J.B. SPEED SCHOOL OF ENGINEERING

Course Project:
Fine-Tuning a Quantized LLM for Sentiment Signal Generation in FOREX News

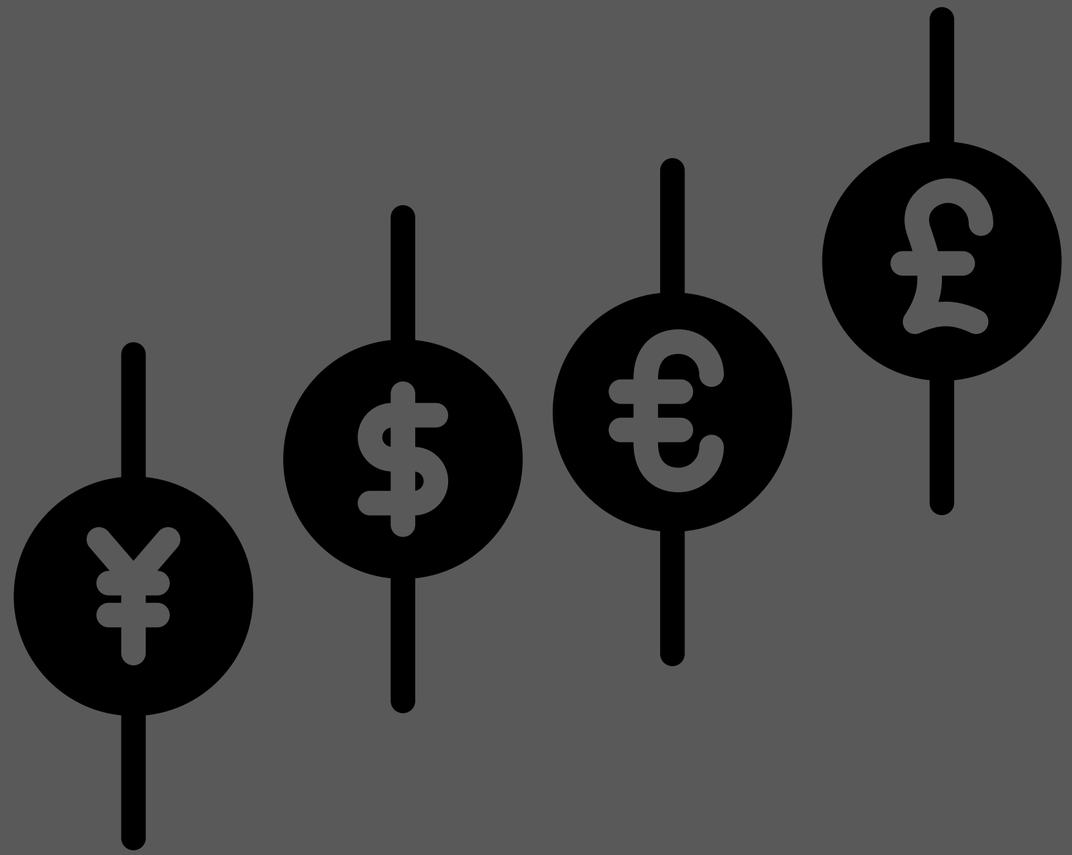
By Matthew Poncini

20 July 2025

CSE 590 GenAI

Training GenAI-Based Sentiment Signal Model with FOREX News and Historical Exchange Rate Data Overview

- Introduction
  - Background
  - Purpose
  - Plan
- Methodology
  - Parameters
  - Data Extracting
  - Pipeline
- Results
- Discussion Points
- Further Work



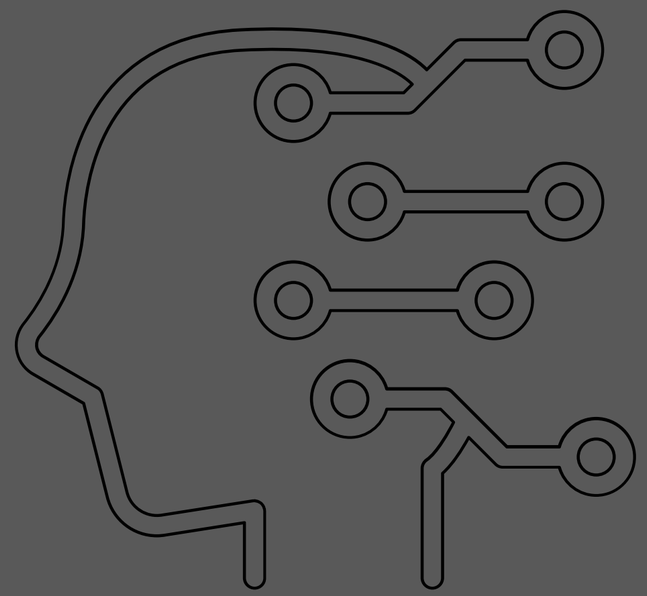
J.B. SPEED SCHOOL OF ENGINEERING

Background: What is FOREX Trading?

- FOREX (Foreign Exchange) trading is the buying and selling of currency pairs on the global market.
- Traders speculate on currency price movements to earn profit
- The market is highly liquid, with daily volumes exceeding $7 trillion
- Prices are influenced by economic indicators, interest rates, and geopolitical events.

Purpose: How can GenAI help traders succeed in the FOREX Market?

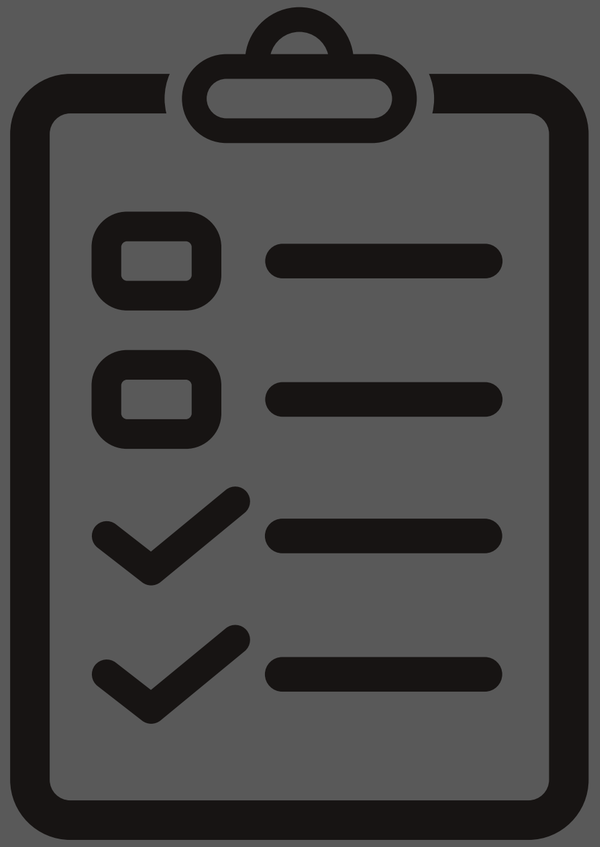
- GenAI can enhance FOREX trading by analyzing vast volumes of real-time news, economic data, and social media to detect market-moving trends.
- Generate trading signals, summarize macroeconomic reports, and assist in sentiment analysis



J.B. SPEED SCHOOL OF ENGINEERING

Plan: Train an LLM to Generate a Signal based on News Sentiment

- Gather News Data
- Gather Historical FOREX Price Data
- Train an LLM based off the News Data and Price Data
- Test the Trained LLM’s Performance



J.B. SPEED SCHOOL OF ENGINEERING

Exchange Rate Parameters

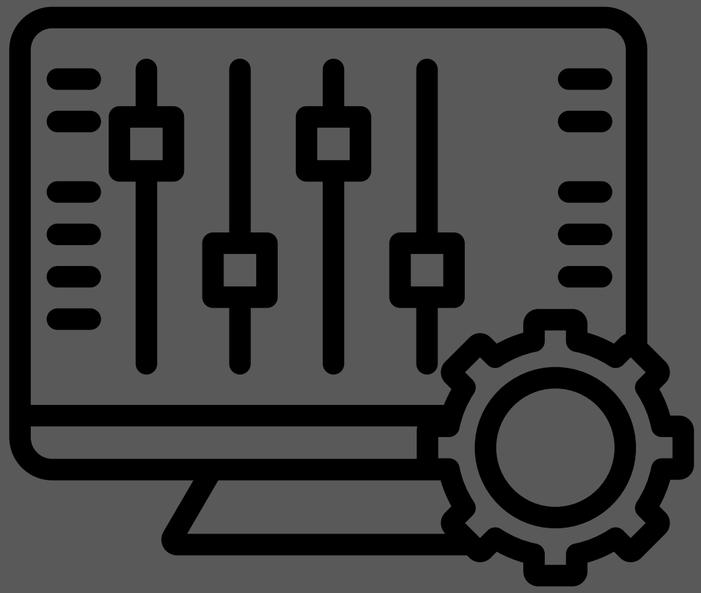
- Three Exchanges will be examined:
  - EURUSD - Euro to US Dollar
  - EURJPY - Euro to Japanese Yen
  - USDJPY - US Dollar to Japanese Yen

