
Epoch 5/10, Train Loss: 0.3245, Val Loss: 0.3090, Val Accuracy: 86.76%
Validation loss improved from 0.3131 to 0.3090. Saving model...
Epoch 6/10, Train Loss: 0.3173, Val Loss: 0.3047, Val Accuracy: 86.67%
Validation loss improved from 0.3090 to 0.3047. Saving model...
Epoch 7/10, Train Loss: 0.3126, Val Loss: 0.3037, Val Accuracy: 86.94%
Validation loss improved from 0.3047 to 0.3037. Saving model...
Epoch 8/10, Train Loss: 0.3070, Val Loss: 0.2999, Val Accuracy: 87.04%
Validation loss improved from 0.3037 to 0.2999. Saving model...
Epoch 9/10, Train Loss: 0.3001, Val Loss: 0.3000, Val Accuracy: 87.35%
Epoch 10/10, Train Loss: 0.2940, Val Loss: 0.2963, Val Accuracy: 87.44%
Validation loss improved from 0.2999 to 0.2963. Saving model...
Best Validation Accuracy: 87.44% at Epoch 10
Final Validation Loss (Best Model): 0.2963
```

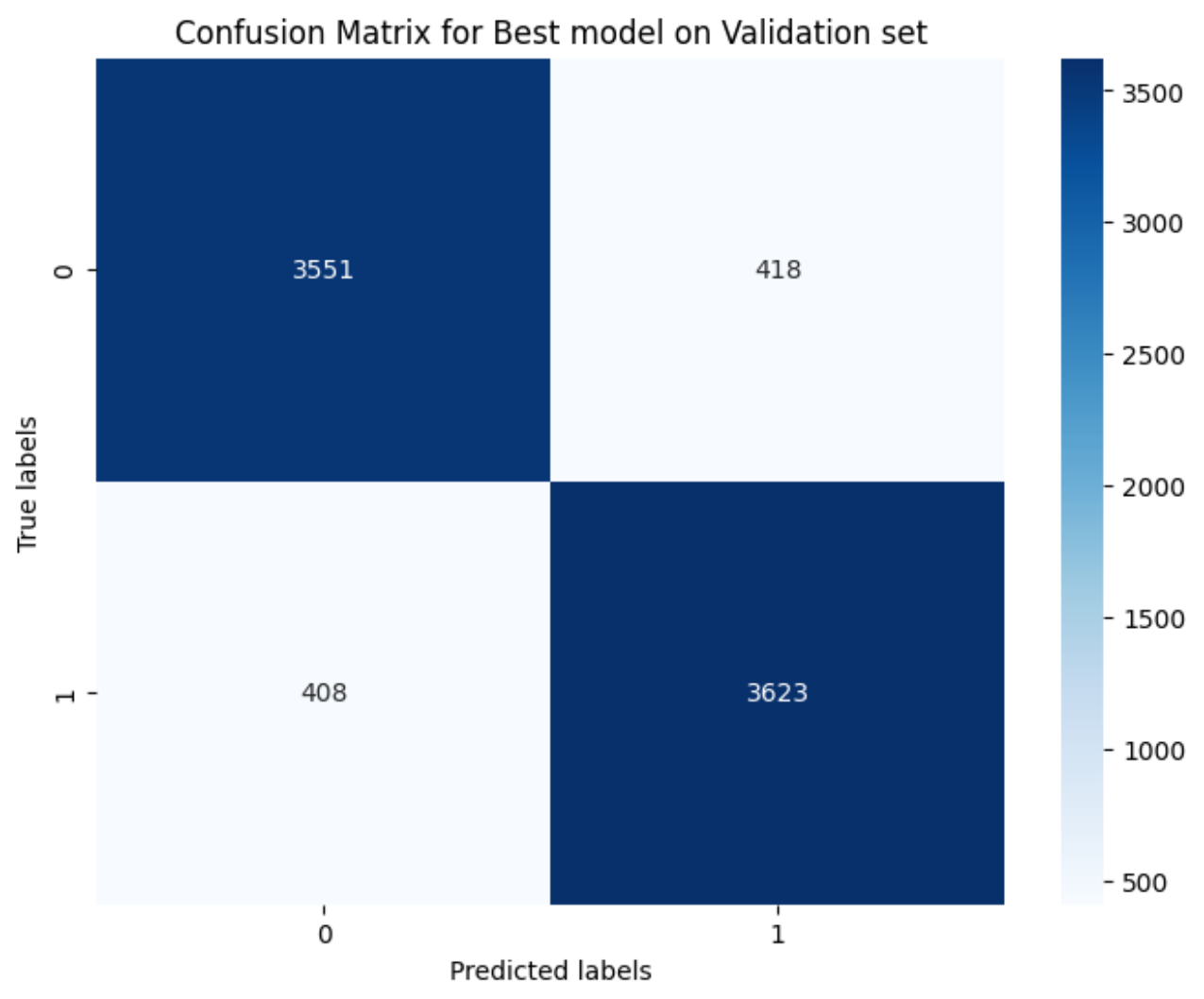
Metrics while training (using best trained model checkpoint) on dataset 2

Confusion matric visualisations

Case 1: BoW

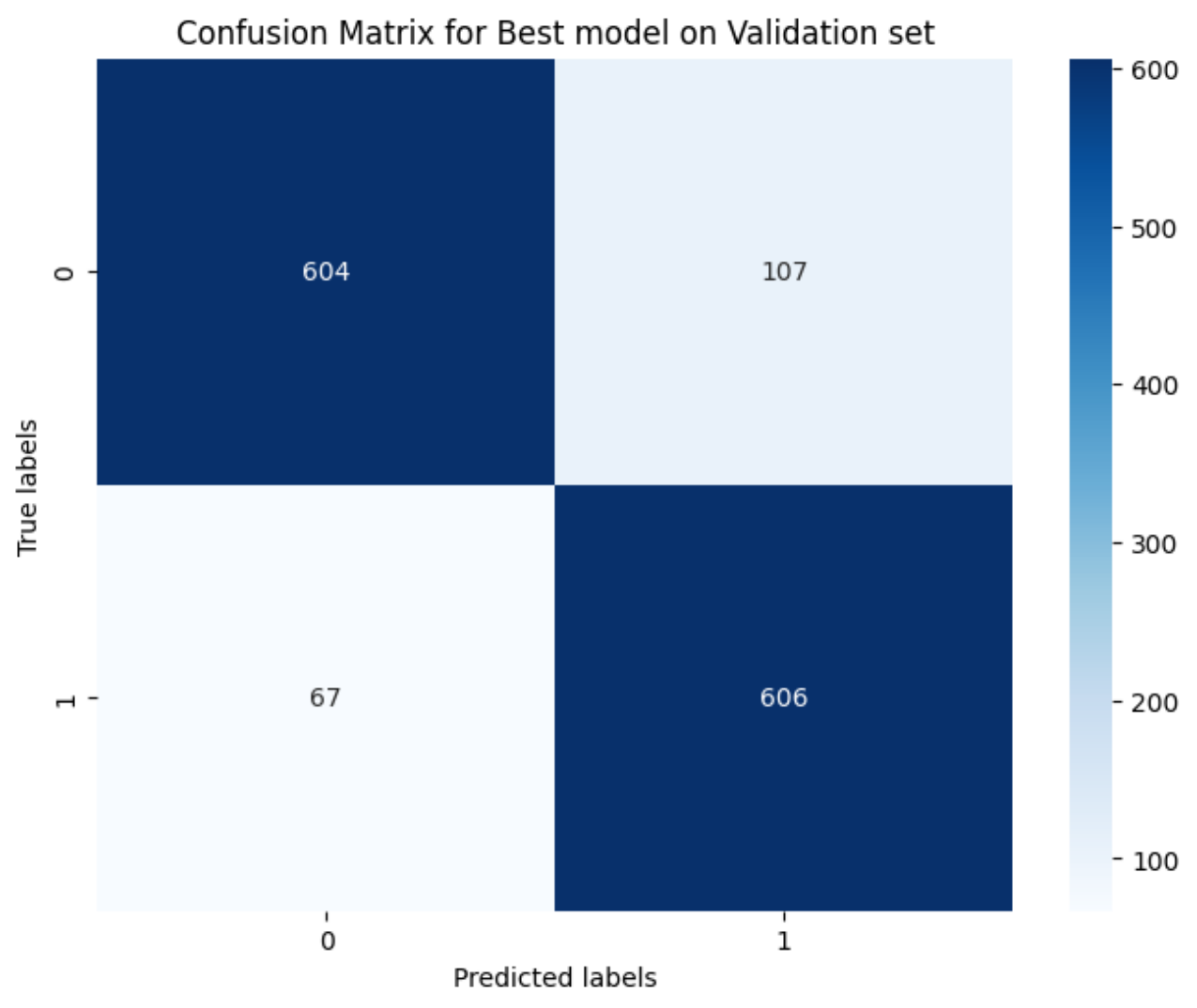


Confusion matrix for best model on dataset 1 Validation set

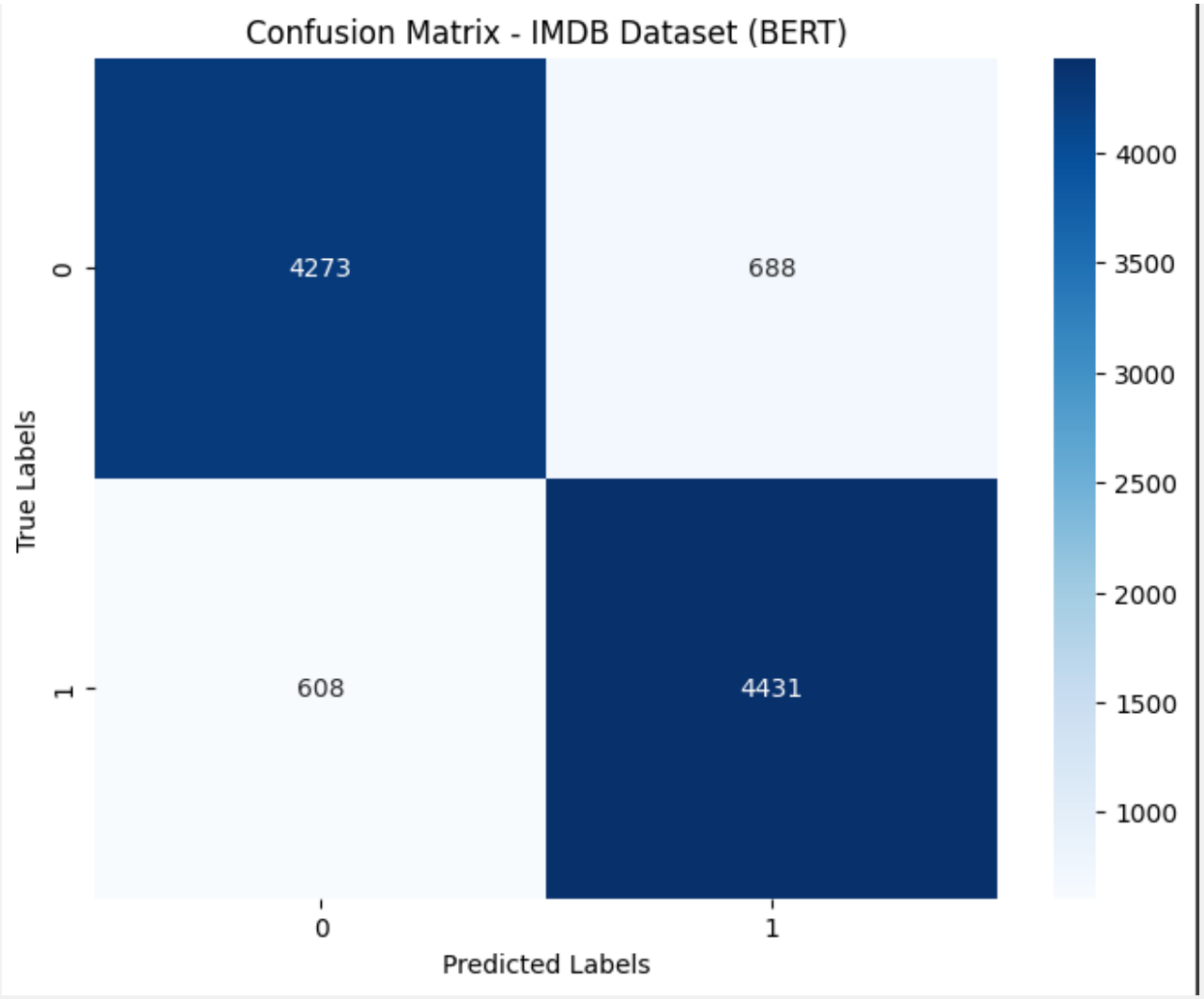


