Tensoreflow teachable machine model API end point

Environment requirement:

# Create a virtual environment

```
python -m venv fastapi-teachable-env
```

# Activate the virtual environment

# Windows

```
fastapi-teachable-env\Scripts\activate
```

```
# macOS/Linux
source fastapi-teachable-env/bin/activate
```

# Install required packages

```
pip install fastapi uvicorn tensorflow pillow numpy
```

API Code in python

```
from fastapi import FastAPI, File, UploadFile

import numpy as np

from tensorflow.keras.layers import TFSMLayer

from io import BytesIO

from PIL import Image


app = FastAPI()


# Load your model using TFSMLayer

model = TFSMLayer('Model/', call_endpoint='serving_default')


# Load class names

with open('labels.txt', 'r') as f:

    class_names = f.read().splitlines()


def preprocess_image(image_bytes):

    img = Image.open(image_bytes).convert("RGB").resize((224, 224))  # Convert to RGB

    img_array = np.array(img) / 255.0  # Normalize

    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension

    return img_array
```

```
@app.post("/predict/")
async def predict(file: UploadFile = File(...)):
   try:
      # Read the file and preprocess it
      image_data = await file.read()
      img = preprocess_image(BytesIO(image_data))

      # Make the prediction using TFSMLayer
      predictions = model(img)

      # Log the model output for debugging
      print("Model Output:", predictions)  # Log the raw predictions

      # Access the predictions from the output dictionary
      prediction_tensor = predictions['sequential_3']

      # Convert tensor to numpy array for further processing
      predictions_array = prediction_tensor.numpy()

      # Extract the predicted class and confidence
      predicted_class_index = np.argmax(predictions_array)
      predicted_class = class_names[predicted_class_index]  # Ensure you have the correct class
names
      confidence = predictions_array[0][predicted_class_index]  # Access confidence

      # Return the result as a JSON response
      return {"predicted_class": predicted_class, "confidence": float(confidence)}
   except Exception as e:
      return {"error": str(e)}
```

Project Structure

```
project/
├── app.py            # FastAPI app code
├── model/             # Model folder
   ├── saved_model.pb   # TensorFlow model
   ├── assets/        # (Optional, may be empty)
   └── variables/      # Model weights
        ├── variables.data-00000-of-00001
        └── variables.index
├── labels.txt         # Contains the class labels
└── fastapi-teachable-env/
```