- Exchange Prices that increased over 0.5% over the next 2-3 are labeled as “Up”

- Exchange Prices that decreased over 0.5% over the next 2-3 are labeled as “Down”

- All other Exchange Prices are labeled as “Neutral”

- The LLM was only trained using the FOREX Price Change and the title of a FOREX News Article



J.B. SPEED SCHOOL OF ENGINEERING

Where the Data Came From

FOREX PRICES

- Imported from Yahoo Finance
- import yfinance as yf

FOREX NEWS ARTICLES

- Used webscraping techniques from SerPapi website

```python
import requests
import pandas as pd

API_KEY = " # Replace with your real key"
all_results = []  # Global list to hold all results

def get_forex_articles(start_date, end_date, currency1, currency2, num_results=100, api_key=None):
    global all_results

    if api_key is None:
        raise ValueError("API key is required")

    query = f'site:forexlive.com "{currency1}/{currency2}"'
    tbs_range = f"cdr:1,cd_min:{start_date},cd_max:{end_date}"

    params = {
        "engine": "google",
        "q": query,
        "num": num_results,
        "api_key": api_key,
        "hl": "en",
        "filter": 1,
        "tbs": tbs_range
    }

    response = requests.get("https://serpapi.com/search", params=params)
```

J.B. SPEED SCHOOL OF ENGINEERING

Data Cleaning and Processing

Duplicates were removed

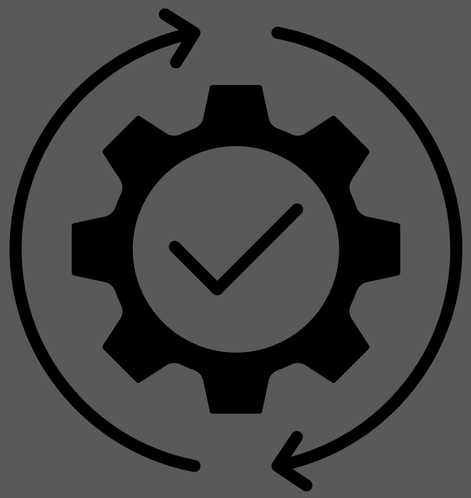
Titles with “Video“ in the Title were removed

Articles were labeled for each exchange (EURUSD, EURJPY, USDJPY) as “Up”, “Down” or “Neutral” depending on how the price changed over the next few days after the article publish date

title	parsed_date	EURUSD_2d_change_pct	EURJPY_2d_change_pct	USDJPY_2d_change_pct	eurusd_label	eurjpy_label	usdjpy_label
China to suspend capital raisings by property firms	Apr 27, 2010	0.7003	1.636	0.933	up	up	up
Fed's Hoenig:To Face Pressure to Monetize Debt W/O Fisc ...	May 6, 2010	1.1578	3.8095	2.5872	up	up	up
Japan Oct Foreign Reserves Hit New Record High of ...	Nov 7, 2010	-2.5187	-2.2011	0.3887	down	down	neutral
EUR/USD ticks higher as Europe settles in	Mar 7, 2011	-0.6629	-0.1302	0.555	down	neutral	up
EUR/USD session outlook	Mar 10, 2011	0.2327	-1.6439	-1.87	neutral	down	down
Japan FinMin watching markets closely	Mar 16, 2011	0.334	-1.6151	-1.9197	neutral	down	down

1,160 articles were left after data cleaning

Testing the Base Model



- Model: LLaMA-2-7B with 4-bit quantization
- 260 news articles were used in the test sample

Prompt

```
def make_prompt_for_inference(title, pair):
    return f"""### Instruction:
Predict the {pair.upper()} movement direction based on the headline. Only respond with one of the following: up, down, or neutral.

### Input:
{title}

### Response:"""



pairs = ["EURUSD", "EURJPY", "USDJPY"]
```

Fine-Tuning with with QLoRA

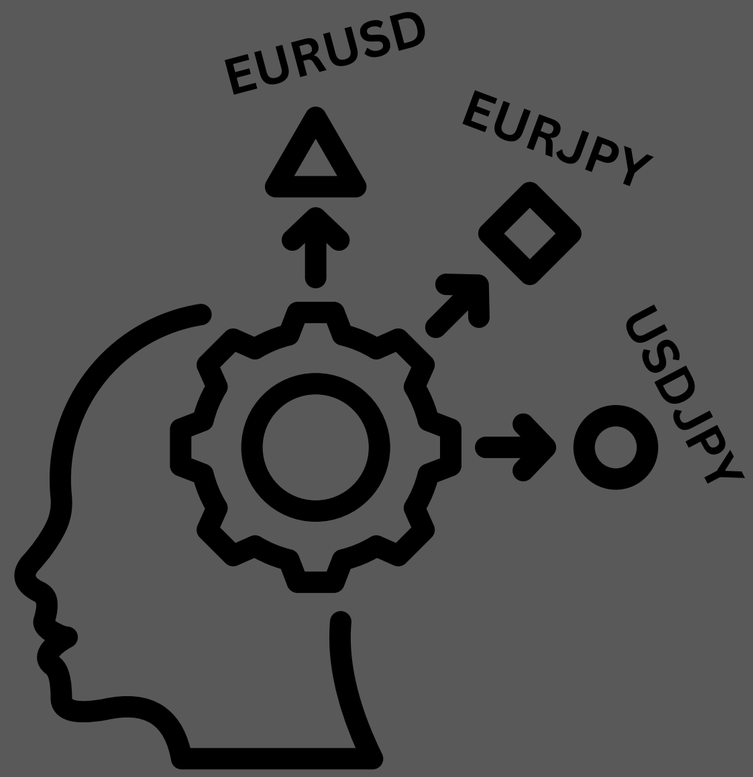
Base Model: LLaMA-2-7B with 4-bit quantization

Libraries: transformers, peft, trl, bitsandbytes

900 examples in the training sample

Parameter-Efficient Tuning: Only the q_proj and v_proj attention layers were trained

Three separate LoRA adapters trained, one for each exchange



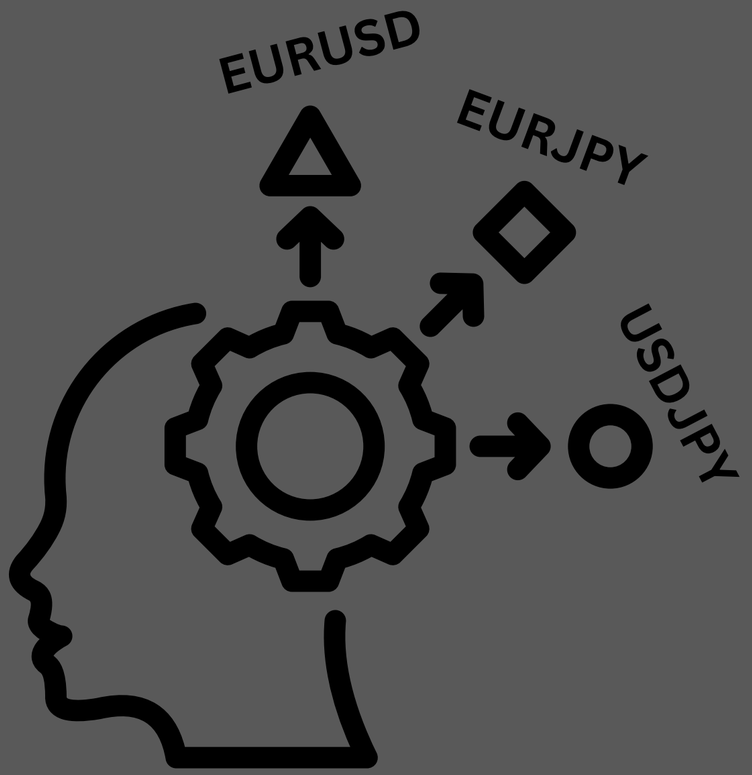
J.B. SPEED SCHOOL OF ENGINEERING

Testing the Fine-Tuned LLM

Test Set: Same 260 news article titles from the BaseModel, unseen during training

