PYTHON Tensorflow Bildsortierung

> Hierfür braucht man relativ große Bilddatensätze als Beispiele für Tensorflow um zuverlässige Ergebnisse zu bekommen.

Situation

Ihr habt ein großen Ordner voller Bilder und mit diesen über

https://teachablemachine.withgoogle.com ein Modell mit zwei Klassen erstellt und ALS QUANTITISIERT exportiert.

Script

python 3.9

pip install tensorflow pillow

```
#!/usr/bin/env python3
# sorter_tm.py

import os
import sys
import shutil
from PIL import Image
import numpy as np
import tensorflow as tf
tflite = tf.lite

def load_labels(label_path):
    """Liest die Klassennamen aus einer labels.txt im Format '0 Klasse1'."""
    labels = []
    with open(label_path, 'r', encoding='utf-8') as f:
```

```
        for line in f:
            line = line.strip()
            if not line:
                continue
            parts = line.split(' ', 1)
            if len(parts) < 2:
                raise ValueError(f"Ungültiges Format in labels.txt: '{line}'")
            labels.append(parts[1])
    if not labels:
        raise ValueError("Die labels.txt ist leer oder enthält keine gültigen Einträge.")
    return labels

def prepare_interpreter(model_path):
    """Initialisiert den TFLite-Interpreter."""
    interpreter = tflite.Interpreter(model_path=model_path)
    interpreter.allocate_tensors()
    input_details  = interpreter.get_input_details()[0]
    output_details = interpreter.get_output_details()[0]
    return interpreter, input_details, output_details

def classify_image(interpreter, input_details, output_details, image, target_size):
    input_shape = input_details['shape']
    input_dtype = input_details['dtype']

    # Resize des PIL-Bildes
    img = image.resize(target_size)
    arr = np.array(img)

    if input_dtype == np.float32:
        arr = arr.astype(np.float32) / 255.0
    elif input_dtype == np.uint8:
        arr = arr.astype(np.uint8)
    else:
        raise ValueError(f"Nicht unterstützter Eingabetyp: {input_dtype}")

    arr = np.expand_dims(arr, axis=0)
    interpreter.set_tensor(input_details['index'], arr)
    interpreter.invoke()

    output = interpreter.get_tensor(output_details['index'])[0]
```

```
    output = output.astype(np.float32)  # Konvertiere Ausgabe zu float32 für Softmax
    probabilities = tf.nn.softmax(output).numpy()
    idx = np.argmax(probabilities)
    confidence = probabilities[idx]
    return idx, confidence, probabilities

def main(model_path, label_path, src_dir, dst_root, image_size):
    # Labels und Interpreter laden
    labels = load_labels(label_path)
    interpreter, in_det, out_det = prepare_interpreter(model_path)

    # Ausgabeordnerstruktur anlegen
    for lbl in labels:
        os.makedirs(os.path.join(dst_root, lbl), exist_ok=True)

    # Jedes Bild klassifizieren und verschieben
    for filename in os.listdir(src_dir):
        if not filename.lower().endswith((".jpg", ".jpeg", ".png", ".bmp", ".gif")):
            continue

        image_path = os.path.join(src_dir, filename)
        try:
            with Image.open(image_path) as img:
                img = img.convert("RGB")
        except Exception as e:
            print(f"Bild konnte nicht geöffnet werden: {filename} – {e}")
            continue

        idx, confidence, probabilities = classify_image(interpreter, in_det, out_det, img,
image_size)
        label = labels[idx]

        target_dir = os.path.join(dst_root, label)
        os.makedirs(target_dir, exist_ok=True)

        target_path = os.path.join(target_dir, filename)
        try:
            shutil.move(image_path, target_path)
            print(f"{filename} → {label} ({confidence:.2%}) – Verteilung: {probabilities}")
        except Exception as e:
```

```
            print(f"Fehler beim Verschieben von {filename}: {e}")

if __name__ == "__main__":
    if len(sys.argv) != 6:
        print("Usage: python sorter_tm.py <model.tflite> <labels.txt> "
              "<source_dir> <destination_root> <image_size>")
        print("Beispiel: python sorter_tm.py model/model.tflite model/labels.txt "
              "input_images sorted_images 224")
        sys.exit(1)

    model_file       = sys.argv[1]
    labels_file      = sys.argv[2]
    source_folder    = sys.argv[3]
    destination_root = sys.argv[4]
    size             = int(sys.argv[5])

    main(model_file, labels_file, source_folder, destination_root, (size, size))
```

