**FinBERT Fine-tuning Project Documentation**

**Project Overview**

This project implements fine-tuning of the FinBERT model for financial sentiment analysis using k-fold cross-validation and LoRA (Low-Rank Adaptation). The model is optimized for MacOS with MPS (Metal Performance Shaders) support.

**Technical Specifications**

**Environment Setup**

* Python 3.11
* PyTorch with MPS support
* Transformers 4.41.2
* PEFT 0.11.1
* Running on MacOS with Metal GPU acceleration

**Base Model**

* Model: ProsusAI/finbert
* Architecture: BERT-based
* Task: Financial Sentiment Classification
* Classes: 3 (Negative, Neutral, Positive)

**Dataset Structure**

Columns: ['Date', 'text', 'label']

Label Distribution:

- Positive (1): 94 samples

- Neutral (0): 81 samples

- Negative (-1): 77 samples

Total Samples: 252

**Implementation Details**

**Data Preprocessing**

1. Label Mapping:
   * -1 → 0 (Negative)
   * 0 → 1 (Neutral)
   * 1 → 2 (Positive)
2. Tokenization Parameters:
   * Max Length: 512 tokens
   * Padding: max\_length
   * Truncation: True

**Training Configuration**

training\_args = TrainingArguments(

    per\_device\_train\_batch\_size=8,

    per\_device\_eval\_batch\_size=4,

    num\_train\_epochs=10,

    learning\_rate=5e-5,

    warmup\_steps=500,

    weight\_decay=0.1,

    evaluation\_strategy="steps",

    eval\_steps=100

)

**Model Adaptation (LoRA)**

lora\_config = LoraConfig(

    r=16,               # Rank

    lora\_alpha=32,      # Alpha scaling

    target\_modules=["query", "value", "key", "dense"],

    lora\_dropout=0.1,

    bias="none",

    task\_type=TaskType.SEQ\_CLS

)

**Cross-Validation Setup**

* K-folds: 5
* Stratification: Yes
* Random Seed: 42
* Class Weight: Balanced

**Evaluation Metrics**

1. Per-fold Metrics:
   * Accuracy
   * Macro F1-score
   * Confusion Matrix
   * Detailed Classification Report
2. Aggregate Metrics:
   * Average Training Accuracy
   * Average Training F1
   * Average Validation Accuracy
   * Average Validation F1

**Model Export**

* Format: TorchScript
* Saved Components:
  + Best model weights
  + Tokenizer
  + Training metrics
  + Configuration files

**Usage Instructions**

**Running Training**

python finbert\_finetune\_refactored.py

**Model Inference**

def predict\_sentiment(text, model, tokenizer, device):

    inputs = tokenizer(

        text,

        return\_tensors="pt",

        padding=True,

        truncation=True,

        max\_length=512

    ).to(device)

    with torch.no\_grad():

        outputs = model(\*\*inputs)

        probs = torch.nn.functional.softmax(outputs.logits, dim=-1)

    predicted\_class = torch.argmax(probs, dim=1)[0].item()

    return {

        "sentiment": ["negative", "neutral", "positive"][predicted\_class],

        "probabilities": probs[0].cpu().numpy()

    }

**Performance Monitoring**

* Training progress tracked via Weights & Biases
* Metrics logged every 20 steps
* Model checkpoints saved every 100 steps
* Best model selected based on validation F1 score

**Error Handling**

* Comprehensive exception handling for:
  + Data loading
  + Model training
  + Evaluation
  + Model export
* Detailed error messages and debugging information

**Future Improvements**

1. Implement gradient accumulation for larger batch sizes
2. Add early stopping based on validation metrics
3. Experiment with different LoRA configurations
4. Add support for mixed precision training
5. Implement cross-validation result visualization

*Note: Update this documentation with actual performance metrics after completing a full training run.*