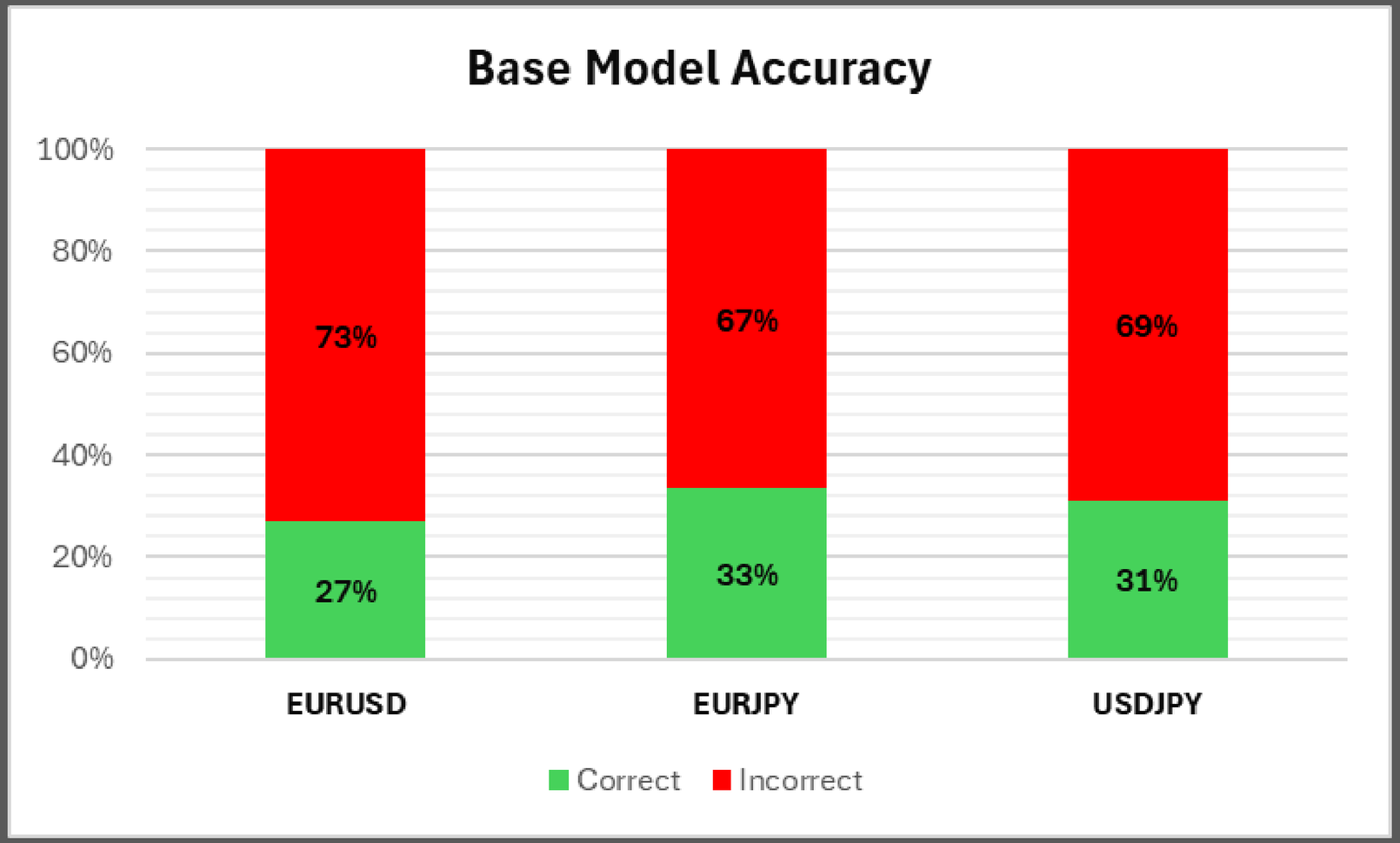
Load one adapter at a time (EURUSD, EURJPY, or USDJPY)

Test for sentiment on one exchange at a time



Results: LLM Performance without Training

Test Data included 260 Samples



Correctly or Incorrectly predicted a Exchange Price went Up, Down, or stayed the same given a news article title