Confusion matrix for best model (from pretrained best checkpoint) on dataset 2 Validation set

Case 2: BERT



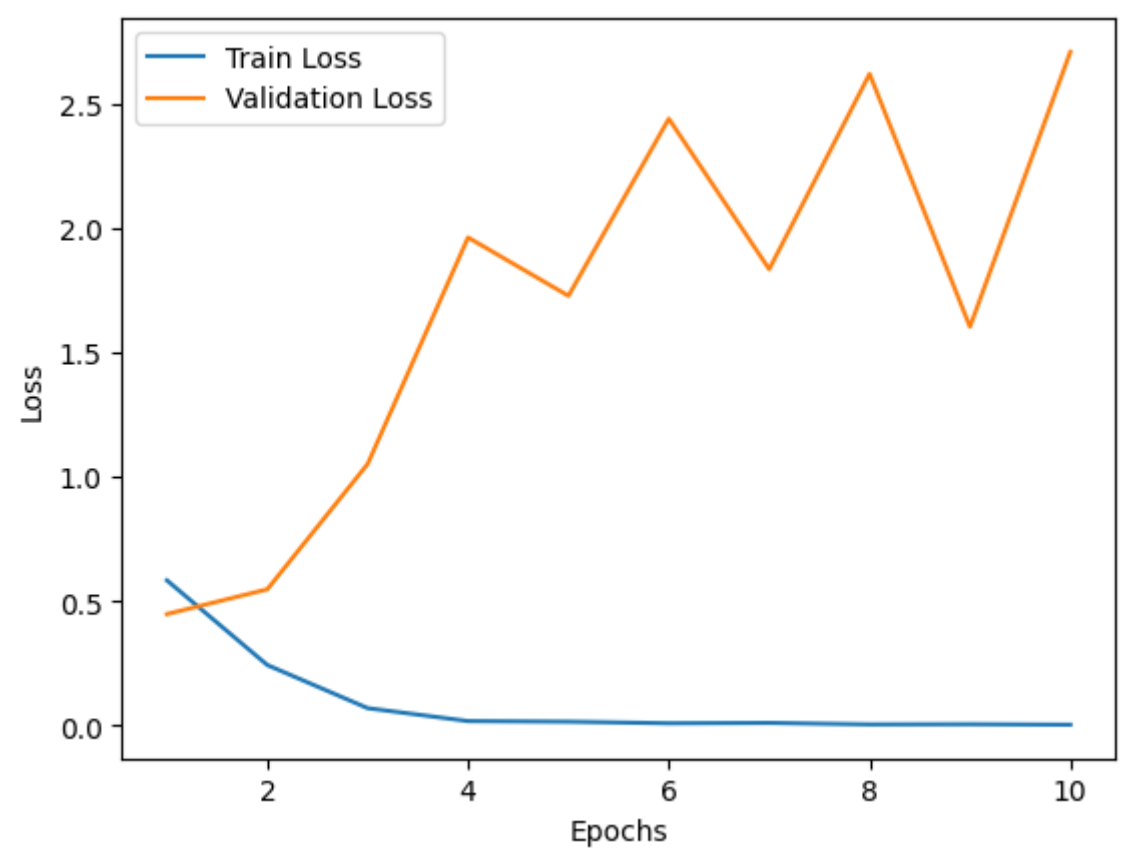
Confusion matrix for best model on dataset 1 Validation set



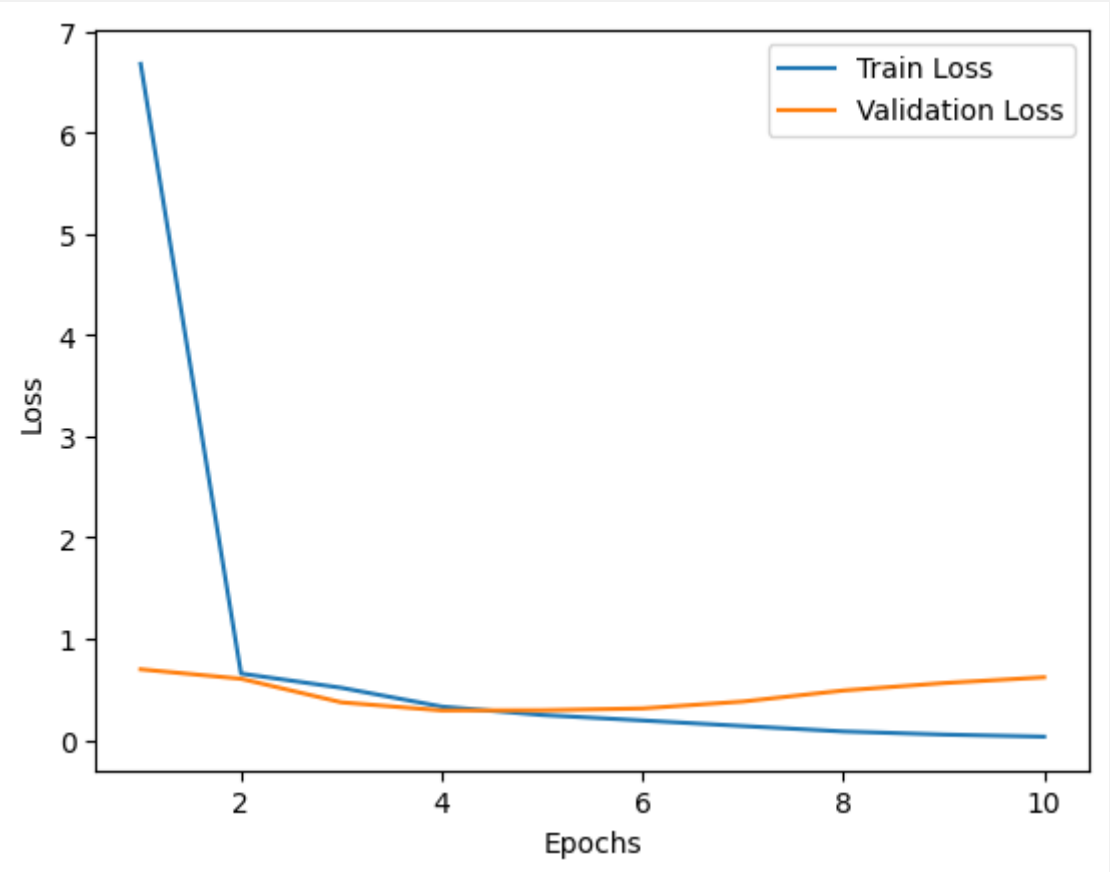
Confusion matrix for best model (from pretrained best checkpoint) on dataset 2 Validation set

Training and validation loss curves.