Training parameters

per\_device\_train\_batch\_size=4,

per\_device\_eval\_batch\_size=2,

num\_train\_epochs=5,

learning\_rate=5e-5,

warmup\_steps=500,

weight\_decay=0.1,

logging\_steps=500,

eval\_steps=1500,

save\_steps=1500,

evaluation\_strategy="steps",

save\_strategy="steps",

load\_best\_model\_at\_end=True,

metric\_for\_best\_model="f1",

Fold 1 results as follows

=== Preparing Fold 1/3 ===

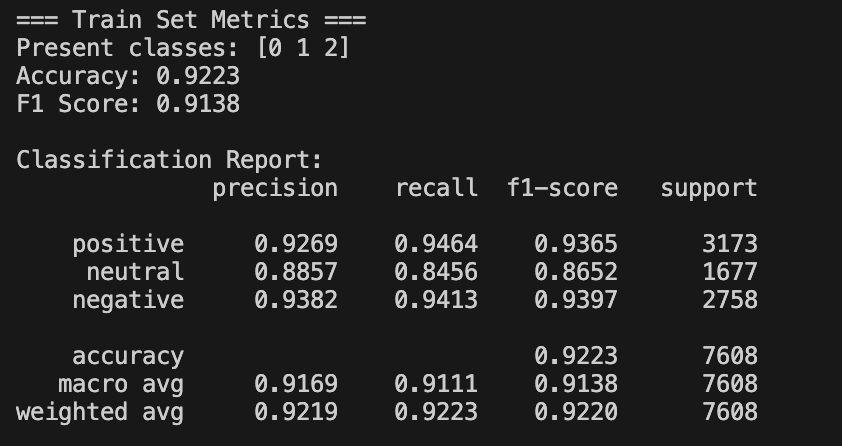
Train label counts for fold 1: [3173 1677 2758]

Validation label counts for fold 1: [1586 838 1380]

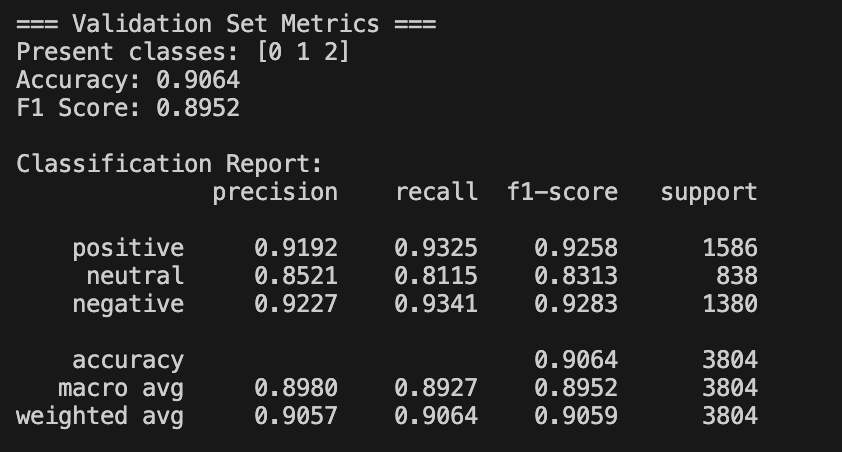
Dataset keys: ['text', 'label']

Number of examples: 7608

Training Metrics



Validation Metrics



Fold 2 results as follows

Train label counts for fold 2: [3172 1677 2759]

Validation label counts for fold 2: [1587 838 1379]

Dataset keys: ['text', 'label']

Number of examples: 7608

A graph with blue squares and numbers

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A blue squares with numbers and a blue box

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Fold 3 details as follows

=== Preparing Fold 3/3 ===

Train label counts for fold 3: [3173 1676 2759]

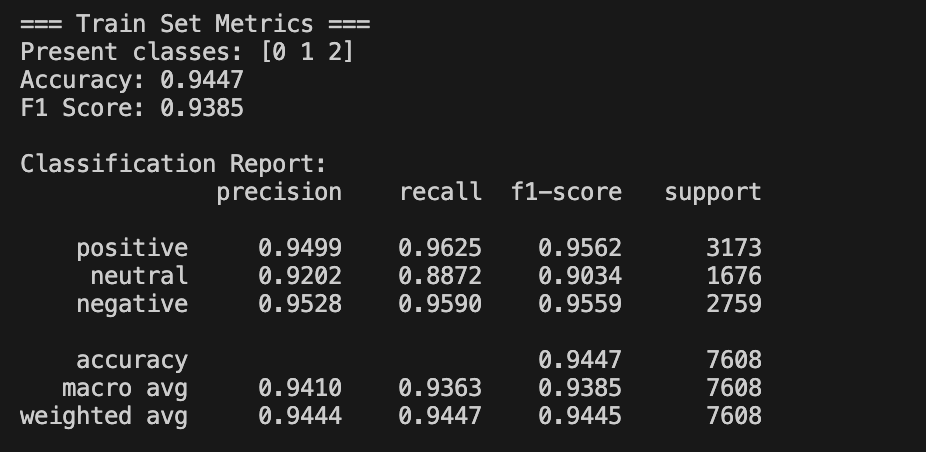
Validation label counts for fold 3: [1586 839 1379]