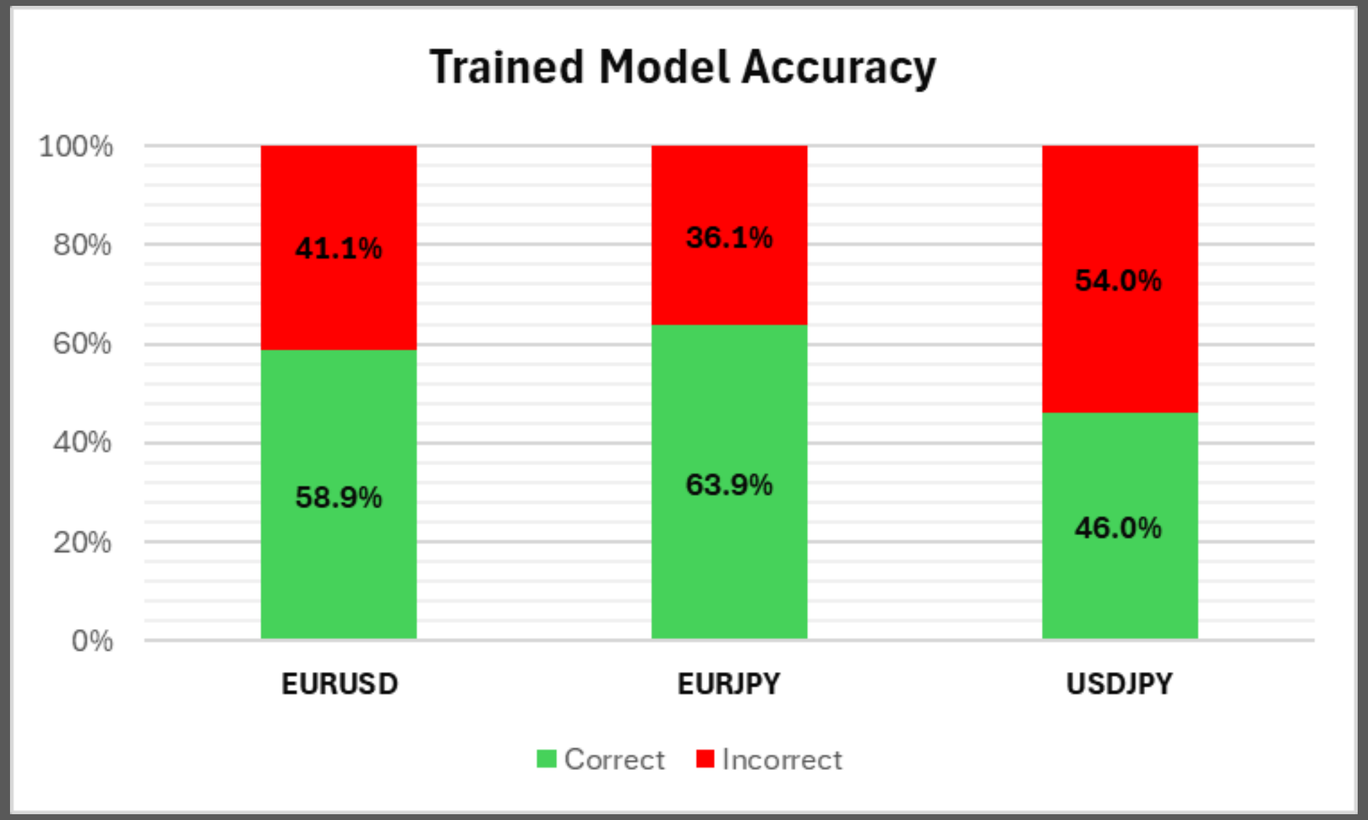
J.B. SPEED SCHOOL OF ENGINEERING

Results: LLM Performance Trained on FOREX News and Price Data

Model:
meta-llama/Llama-2-7b-chat-hf

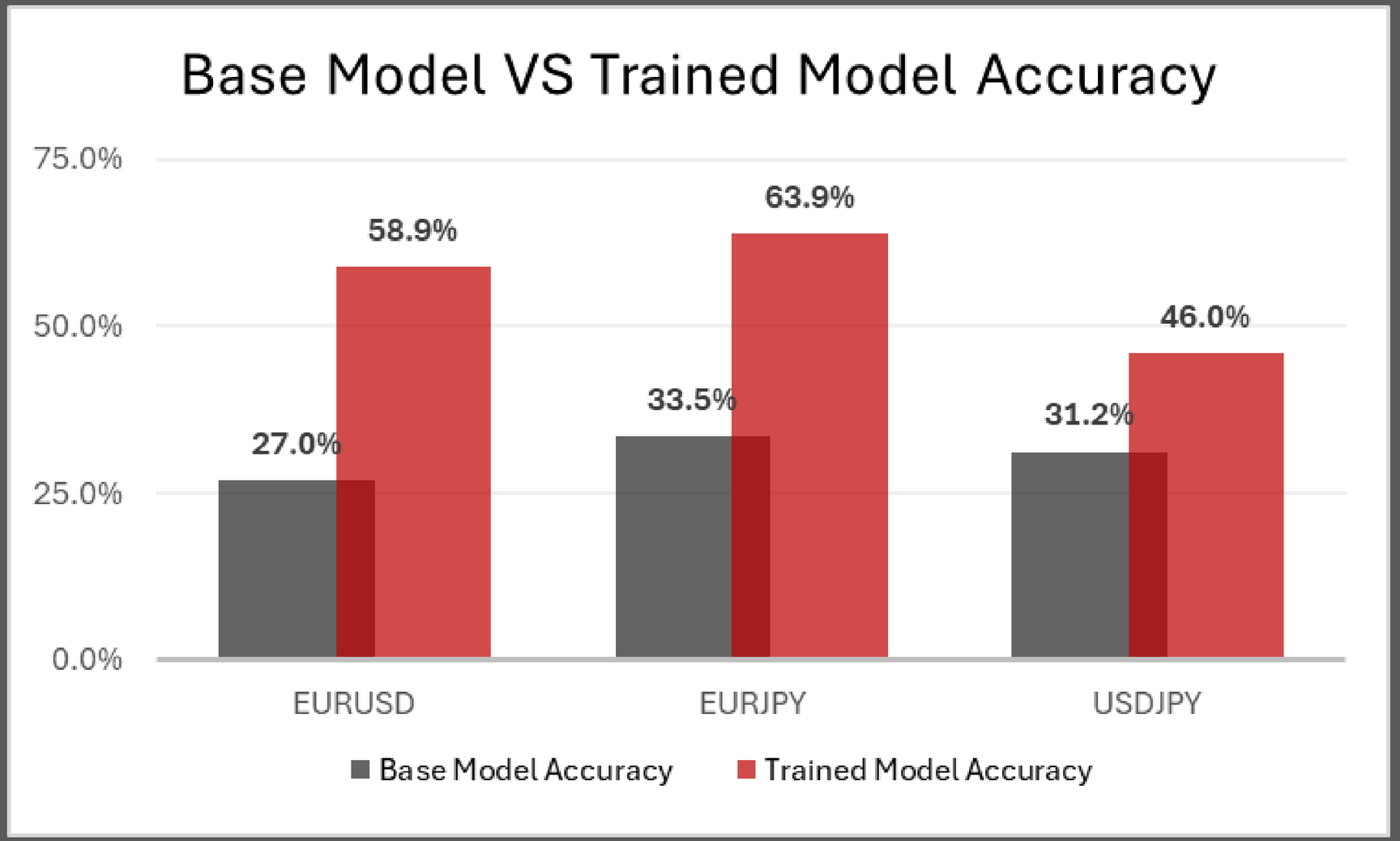
Tested on the same 260 Samples

Correctly or Incorrectly predicted a Exchange Price went Up, Down, or stayed the same given a news article title



J.B. SPEED SCHOOL OF ENGINEERING

Results Summary: The Trained LLM Clearly Outperformed the Base Model for each Currency



J.B. SPEED SCHOOL OF ENGINEERING

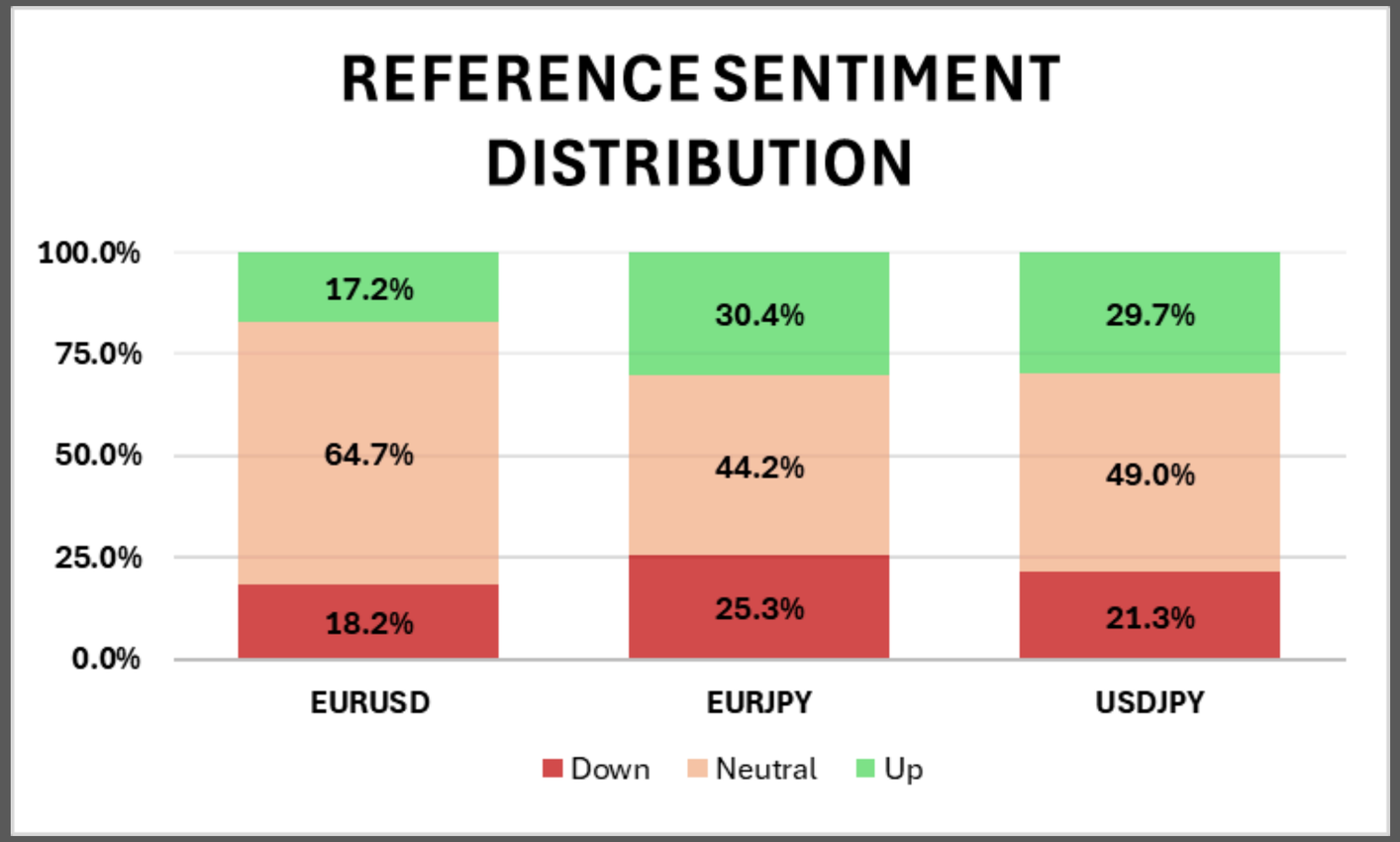
The Trained LLM Clearly Outperformed the Base Model for each Currency

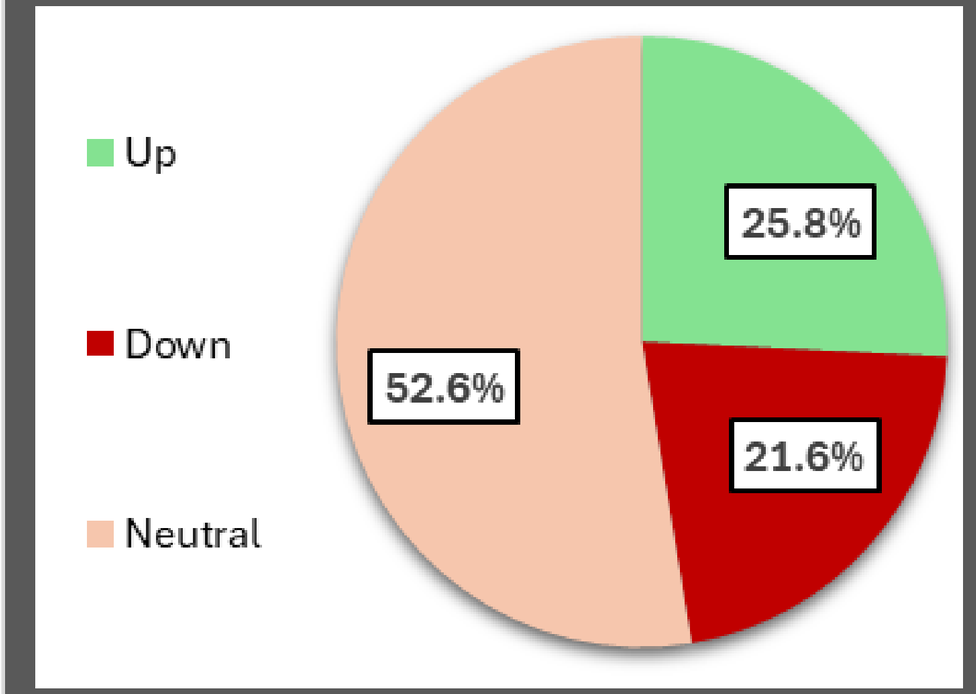
However …

parsed_date	title	predicted_label	currency_pair
Dec 31, 2024	USD KZT | US Dollar Kazakh Tenge	neutral	eurusd
Nov 6, 2024	USD/JPY tumbling	neutral	usdjpy
Oct 21, 2024	Brent Spot US Dollar (XBR USD) Forum	neutral	usdjpy
Oct 3, 2024	Euro zone business activity contracted in September, PMI …	neutral	eurusd
Oct 2, 2024	France plans 60 bln euro budget squeeze for next year	neutral	eurusd
Sep 3, 2024	Dollar Gains for Fifth Day as US Traders Return From Holiday	neutral	eurusd
Aug 22, 2024	US stock futures higher despite climb in yields	neutral	usdjpy
Jul 30, 2024	Another Bank of Japan leak says 0.25% is under …	neutral	eurusd
Nov 24, 2023	Cimolai Wins Business Award After Surviving â‚¬300 Million …	neutral	usdjpy
Nov 20, 2023	Northern Ireland defeat group winners Denmark in final …	neutral	eurusd

The trained LLM predicted “Neutral” for every article it was given.

