Image Classifier with TensorFlow/Keras

I'll create a complete image classifier application that allows users to upload images and get predictions about their content. This solution uses a pre-trained model for accurate object classification.

```python
import tkinter as tk

from tkinter import ttk, filedialog

from PIL import Image, ImageTk

import numpy as np

import tensorflow as tf

from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2, preprocess_input, decode_predictions


class ImageClassifierApp:

    def __init__(self, root):

        self.root = root

        self.root.title("Image Classifier")

        self.root.geometry("800x600")

        self.root.configure(bg="#f0f0f0")


        # Load pre-trained model

        self.model = MobileNetV2(weights='imagenet')


        # Create GUI

        self.create_widgets()
    def create_widgets(self):
        # Title
        title_label = tk.Label(self.root, text="Image Classifier", font=("Arial", 18, "bold"),
bg="#f0f0f0")
        title_label.pack(pady=10)

        # Instructions
        instruction_label = tk.Label(self.root, text="Upload an image to classify objects within
it",
                                font=("Arial", 12), bg="#f0f0f0")
        instruction_label.pack(pady=5)

        # Upload button
        self.upload_btn = ttk.Button(self.root, text="Upload Image",
command=self.upload_image)
        self.upload_btn.pack(pady=10)

        # Image display
        self.image_label = tk.Label(self.root, bg="#f0f0f0")
        self.image_label.pack(pady=10)

        # Results frame
        results_frame = tk.Frame(self.root, bg="#f0f0f0")
        results_frame.pack(pady=10, fill=tk.BOTH, expand=True)

        # Results title
        results_title = tk.Label(results_frame, text="Classification Results",
                    font=("Arial", 14, "bold"), bg="#f0f0f0")
        results_title.pack(pady=5)

        # Results treeview
        columns = ("rank", "class", "confidence")
        self.results_tree = ttk.Treeview(results_frame, columns=columns, show="headings",
height=8)

        self.results_tree.heading("rank", text="Rank")
        self.results_tree.column("rank", width=60, anchor=tk.CENTER)

        self.results_tree.heading("class", text="Class")
        self.results_tree.column("class", width=300, anchor=tk.W)

        self.results_tree.heading("confidence", text="Confidence")
        self.results_tree.column("confidence", width=100, anchor=tk.CENTER)

        # Scrollbar for results
        scrollbar = ttk.Scrollbar(results_frame, orient=tk.VERTICAL,
command=self.results_tree.yview)
        self.results_tree.configure(yscrollcommand=scrollbar.set)

        self.results_tree.pack(side=tk.LEFT, fill=tk.BOTH, expand=True, padx=10)
        scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

        # Status bar
```

```
        self.status_var = tk.StringVar()
        self.status_var.set("Ready to upload an image")
        status_bar = tk.Label(self.root, textvariable=self.status_var, relief=tk.SUNKEN,
                              anchor=tk.W, bg="#f0f0f0")
        status_bar.pack(side=tk.BOTTOM, fill=tk.X)

    def upload_image(self):
        file_path = filedialog.askopenfilename(
            title="Select an image",
            filetypes=[("Image files", "*.jpg *.jpeg *.png *.bmp")]
        )

        if file_path:
            self.status_var.set("Processing image...")
            self.root.update()

            try:
                # Process and display image
                image = Image.open(file_path)
                image.thumbnail((300, 300))  # Resize for display
                photo = ImageTk.PhotoImage(image)
                self.image_label.configure(image=photo)
                self.image_label.image = photo

                # Classify image
                predictions = self.classify_image(file_path)
            # Display results
            self.display_results(predictions)

            self.status_var.set("Classification complete")

        except Exception as e:
            self.status_var.set(f"Error: {str(e)}")

    def classify_image(self, image_path):
        # Load and preprocess image
        img = Image.open(image_path).convert('RGB')
        img = img.resize((224, 224))  # MobileNetV2 expects 224x224 images

        img_array = np.array(img)
        img_array = preprocess_input(img_array)
        img_array = np.expand_dims(img_array, axis=0)

        # Make prediction
        predictions = self.model.predict(img_array)

        # Decode predictions
        decoded_predictions = decode_predictions(predictions, top=5)[0]

        return decoded_predictions
    def display_results(self, predictions):
        # Clear previous results
        for item in self.results_tree.get_children():
            self.results_tree.delete(item)

        # Add new results
        for i, (imagenet_id, label, score) in enumerate(predictions, 1):
            self.results_tree.insert("", "end", values=(i, label.title(), f"{score:.2%}"))


if __name__ == "__main__":
    root = tk.Tk()
    app = ImageClassifierApp(root)
    root.mainloop()
```

How to Use the Image Classifier

1. Run the program (you'll need to install the required dependencies first)
2. Click the "Upload Image" button
3. Select an image file from your computer
4. The application will display the image and show the top 5 predicted classes with confidence scores

Requirements

To run this application, you'll need to install the following dependencies:

```bash
pip install tensorflow pillow numpy
```

Features

· User-friendly graphical interface

· Supports common image formats (JPG, JPEG, PNG, BMP)

· Uses MobileNetV2 pre-trained on ImageNet for accurate classification

· Displays top 5 predictions with confidence percentages

· Responsive design with scrollable results