📌 Fine-Tuning Florence-2 on a License Plate Dataset with EasyOCR

🏰 Project Overview

This project fine-tunes Microsoft's **Florence-2** model on a **license plate dataset** using **EasyOCR**. The goal is to improve the model's accuracy in detecting and recognizing license plates from images.

👅 Dependencies Installation

The notebook begins by installing the required Python libraries:

!pip install einops sklearn python-Levenshtein datasets timm jiwer

These libraries are essential for:

- **Image processing** (`einops`, `timm`)
- **Levenshtein distance for OCR accuracy** (`python-Levenshtein`)
- **Dataset handling** (`datasets`)
- **Evaluation metrics** (`jiwer`)

🚀 Loading Florence-2 Model & Processor

from transformers import AutoModelForCausalLM, AutoProcessor

import torch

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

print(device)

model = AutoModelForCausalLM.from_pretrained(

"microsoft/Florence-2-base-ft", trust_remote_code=True, revision='refs/pr/6'

).to(device)

```
processor = AutoProcessor.from_pretrained(
    "microsoft/Florence-2-base-ft", trust_remote_code=True, revision='refs/pr/6'
)
torch.cuda.empty_cache()
```

- **Loads the Florence-2 model**, optimized for image-to-text tasks.
- **Detects GPU availability** and loads the model onto it for faster processing.
- **Clears CUDA cache** to free up memory.

📊 Loading the License Plate Dataset

```
import datasets

dataset = datasets.load_dataset("keremberke/license-plate-object-detection", 'mini')
dataset
```

This dataset contains:

- Images of license plates 📸
- Corresponding text labels of detected plates 🔤
- Bounding box information 🛁

🔍 Inspecting Dataset Samples

```
for idx in range(len(dataset["train"])):
    sample = dataset["train"][idx]
    print(f"🔍 Image ID: {sample['image_id']}")
    print(f"📸 Image: {sample['image']}")
    print(f"✅ Ground Truth Label: {sample['objects']}")
```

- **Loops through training samples** and prints:
    - Image ID

    - Image data
    - Ground truth label (actual text on license plate)

📂 Splitting Dataset into Training, Testing & Validation Sets

```
ds_train = dataset["train"]

ds_test = dataset["test"]

ds_val = dataset["validation"]
```

- **Divides data** into training, validation, and test sets.

🏰 Running Inference on the Model

```
# Function to run the model on an example

def run_example(task_prompt, text_input, image):

    prompt = task_prompt + text_input


    # Ensure the image is in RGB mode

    if image.mode != "RGB":

        image = image.convert("RGB")


    inputs = processor(text=prompt, images=image, return_tensors="pt").to(device)

    generated_ids = model.generate(

        input_ids=inputs["input_ids"],

        pixel_values=inputs["pixel_values"],

        max_new_tokens=1024,

        num_beams=3

    )

    generated_text = processor.batch_decode(generated_ids, skip_special_tokens=False)[0]
```

```
    parsed_answer = processor.post_process_generation(generated_text, task=task_prompt, image_size=(image.width, image.height))

    return parsed_answer
```

- **Takes an image and runs the model inference**.
- **Extracts the license plate text** based on a given question.

🧪 Testing the Model on Sample Images

```
# Test on the first 3 samples from the train dataset

for idx in range(3):

    sample = dataset["train"][idx]

    image = sample["image"]


    print(run_example("DocVQA", "What is the license plate number?", image))

    display(image)
```

- **Runs inference on three sample images**.
- **Displays images and predicted license plate numbers**.

🎯 Evaluating OCR Accuracy

```
from jiwer import wer


def calculate_ocr_accuracy(predictions, ground_truths):

    errors = [wer(gt, pred) for gt, pred in zip(ground_truths, predictions)]

    return 1 - sum(errors) / len(errors)
```

- Uses **Word Error Rate (WER)** to measure OCR accuracy.
- Compares predicted license plate text with ground truth labels.

❓ Frequently Asked Questions (FAQs)

1️⃣ Why is the OCR result sometimes incorrect?

OCR errors can occur due to:

- Blurry or low-resolution images 📷
- Poor lighting conditions 🌑
- Unusual fonts or characters 🔤

2️⃣ How can OCR accuracy be improved?

- Increase dataset size 📊
- Use higher-quality images 📸
- Fine-tune the model further with more epochs 🏃‍♂️

3️⃣ Why does the model fail on some images but not others?

Different conditions such as angle, lighting, and occlusions affect the recognition performance. Proper dataset augmentation can help!

4️⃣ How is OCR accuracy measured in this project?

We use **Word Error Rate (WER)** and **Levenshtein distance**, which compare the predicted text with the ground truth label.

5️⃣ Can this model be used in real-time applications?

Yes! With GPU acceleration, inference time is reduced, making it suitable for **real-time license plate recognition** in surveillance and traffic systems 🚦.

📢 **Client Takeaway:** This documentation provides a step-by-step breakdown of the model training, dataset processing, and OCR evaluation for license plate recognition using **Microsoft Florence-2 and EasyOCR**.