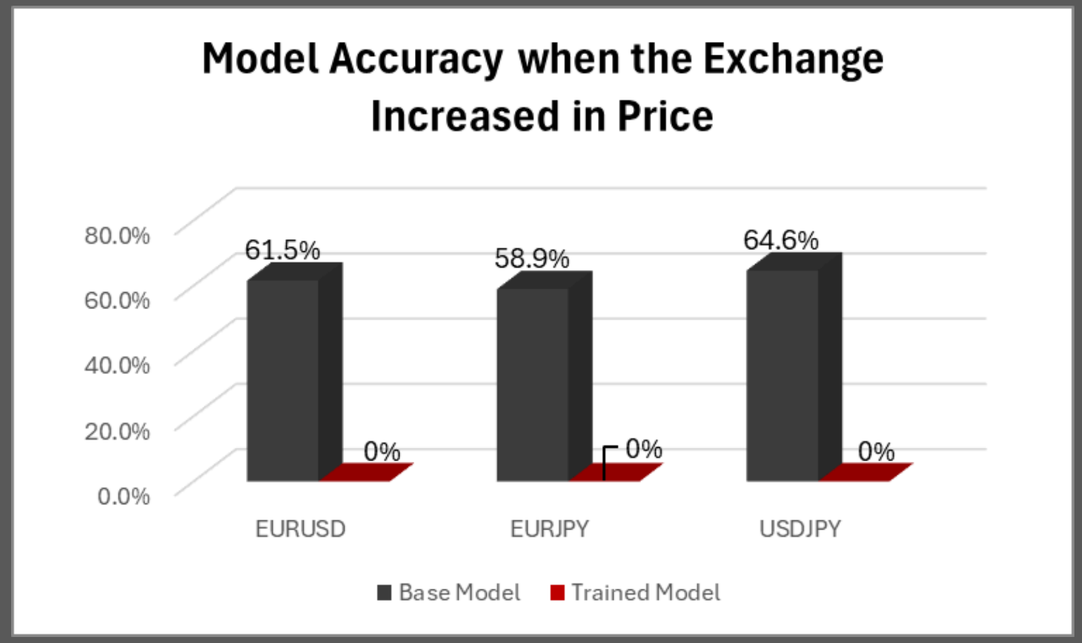
The trained LLM predicted “Neutral” for every article it was given because over 50% of the training data was had “Neutral” Signaling

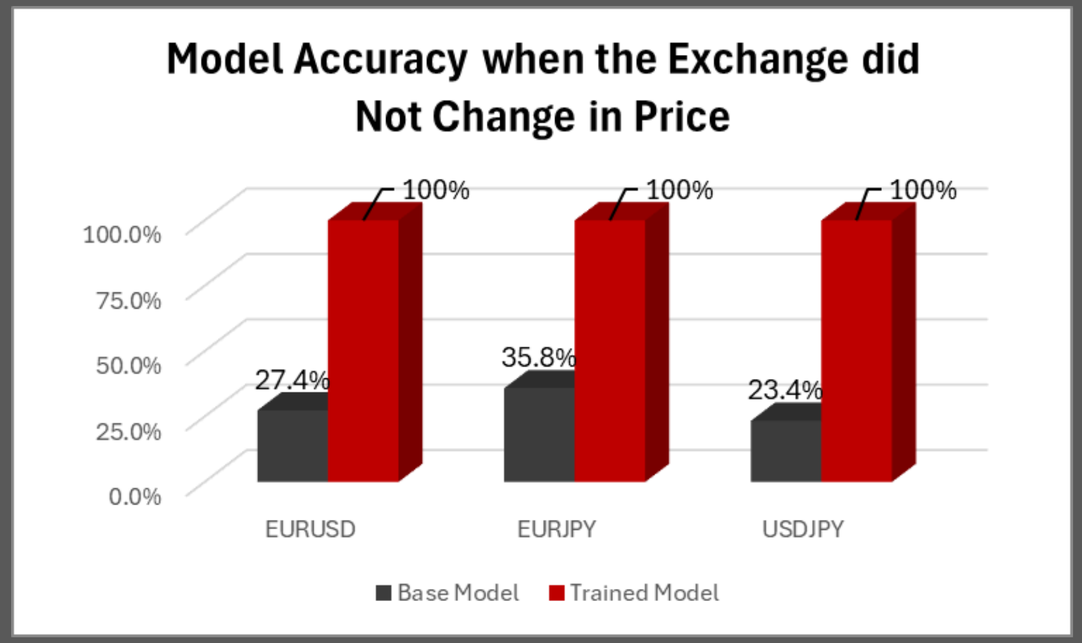


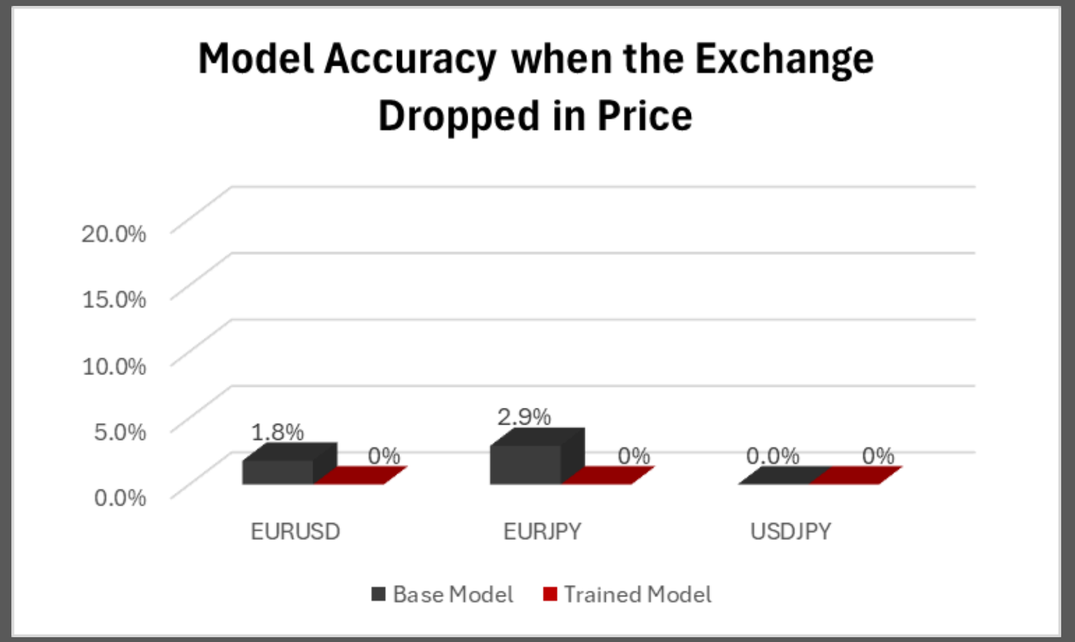


J.B. SPEED SCHOOL OF ENGINEERING

LLM's Performance Given the Reference Label







J.B. SPEED SCHOOL OF ENGINEERING

Discussion Points

If you knew your opponent would play rock 60% of the time, you should always choose paper if there are no other signs that they would play anything else

The training set was skewed heavily toward "neutral" labels, so the model learn to default to "neutral" to maximize accuracy

News articles don’t always indicate what direction the market will head in

It is safer to default to neutral

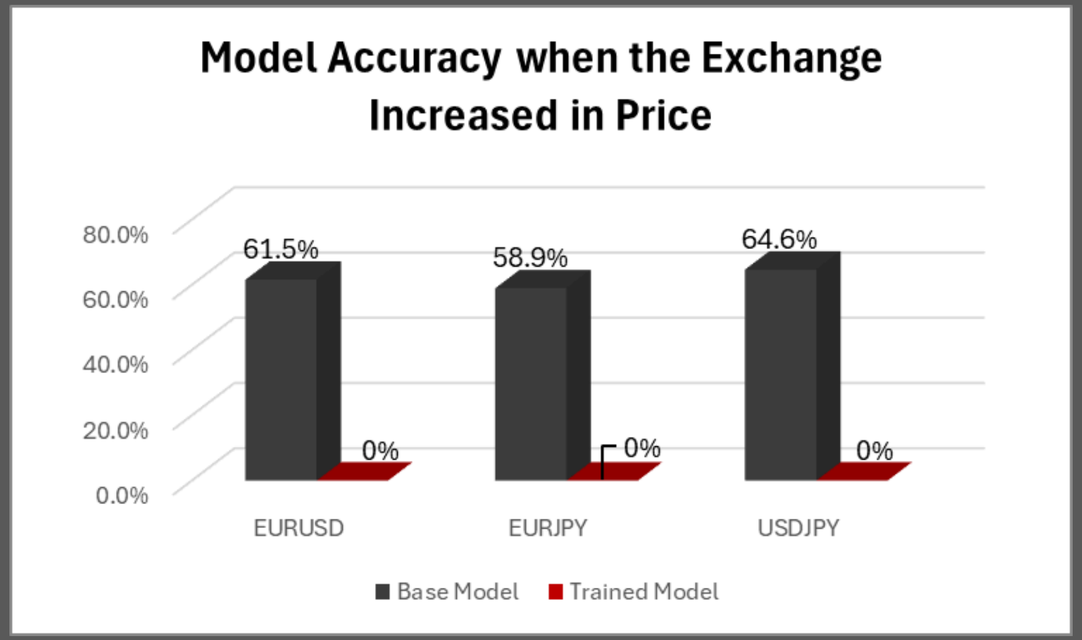
J.B. SPEED SCHOOL OF ENGINEERING

Discussion Points

The Base Model was biased to choose upward trends, why?

