# Meme Coin Generator with LLaVA-Next

This project implements a meme coin name and ticker generator using the LLaVA-Next (7B Mistral) model. It takes an image and tweet text as input and generates a meme coin name and ticker in JSON format.

## Table of Contents

## Requirements

- Python 3.10+
- PyTorch 2.0+
- Transformers 4.36+
- CUDA-compatible GPU with at least 24GB VRAM (recommended)
- RunPod account

## Installation

1. Clone this repository:

```
git clone https://github.com/yourusername/meme-coin-generator.git
cd meme-coin-generator
```

2. Install dependencies:

```
chmod +x setup.sh
./setup.sh
```

# Dataset Preparation

1. Prepare your dataset in CSV format with the following columns:
   - Tweet Text
   - Image URL
   - Token Name
   - Ticker
2. Run the dataset preparation script:

```
python prepare_dataset.py --csv_path your_dataset.csv --output_dir data
```

This will:

- Download images from the URLs
- Format the data for fine-tuning
- Split the data into training and validation sets

# Fine-tuning

To fine-tune the LLaVA-Next model on your dataset:

```
python finetune.py \
  --model_name llava-hf/llava-v1.6-mistral-7b-hf \
  --data_dir data \
  --output_dir fine_tuned_model \
  --batch_size 1 \
  --epochs 3 \
  --learning_rate 2e-5 \
  --gradient_accumulation_steps 4 \
  --use_lora \
  --quantize
```

Options:

- `--model_name`: Base model name or path
- `--data_dir`: Directory containing the prepared dataset
- `--output_dir`: Directory to save the fine-tuned model
- `--batch_size`: Batch size for training
- `--epochs`: Number of training epochs
- `--learning_rate`: Learning rate
- `--gradient_accumulation_steps`: Gradient accumulation steps

- `--use_lora`: Use LoRA for parameter-efficient fine-tuning
- `--quantize`: Use 4-bit quantization to reduce memory usage

## Inference

To run inference with the fine-tuned model:

```
python inference.py \
  --model_path fine_tuned_model \
  --image_url "https://example.com/image.jpg" \
  --tweet_text "Sample tweet text" \
  --use_lora \
  --quantize
```

Options:

- `--model_path`: Path to the fine-tuned model
- `--image_url`: URL of the image
- `--tweet_text`: Tweet text
- `--use_lora`: Whether the model is fine-tuned with LoRA
- `--base_model`: Base model path (for LoRA)
- `--quantize`: Use 4-bit quantization

## API Server

To start the API server:

```
export MODEL_PATH="fine_tuned_model"
export USE_LORA="true"
export BASE_MODEL="llava-hf/llava-v1.6-mistral-7b-hf"
export QUANTIZE="true"
export PORT=8000

python api_server.py
```

The API server will be available at `http://localhost:8000`.

## RunPod Deployment

To deploy on RunPod:

1. Create a new RunPod instance with a GPU that has at least 24GB VRAM

2. Upload all the files to the RunPod instance
3. Run the RunPod setup script:

```
chmod +x runpod_setup.sh
./runpod_setup.sh
```

This will:

- Set up the environment
- Install dependencies
- Start the API server

# Sample Usage

To use the API:

```
python sample_usage.py \
  --api_url "http://your-runpod-ip:8000" \
  --image_url "https://example.com/image.jpg" \
  --tweet_text "Sample tweet text"
```

This will send a request to the API and print the response in JSON format:

```
{
  "tokenName": "Generated Token Name",
  "ticker": "TICKER"
}
```

# Notes

- The model requires a GPU with at least 24GB VRAM for fine-tuning
- Using LoRA and quantization can reduce memory requirements
- The API server can be deployed on RunPod or any other cloud provider
- The model is fine-tuned to generate meme coin names and tickers based on images and tweet text