Case 1: BoW

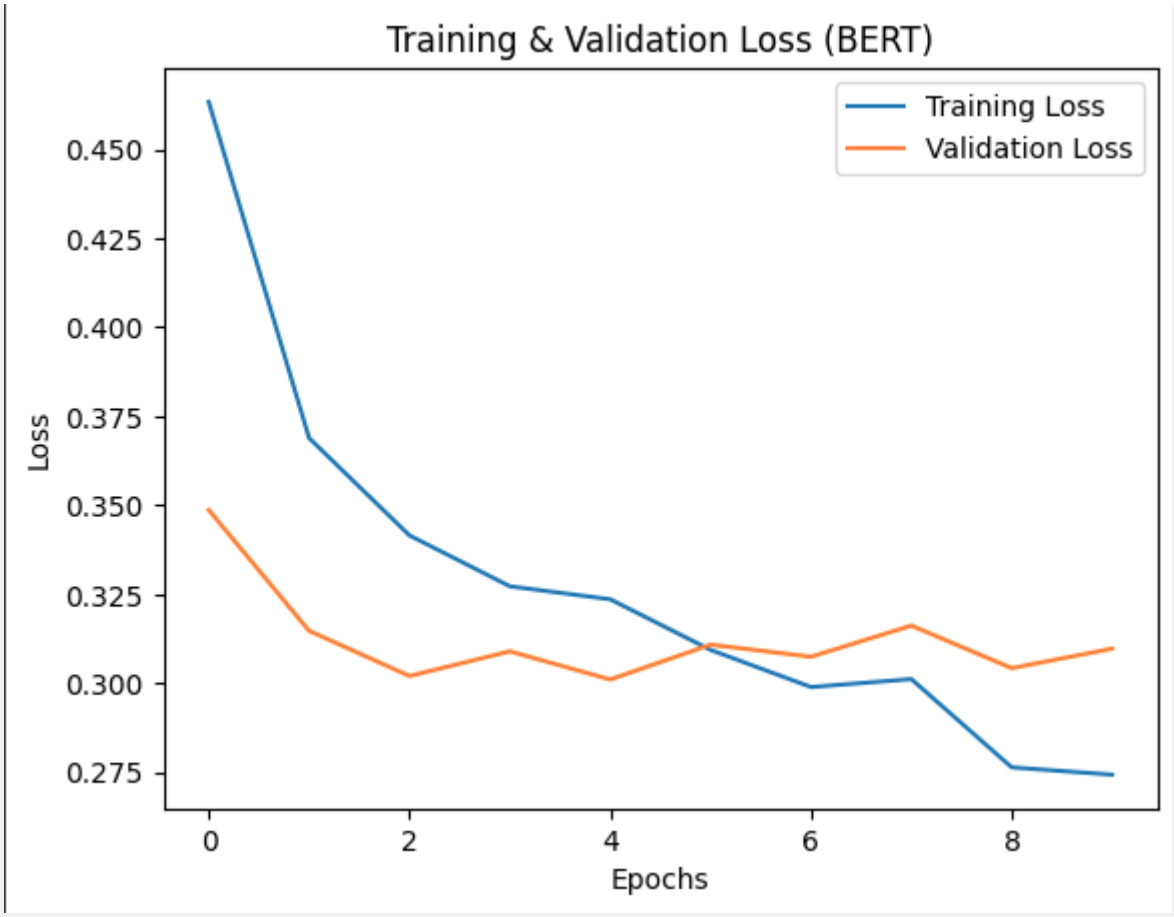


Curves for training on Dataset 1

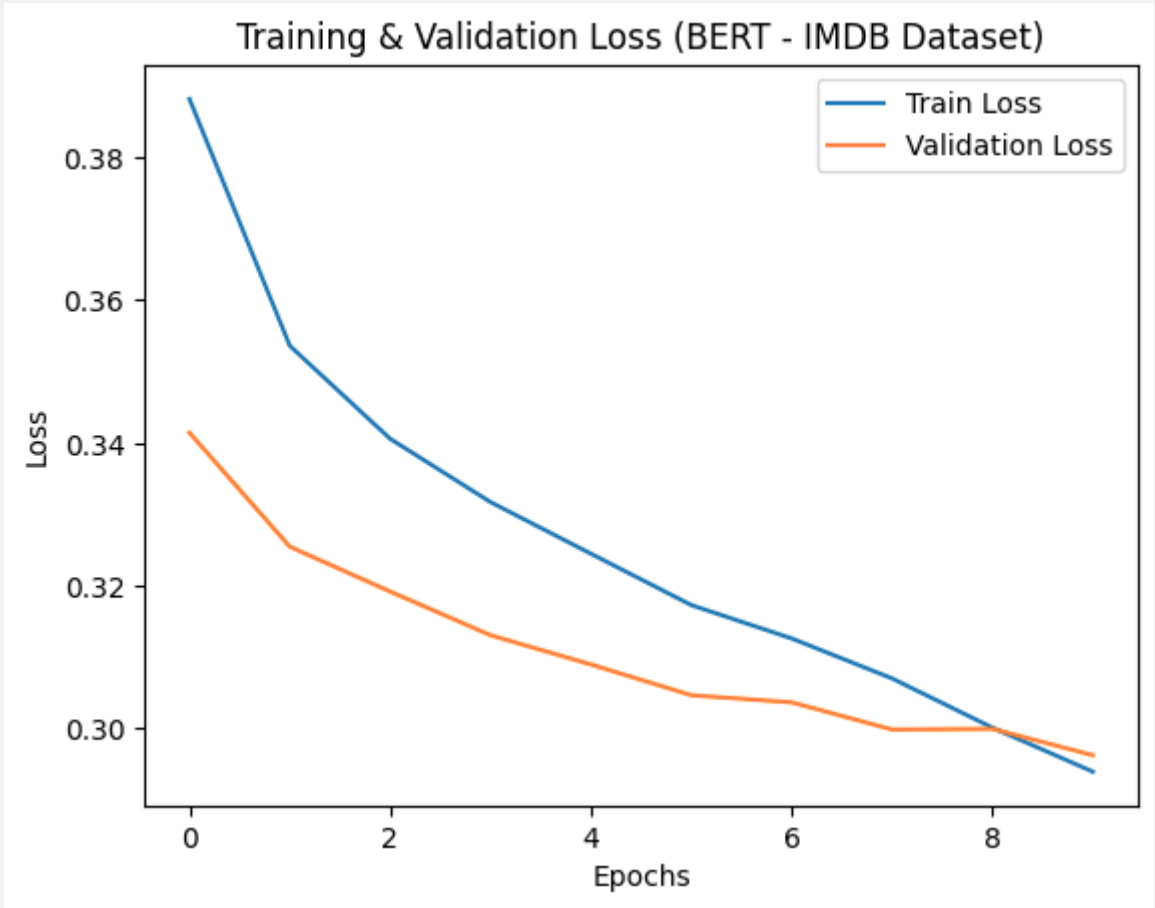


Curves for ReSuMe training on Dataset 2 (IMDB)

Case 2: BERT

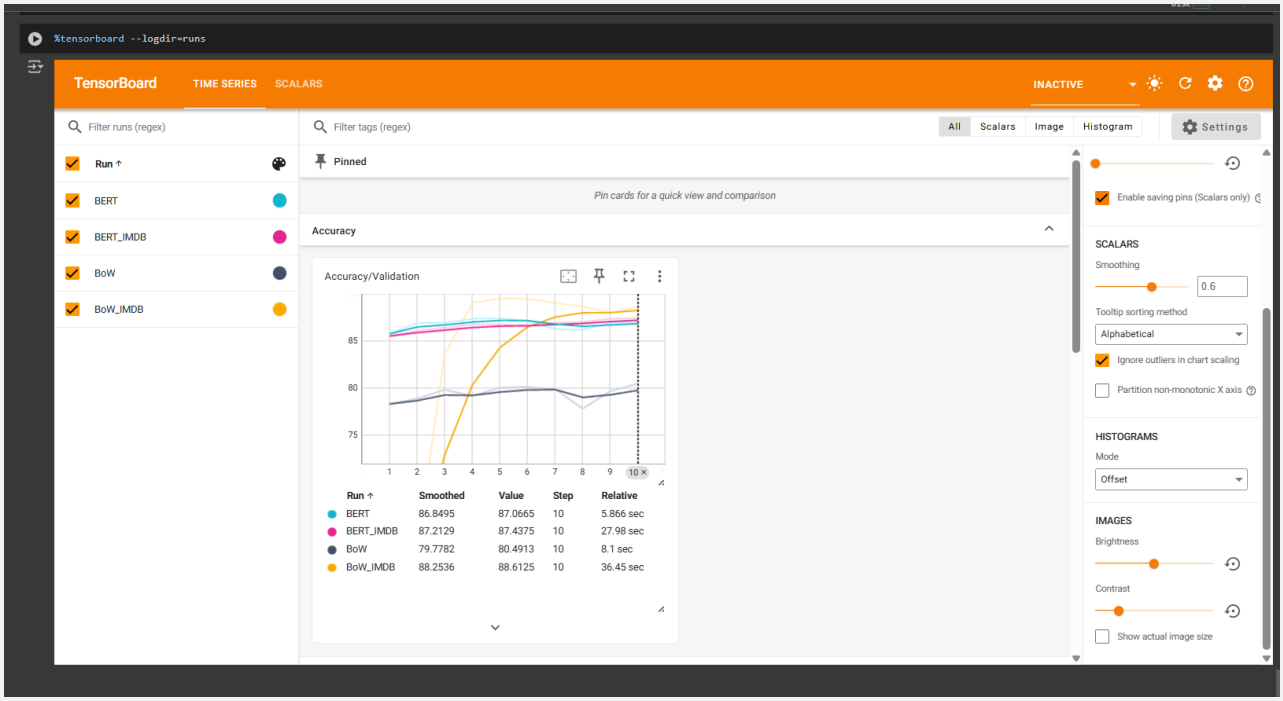