Did the model really understand how FOREX Pricing work?

Was the prompt too vague for the LLM?



J.B. SPEED SCHOOL OF ENGINEERING

Room for Improvements

- Larger and More Diverse Dataset
- Include longer article excerpts, not just titles
- Lower thresholds for more UP and DOWN labels
- Improve prompt clarity, use few-shot examples to guide the model more effectively
- Adapter Fusion: Combine adapters or use one multi-task model

Matthew Poncini
J.B. Speed School of Engineering
CSE-590-52-4255: Special Topics - GenAI
20 July 2025

CSE 590 Project: Fine-Tuning a Quantized LLM for Sentiment Signal Generation in FOREX News

Introduction

Thisproject involved utilizing an LLM to gain a competitive edge in the market. One of the models trained in this project gained more than 30% in accuracy. However, this statistic is not as impressive as it might seem. For this project, I fine-tuned a large language model (LLaMA- 2-7B) to classify sentiment in FOREX-related news article titles. The model was trained using LoRA adapters with QLoRA on a small labeled dataset consisting of economic news headlines aligned with future exchange rate changes for EURUSD, EURJPY, and USDJPY pairs. The model was trained to accurately predict whether an exchange rate would increase, decrease, or stay neutral given a news article title. A model trained specifically on the EURJPY exchange rate accurately classified the two-day pricing change 63.9% of the time. This model improved its score from 30.5% prior to fine-tuning. The 63.9% also beats ChatGPT GPT-4o's score of 47.9% accuracy on the same dataset.

As previously mentioned, this accurate model is not as impressive as it seems. The model accomplished this scoring by classifying every signal as "neutral." Historically, most exchange rates do not increase or decrease by more than 0.5% over two days. That's the threshold selected for each label. Neutral falls in between that threshold. The LLM did its job to minimize loss and yielded higher accuracy than GPT-4o. Some choices could have been made to ensure that the model would not simply predict neutral for every article it was fed, but how much accuracy would have been compromised? How much training would be required? All FOREX articles try to give insights on what the market will do, but are they right even half the time?

Despite the trivial results of the project, this paper will focus on the methodology of LLM training. It will continue the discussion on why the trained LLM performed the way it did. The paper will then provide next steps that could possibly lead an even more accurate model.

Methodology

Thisproject wascompleted in 4 discrete steps:

1.Data Collection / Processing
2.Base Model Testing 3.Model Fine-Tuning 4.Fine-Tuned Model Testing

Steps 2-4 were largely guided by the HW2 tutorial, while Step 1 introduced me to the world of webscraping. For the first time, I was unable to complete the human verification test.

Data Collection / Processing

It was a challenge to create a dataframe with real news articles and their publication dates. In the project proposal, I aimed to only use historical news articles from July 01, 2024 to December 31, 2024. However, I had to extend this back to 2010 to obtain an appropriate amount of data. I used SerpAPI to make searches on Google, Forexlive, Investing.com, Reuters, Bloomberg and other relevant websites to provide article titles and dates. I filtered my search by only looking at specific date ranges and looked only for articles pertaining to USD, EUR, and JPY.

```
queries = [
    'site:forexlive.com "USD/JPY"',
    'site:fxstreet.com "EUR/JPY""',
    'site:investing.com "US Dollar"',
    'site:reuters.com "Euro"',
    'site:bloomberg.com "Japanese yen"'
]

date_ranges = [
    ("01/01/2024", "01/01/2025"),
    ("01/01/2023", "01/01/2024")
]

for query in queries:
    for start_date, end_date in date_ranges:
        get_forex_articles(start_date, end_date, query, num_results=100, api_key=API_KEY)

# Save results to CSV
df = pd.DataFrame(all_results).drop_duplicates(subset="link")
df.to_csv("forex_articles_2024.csv", index=False)
print(f"Saved {len(df)} unique articles to 'forex_articles_2024.csv'")
```

I made sure to always utilize the pandas drop_duplicates function so that my dataset did not include two of the same article. A lot of my news articles were also just videos. I deleted all data containing the word video in it since I didn't think they would work as well to train the LLM. I ended up with 1160 news articles with dates.

It was important to only take articles with published dates because the dates are what will allow me to evaluate an exchange's price change over the next two days. To collect all of the FOREX price data, I was easily able to use the Yahoo Finance library in Python.

From there, all it took was some dataframe manipulation to provide me a reference table. The table provided the article title, the date, the price change of all three exchanges over the next two days, and the sentiment label that corresponded to each of those price changes.

title	link	parsed_date	EURUSD_2d_change_pct	EURJPY_2d_change_pct	USDJPY_2d_change_pct	eurusd_label	eurjpy_label	usdjpy_label
"A $3 Trillion Threat to Global Financial Markets Looms in ...	https://www.forexlive.com/centralbank/	3/29/2023	0.6008	2.255	1.6445	up	up	up
$MBAG USD | MoonBag US Dollar MEXC	https://www.investing.com/crypto/moor	10/23/2024	0.249	0.6796	0.4268	neutral	up	neutral
/macroeconomics/central-banks/boj	https://www.fxstreet.com/macroeconom	11/20/2024	-1.3034	-1.6245	-0.3232	down	down	neutral
1 EUR to USD | Euro to US Dollar Exchange Rate	https://uk.investing.com/currencies/eur	6/20/2023	0.6255	0.4709	-0.1487	up	neutral	neutral
1 USD to INR | US Dollar to Indian Rupee Exchange Rate	https://uk.investing.com/currencies/usc	12/11/2023	0.3056	0.5074	0.2048	neutral	up	neutral
1 USD to RUB | US Dollar to Russian Ruble Exchange Rate	https://uk.investing.com/currencies/usc	9/27/2024	-0.363	-1.2916	-0.9416	neutral	down	down
1.08 is a big level for EUR/USD, 2.8bn in strikes for May 30	https://www.forexlive.com/Orders/108-i	5/29/2024	-0.1517	-0.3856	-0.1965	neutral	neutral	neutral
104 years ago today, the Titanic hit an iceberg	https://www.forexlive.com/news/!/104-y	4/14/2016	0.1853	-1.1263	-1.2822	neutral	down	down
17311187 Quote - JPM The Japan Best Idea Fund	https://www.bloomberg.com/quote/173	4/1/2024	-0.224	-0.1316	0.0938	neutral	neutral	neutral
3 great trading tools to predict the price line movement	https://www.forexlive.com/Education/!/3	7/31/2020	-0.9175	0.3557	1.2867	down	neutral	up
3 Top Stock Picks to Benefit From US Dollar's Strength	https://www.investing.com/analysis/3-t	12/31/2024	-1.3256	-1.1176	0.2338	down	down	neutral
A better understanding of technical analysis and related ...	https://www.forexlive.com/Education/!/a	8/12/2019	-0.2805	0.8747	1.1463	neutral	up	up
A Bumper ISA Season â€" But Not For UK Stocks	https://www.bloomberg.com/news/new	5/8/2024	0.3225	0.7075	0.4142	neutral	up	neutral

Base Model Testing

I used model meta-llama/Llama-2-7b-hf as my base model. Meta-llama/Llama-2-7b-hf performs well on benchmarks like coding, reasoning, and knowledge tasks. It was used for HW 2 in further instruction-fine-tuning and so I thought it'd be a good choice for the project. To run the model on Kaggle, I quantized it using BitsAndBytesConfig, allowing for 4-bit precision and reduced memory usage.

I used a very straightforward, direct prompt. This choice was intentional to conserve memory during inference. In hindsight, I'm curious how altering the prompt structure might have affected model performance or output quality.

```
def make_prompt_for_inference(title, pair):
    return f"""### Instruction:
Predict the {pair.upper()} movement direction based on the headline. Only respond with one of the following: up, down, or neutral.

### Input:
{title}

### Response:"""



pairs = ["EURUSD", "EURJPY", "USDJPY"]
```

Model Fine-Tuning

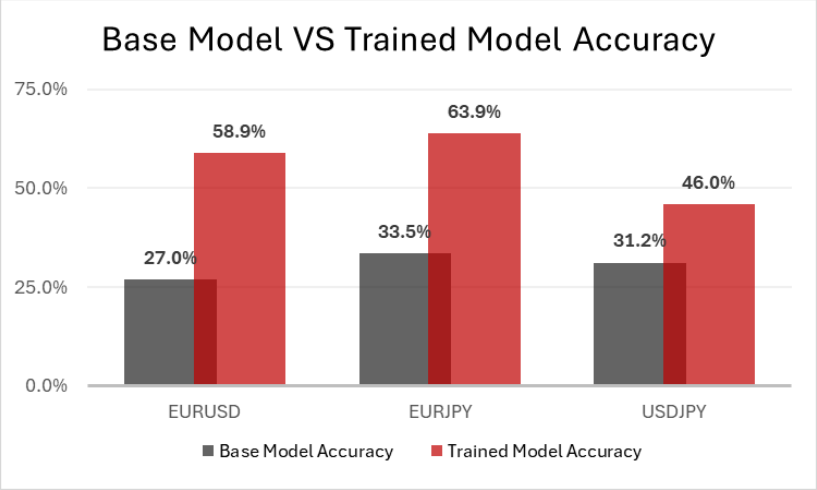
For fine-tuning, I used the LLaMA-2-7B base model with 4-bit quantization to reduce memory usage (and enable training on Kaggle hardware). The training process followed a parameter-efficient fine-tuning strategy using LoRA adapters. Only the q_proj and v_proj attention layers were updated. This approach significantly reduces computational overhead while maintaining performance. The training was implemented using the transformers, peft, trl, and bitsandbytes libraries. A total of 900 examples were used in the training dataset, and I fine-tuned three separate LoRA adapters, one for each currency pair: EURUSD, EURJPY, and USDJPY. This allowed each model to specialize in the sentiment patterns of its respective exchange.