Vorgängerversionen

V2: Kann noch nicht mit Tensorflow Labels umgehen und erwartet feste Klassen.

```
#!/usr/bin/env python3
# sorter_tm.py

import os
import sys
import shutil
from PIL import Image
import numpy as np
import tensorflow as tf
tflite = tf.lite

def load_labels(label_path):
    """Liest die Klassennamen aus einer labels.txt."""
    with open(label_path, 'r', encoding='utf-8') as f:
        labels = [line.strip() for line in f if line.strip()]
```

```
    if len(labels) != 2 or set(labels) != {"Good", "Bad"}:
        raise ValueError("labels.txt muss genau die Einträge 'Good' und 'Bad' enthalten.")
    return labels

def prepare_interpreter(model_path):
    """Initialisiert den TFLite-Interpreter."""
    interpreter = tflite.Interpreter(model_path=model_path)
    interpreter.allocate_tensors()
    input_details  = interpreter.get_input_details()[0]
    output_details = interpreter.get_output_details()[0]
    return interpreter, input_details, output_details

def classify_image(interpreter, input_details, output_details, image, target_size):
    input_shape = input_details['shape']
    input_dtype = input_details['dtype']

    # Resize des PIL-Bildes
    img = image.resize(target_size)
    arr = np.array(img)

    if input_dtype == np.float32:
        arr = arr.astype(np.float32) / 255.0
    elif input_dtype == np.uint8:
        arr = arr.astype(np.uint8)
    else:
        raise ValueError(f"Nicht unterstützter Eingabetyp: {input_dtype}")

    arr = np.expand_dims(arr, axis=0)
    interpreter.set_tensor(input_details['index'], arr)
    interpreter.invoke()

    output = interpreter.get_tensor(output_details['index'])[0]
    output = output.astype(np.float32)  # Konvertiere Ausgabe zu float32 für Softmax
    probabilities = tf.nn.softmax(output).numpy()
    idx = np.argmax(probabilities)
    confidence = probabilities[idx]
    return idx, confidence, probabilities

def main(model_path, label_path, src_dir, dst_root, image_size):
```

```
    # Labels und Interpreter laden
    labels = load_labels(label_path)
    interpreter, in_det, out_det = prepare_interpreter(model_path)

    # Ausgabeordnerstruktur anlegen
    for lbl in labels:
        os.makedirs(os.path.join(dst_root, lbl), exist_ok=True)

    # Jedes Bild klassifizieren und verschieben
    for filename in os.listdir(src_dir):
        if not filename.lower().endswith((".jpg", ".jpeg", ".png", ".bmp", ".gif")):
            continue

        image_path = os.path.join(src_dir, filename)
        try:
            with Image.open(image_path) as img:
                img = img.convert("RGB")
        except Exception as e:
            print(f"Bild konnte nicht geöffnet werden: {filename} – {e}")
            continue

        idx, confidence, probabilities = classify_image(interpreter, in_det, out_det, img,
image_size)
        label = labels[idx]

        target_dir = os.path.join(dst_root, label)
        os.makedirs(target_dir, exist_ok=True)

        target_path = os.path.join(target_dir, filename)
        try:
            shutil.move(image_path, target_path)
            print(f"{filename} → {label} ({confidence:.2%}) – Verteilung:
{probabilities}")
        except Exception as e:
            print(f"Fehler beim Verschieben von {filename}: {e}")

if __name__ == "__main__":
    if len(sys.argv) != 6:
        print("Usage: python sorter_tm.py <model.tflite> <labels.txt> "
```

```
            "<source_dir> <destination_root> <image_size>")
        print("Beispiel: python sorter_tm.py model/model.tflite model/labels.txt "
            "input_images sorted_images 224")
        sys.exit(1)

    model_file       = sys.argv[1]
    labels_file      = sys.argv[2]
    source_folder    = sys.argv[3]
    destination_root = sys.argv[4]
    size             = int(sys.argv[5])

    main(model_file, labels_file, source_folder, destination_root, (size, size))
```

Notige Struktur:
C:\BildSorter\
├── model\
│ ├── model.tflite ← exportiertes Modell von Teachable Machine
│ └── labels.txt ← labels
├── input_images\ ← HIER liegen Ihre zu sortierenden Bilder
├── sorted_images\ ← WIRD automatisch befüllt mit Unterordnern
├── sorter_tm.py ← das Python-Skript

Run:

python sorter_tm.py model/model.tflite model/labels.txt input_images sorted_images 224

Version #5
Erstellt: 2025-04-19 13:54:23 UTC von Konstantin
Zuletzt aktualisiert: 2025-06-05 14:44:03 UTC von Konstantin