Curves for training on Dataset 1

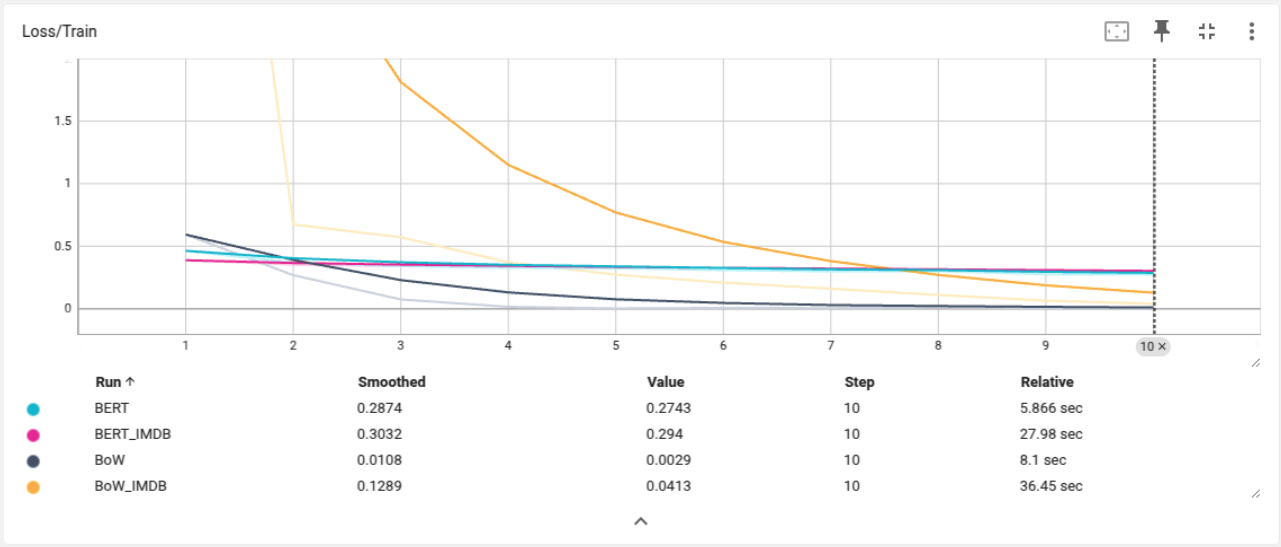


Curves for ReSuMe training on Dataset 2 (IMDB)

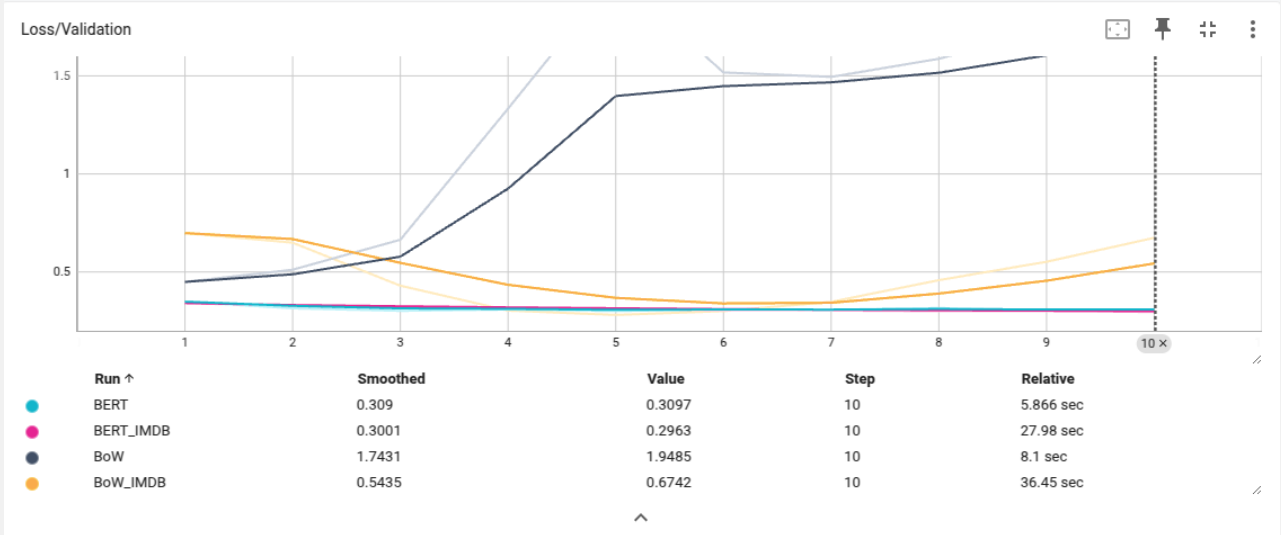
Tensorboard Integration (Final Evaluation)



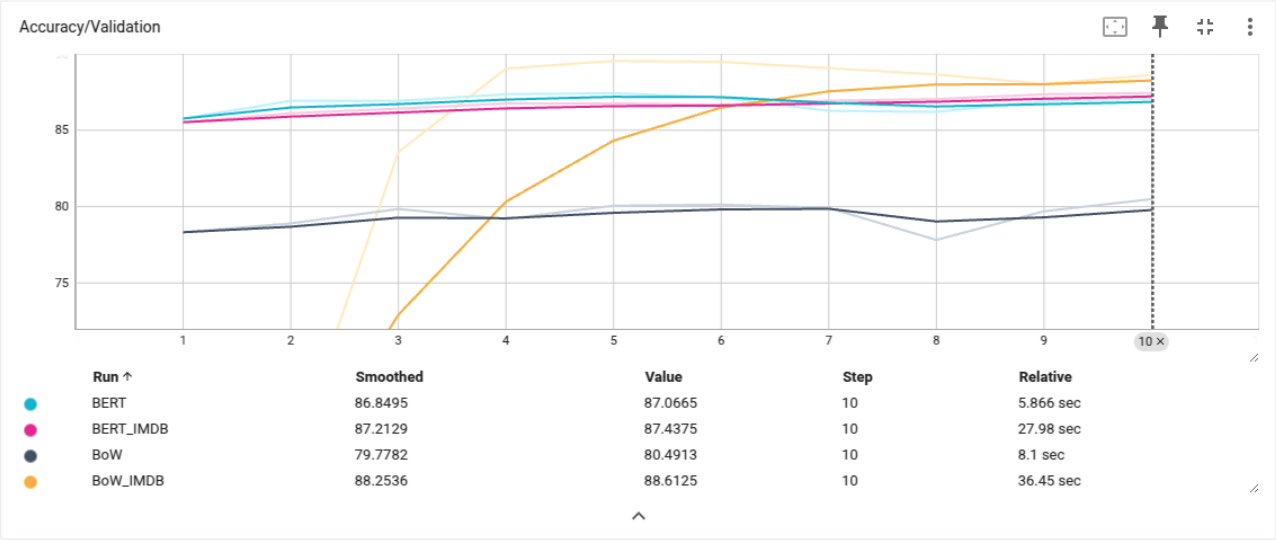
You can see the tensorboard integration of the models metrics on colab



Train Loss plot and comparison



Validation Loss plot and comparison

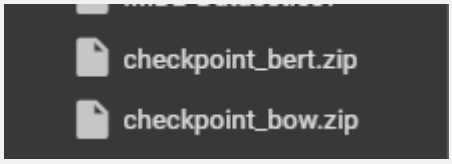


Validition Accuracy plot and comparison

Final Validation Accuracies

Method	Dataset 1	Dataset 2
BoW	~80%	~89%
BERT Embeddings	~87%	~87.5%

Checkpoint Compression



Here you can see we are saving the checkpoint as a compressed zip file, and then reading from it as well.