Fine-Tuned Model Testing

For evaluation, I used a test set of 260 news article titles (outside of the training sample) that were also used during base model testing, ensuring a consistent comparison. To assess each fine-tuned model individually, I loaded one LoRA adapter at a time, corresponding to either EURUSD, EURJPY, or USDJPY. Each adapter was evaluated exclusively on its target exchange, predicting sentiment labels (Up, Down, or Neutral) based on the associated price movement.

Results

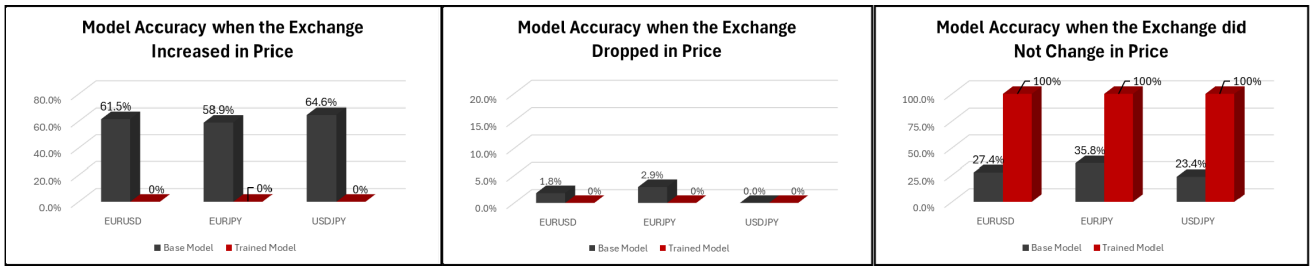
As mentioned in the introduction, the results are very impressive and trivial at the same time. Each training layer improved the model's performance significantly. The EURUSD exchange signal accuracy doubled and the EURJPY accuracy was significantly higher than GPT-4o's performance on the same test set.



This, of course, is trivial because the trained LLM classified every article as a neutral signal.

parsed_date	title	predicted_label	currency_pair
Dec 31, 2024	USD KZT | US Dollar Kazakh Tenge	neutral	eurusd
Nov 6, 2024	USD/JPY tumbling	neutral	usdjpy
Oct 21, 2024	Brent Spot US Dollar (XBR USD) Forum	neutral	usdjpy
Oct 3, 2024	Euro zone business activity contracted in September, PMI ...	neutral	eurusd
Oct 2, 2024	France plans 60 bln euro budget squeeze for next year	neutral	eurusd
Sep 3, 2024	Dollar Gains for Fifth Day as US Traders Return From Holiday	neutral	eurusd
Aug 22, 2024	US stock futures higher despite climb in yields	neutral	usdjpy
Jul 30, 2024	Another Bank of Japan leak says 0.25% is under ...	neutral	eurusd
Nov 24, 2023	Cimolai Wins Business Award After Surviving â‚¬300 Million ...	neutral	usdjpy
Nov 20, 2023	Northern Ireland defeat group winners Denmark in final ...	neutral	eurusd

It is revealing that the untrained model was simply guessing. Twice, it scored below 33.3% and once slightly above. This hints that it is simply not enough to look only at the article titles and that the article titles might actually be misleading. I also noticed that the pretrained model was way more inclined to select that the exchange rate would rise.



This leads me to think that the model lacked domain-specific content during its initial training. The model had a huge bias towards affirmative or upward outcomes. Possibly, the model defaulted to up when it was confused.

The Default to Neutrality

In a nutshell, I think the number one reason the model defaulted to neutral was that it could not find any patterns in the training data that would lead it to choose anything besides neutral. The training set was small; only 900 training examples. In this set, it's not obvious what would indicate volatility in the market. I also think that the prompts used in this project could have been too vague. Even if the model had FOREX-specific training, I think anyone would have a tough time choosing correctly just by looking at article titles. I chose this as my topic because I was curious if there was a hidden signal that only a trained AI would be able to pick up. The default to neutrality confirms the FOREX market is random and not very volatile (at least in these exchanges during this time period).

To avoid this default to neutrality, the obvious thing to do is to use class weighting or oversample minority classes during training. This would help the model look for patterns, but would most likely decrease its accuracy.

Discussion Points

What if good news for the US Dollar simply meant that it was not falling as fast to the Japanese Yen (or vice versa). The model would pick up the sentiment and still be incorrect. In this project, modern economic trends were not considered. If the model were also given the current exchange rate and its trend at the time, it'd likely help it make a better decision than the article title alone.

1. 

2. 

3. 

Take a look at the three articles above. What do they determine about EURUSD, EURJPY, and USDJPY? Should an AI be able to provide quick and accurate feedback on what will happen to exchange rates? Do the authors' biases cause inaccuracy?

	EURUSD	EURJPY	USDJPY
1.	neutral	down	neutral
2.	up	up	up
3.	neutral	neutral	neutral

Conclusion

Originally, the plan was to extract full articles from websites and use that as input for the sentiment signal. However, I soon found difficulties in extracting that data on a large scale and realized it would slow down the model training. I think that this model could be improved by simultaneously having the model trained on more quantitative data. Given a date, can the model accurately determine if an exchange will go up or down purely based on what it did the week before? The month before?

In a real-world sense, a trader is punished more when they choose “up” when the exchange falls, and similarly punished for choosing “down” when the exchange goes up. This project did not capture that reality, nor did the trained LLM. However, would that not just push the LLM to be safe a choose “neutral” even more? Vice versa, the LLM shouldn’t be punished as much when they choose “up” or “down” when the signal is neutral. After all, AUDJPY news doesn’t affect EURUSD news (unless it does). This project was a solid stepping stone in the world of machine learning and LLM fine-tuning, as well as an application to GenAI in the financial markets.

Work Cited

SerpApi. (n.d.). SerpApi [Computer software organization]. GitHub. Retrieved July 20, 2025, from GitHub repository.

Roussi, R. (n.d.). yfinance: Yahoo! Finance market data downloader [Python library]. GitHub. Retrieved July 20, 2025, from https://github.com/ranaroussi/yfinance

Meta. (2023). Llama-2-7b-hf: Pretrained 7 B LLaMA-2 model in Hugging Face format [Model card]. Hugging Face. Retrieved July 20, 2025, from Hugging Face repository

Hugging Face Documentation.

- PEFT (https://huggingface.co/docs/peft)
- Transformers Library (https://huggingface.co/docs/transformers)
- TRL (https://huggingface.co/docs/trl)