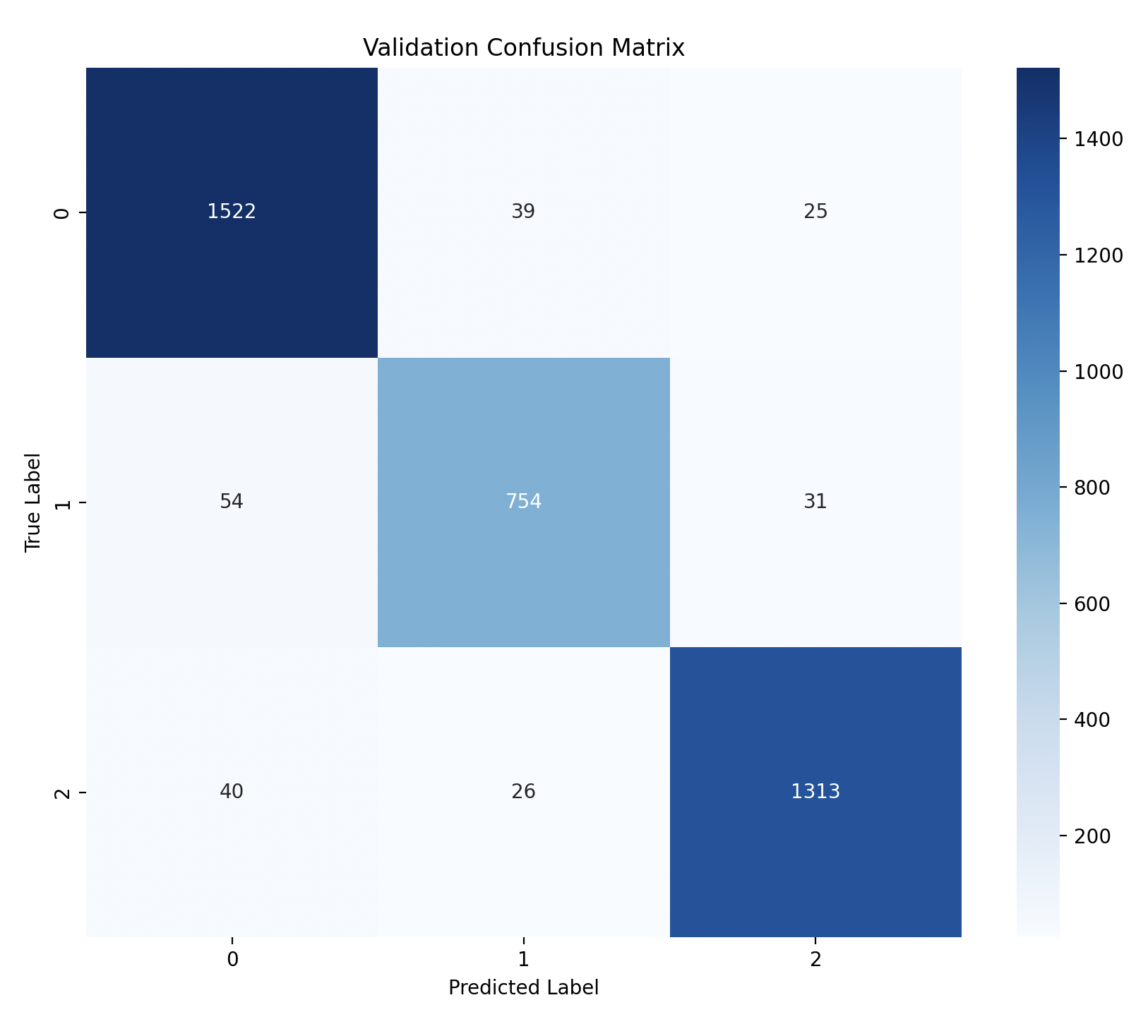
Dataset keys: ['text', 'label']

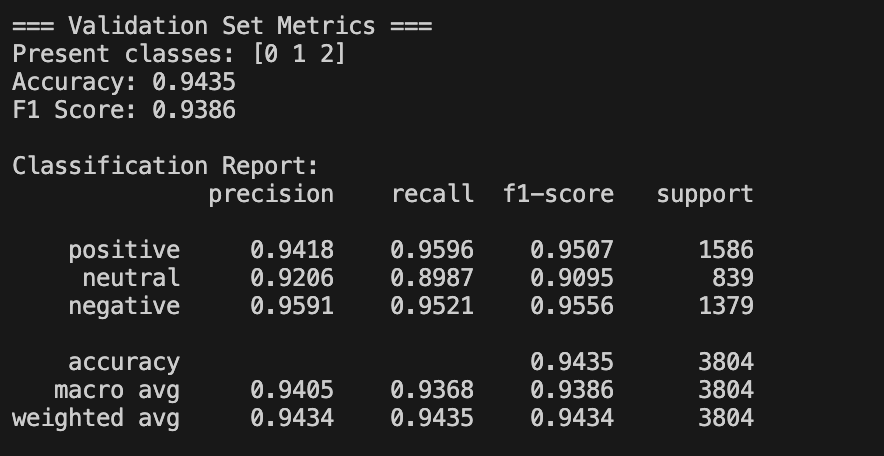
Number of examples: 7608

A graph of a graph with numbers and a blue square

AI-generated content may be incorrect.







A black screen with white text

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.