Appendix

```
!pip install bitsandbytes

!pip install -U bitsandbytes
```

```
!pip install bitsandbytes --no-cache-dir --prefer-binary

!pip install -q trl

!pip install git+https://github.com/huggingface/trl.git

!ls -l /kaggle/working
```

```
import pandas as pd

forex_labeled_dataset =
pd.read_csv("/kaggle/input/forex-labeled-dataset/forex_dataset_labeled.csv
")
```

```
forex_labeled_dataset.head(5)

from huggingface_hub import login

login("private_api_key")

shuffled = forex_labeled_dataset.sample(frac=1,
random_state=42).reset_index(drop=True)

#Split manually
train_data = shuffled.iloc[260:].reset_index(drop=True)

"""# This is Where the Training Happens"""
```

```
from peft import LoraConfig
from transformers import AutoModelForCausalLM, TrainingArguments
from trl import SFTTrainer
from datasets import Dataset

from transformers import AutoTokenizer, BitsAndBytesConfig
import torch
```

```
from datasets import Dataset

train_eurusd_dataset = Dataset.from_pandas(train_data)
```

```
model_id = "meta-llama/Llama-2-7b-chat-hf"

def make_prompt(example):
    return f"""### Instruction:
Predict the EURUSD direction based on the title. Only respond with Up,
Down, or Neutral.

### Input:
{example['title']}

### Response:
{example['eurusd_label']}"""

def tokenize(example):
    prompt = make_prompt(example)
    tokens = tokenizer(prompt, padding="max_length", truncation=True,
max_length=64)
    tokens["labels"] = tokens["input_ids"].copy()
    return tokens

bnb_config =BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=torch.float16
)

#Reloadbasemodel
model=AutoModelForCausalLM.from_pretrained(
    model_id,
    device_map="auto",
    quantization_config=bnb_config
)
model.config.use_cache = False
model.train()
```

```
tokenizer = AutoTokenizer.from_pretrained(model_id)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "left"

raw_dataset = Dataset.from_pandas(train_data)
tokenized_dataset = raw_dataset.map(tokenize, batched=False,
remove_columns=raw_dataset.column_names)

from transformers import DataCollatorForLanguageModeling

train_dataset = train_eurusd_dataset.map(tokenize, batched=False)

data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer,
mlm=False)

#LoRA setup
peft_config = LoraConfig(
    r=8,
    lora_alpha=32,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)

data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer,
mlm=False)

from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir="./eurusd-ft",              #Final model directory
    per_device_train_batch_size=2,
    gradient_accumulation_steps=2,
    num_train_epochs=3,
    learning_rate=2e-4,  fp16=True,
    logging_steps=10,



    #Prevent saving intermediate checkpoints
```

```
    save_strategy="no",                    #Do not save checkpoints
    save_total_limit=1,                   #Optional: Keep only the last
ifsave_strategy != "no"

    report_to="none"
)

trainer = SFTTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    data_collator=data_collator,
    peft_config=peft_config
)

#Train
trainer.train()

import shutil

#Save EURUSD Adapter
trainer.model.save_pretrained("./eurusd_adapter")
tokenizer.save_pretrained("./eurusd_adapter")
shutil.copytree("./eurusd_adapter", "/kaggle/outputs/eurusd_adapter")

#Zip the directory
shutil.make_archive("/kaggle/outputs/eurusd_adapter", 'zip',
"./eurusd_adapter")

import gc
import torch

gc.collect()
torch.cuda.empty_cache()

#Step 1: Load base model
base_model = AutoModelForCausalLM.from_pretrained(
    model_id,
    device_map="auto",
    quantization_config=bnb_config
```

```
)

from peft import PeftModel

#Step 2: Load the LoRA adapter into the base model
base_model.config.use_cache = False
base_model.train()

#Step 3: Load tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_id)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "left"

def make_prompt(example):
    return f"""### Instruction:
Predict the EURJPY direction based on the title. Only respond with Up,
Down, or Neutral.

### Input:
{example['title']}

### Response:
{example['eurjpy_label']}"""

from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir="./eurjpy-ft",            #Final model directory
    per_device_train_batch_size=2,
    gradient_accumulation_steps=2,
    num_train_epochs=3,
    learning_rate=2e-4,  fp16=True,
    logging_steps=10,



    #Prevent saving intermediate checkpoints
save_strategy="no",                       # Do not save checkpoints
save_total_limit=1,                      # Optional: Keep only the last
ifsave_strategy != "no"
```

```
    report_to="none"
)

fromtransformers import DataCollatorForLanguageModeling

data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer,
mlm=False)

raw_dataset = Dataset.from_pandas(train_data)
tokenized_dataset = raw_dataset.map(tokenize, batched=False,
remove_columns=raw_dataset.column_names)

trainer = SFTTrainer(
model=base_model,
args=training_args,
train_dataset=tokenized_dataset,
data_collator=data_collator,
peft_config=peft_config,
)

trainer.train()

import shutil

#Save EURUSD Adapter
trainer.model.save_pretrained("./eurjpy_adapter")
tokenizer.save_pretrained("./eurjpy_adapter")
shutil.copytree("./eurjpy_adapter", "/kaggle/outputs/eurjpy_adapter")

#Zip the directory
shutil.make_archive("/kaggle/outputs/eurjpy_adapter", 'zip',
"./eurjpy_adapter")

#Step 1: Load base model
base_model = AutoModelForCausalLM.from_pretrained(
    model_id,
    device_map="auto",
    quantization_config=bnb_config
)
```

```
base_model.config.use_cache = False
base_model.train()

#Step 3: Load tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_id)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "left"

def make_prompt(example):
    return f"""### Instruction:
Predict the USDJPY direction based on the title. Only respond with Up,
Down, or Neutral.

### Input:
{example['title']}

### Response:
{example['usdjpy_label']}"""

raw_dataset = Dataset.from_pandas(train_data)
tokenized_dataset = raw_dataset.map(tokenize, batched=False,
remove_columns=raw_dataset.column_names)

def tokenize(example):
    prompt = make_prompt(example)
    tokens = tokenizer(prompt, padding="max_length", truncation=True,
max_length=64)
    tokens["labels"] = tokens["input_ids"].copy()
    return tokens

fromtransformers import TrainingArguments

training_args = TrainingArguments(
    output_dir="./usdjpy-ft",          #Final model directory
    per_device_train_batch_size=2,
    gradient_accumulation_steps=2,
    num_train_epochs=3,
    learning_rate=2e-4,  fp16=True,
    logging_steps=10,
```

```
    #Prevent saving intermediate checkpoints
save_strategy="no",                          # Do not save checkpoints
save_total_limit=1,                         # Optional: Keep only the last
ifsave_strategy != "no"

    report_to="none"
)

data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer,
mlm=False)

trainer = SFTTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    data_collator=data_collator,
    peft_config=peft_config,
    compute_metrics=None, # <--- disable this
)

trainer.train()

trainer.model.save_pretrained("./usdjpy_adapter")
tokenizer.save_pretrained("./usdjpy_adapter")
shutil.copytree("./usdjpy_adapter", "/kaggle/outputs/usdjpy_adapter")

#Zip the directory
shutil.make_archive("/kaggle/outputs/usdjpy_adapter", 'zip',
"./usdjpy_adapter")

import shutil

#Create a zip file of the entire working directory
shutil.make_archive("/kaggle/working/working_dir_backup", 'zip',
"/kaggle/working/")

from peft import PeftModel
from transformers import AutoTokenizer, AutoModelForCausalLM,
BitsAndBytesConfig
```

```
import torch

model_id = "meta-llama/Llama-2-7b-chat-hf"
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=torch.float16
)

#Step 1: Load base model
base_model = AutoModelForCausalLM.from_pretrained(model_id,
device_map="auto", quantization_config=bnb_config)

#Step 2: Wrap with PEFT (initial adapter)
model = PeftModel.from_pretrained(base_model,
"/kaggle/working/eurusd_adapter", adapter_name="eurusd")

#Step 3: Load additional adapters
model.load_adapter("/kaggle/working/eurjpy_adapter",
adapter_name="eurjpy")
model.load_adapter("/kaggle/working/usdjpy_adapter",
adapter_name="usdjpy")

#Step 4: Choose one adapter to activate
model.set_adapter("usdjpy") # or "eurjpy", "eurusd"
model.eval()

#Step 5: Load tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_id)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "left"

#Example test input
title = "BOJ considers policy shift amid inflation pressure"

#Format the prompt just like during training
prompt = f"""### Instruction:
Predict the USDJPY direction based on the title. Only respond with Up,
Down, or Neutral.
```

```
### Input:
{title}

###Response:"""

#Tokenize input
inputs = tokenizer(prompt, return_tensors="pt").to(model.device)

#Generateoutput
withtorch.no_grad():
    output_ids = model.generate(
        input_ids=inputs["input_ids"],
        attention_mask=inputs["attention_mask"],
        max_new_tokens=10,
        do_sample=False,
        pad_token_id=tokenizer.eos_token_id
    )

#Decodeoutput
response=tokenizer.decode(output_ids[0], skip_special_tokens=True)

#Print just the response portion
print("Model output:\n", response.split("### Response:")[-1].strip())

fromdatasetsimport Dataset
importpandasaspd

#Step1:Loadyour test data
df = pd.read_csv("/kaggle/input/your-test-data.csv")  #Replace with your
actual test CSV
dataset=Dataset.from_pandas(df)

#Step 2: Set active adapter (e.g., "eurusd", "eurjpy", "usdjpy")
model.set_adapter("eurusd") # change this based on test

#Step 3: Create prompt
defmake_prompt(example):
    return f"""### Instruction:
```

```
Predict the EURUSD direction based on the title. Only respond with Up,
Down, or Neutral.

### Input:
{example['title']}

###Response:"""

#Step 4: Generate predictions
defgenerate_prediction(example):
prompt = make_prompt(example)
inputs = tokenizer(prompt, return_tensors="pt", padding=True,
truncation=True, max_length=64).to(model.device)

    withtorch.no_grad():
        outputs=model.generate(
            input_ids=inputs["input_ids"],
            attention_mask=inputs["attention_mask"],
            max_new_tokens=10,
            do_sample=False,
            pad_token_id=tokenizer.eos_token_id
        )

    decoded=tokenizer.decode(outputs[0], skip_special_tokens=True)
    response=decoded.split("### Response:")[-1].strip()
    return{"prediction": response}

#Step5:Runpredictions
results=dataset.map(generate_prediction)

#Step 6: Save predictions
results_df = results.to_pandas()
results_df.to_csv("/kaggle/working/eurusd_predictions.csv", index=False)
results_df.head()
```