```
!pip install --upgrade pip
!pip install torch torchvision torchaudio
!pip install fsspec==2024.6.1
!pip install datasets==3.0.0
!pip install gcsfs==2024.6.0
!pip install jiwer
!pip install evaluate

# Imports
from google.colab import drive
import os, sys, itertools
import pandas as pd
from sklearn.model_selection import train_test_split
from PIL import Image

import torch
from torch.utils.data import Dataset

from datasets import load_dataset
import transformers
from transformers import Seq2SeqTrainingArguments, Seq2SeqTrainer
from transformers import VisionEncoderDecoderModel, TrOCRProcessor, default_data_co
import evaluate
```

```
Requirement already satisfied: pip in /usr/local/lib/python3.10/dist-packages
Collecting pip
  Downloading pip-24.3.1-py3-none-any.whl.metadata (3.7 kB)
Downloading pip-24.3.1-py3-none-any.whl (1.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 55.0 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.1.2
    Uninstalling pip-24.1.2:
      Successfully uninstalled pip-24.1.2
Successfully installed pip-24.3.1
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-
Requirement already satisfied: torchaudio in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/pyth
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/di
Collecting fsspec==2024.6.1
  Downloading fsspec-2024.6.1-py3-none-any.whl.metadata (11 kB)
Downloading fsspec-2024.6.1-py3-none-any.whl (177 kB)
Installing collected packages: fsspec
```

```
Installing collected packages: fsspec
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2024.10.0
    Uninstalling fsspec-2024.10.0:
      Successfully uninstalled fsspec-2024.10.0
ERROR: pip's dependency resolver does not currently take into account all the
gcsfs 2024.10.0 requires fsspec==2024.10.0, but you have fsspec 2024.6.1 whicl
Successfully installed fsspec-2024.6.1
Collecting datasets==3.0.0
  Downloading datasets-3.0.0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.10/di
Collecting dill<0.3.9,>=0.3.0 (from datasets==3.0.0)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.10/d
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.10/dist-
Collecting xxhash (from datasets==3.0.0)
  Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x8(
Collecting multiprocess (from datasets==3.0.0)
  Downloading multiprocess-0.70.17-py310-none-any.whl.metadata (7.2 kB)
Requirement already satisfied: fsspec<=2024.6.1,>=2023.1.0 in /usr/local/lib/p
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: huggingface-hub>=0.22.0 in /usr/local/lib/pythc
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/pythc
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/d
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/local/lib/pythc
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist
```

```
# Environment info
print("Python:".rjust(15), sys.version[0:6])
print("Pandas:".rjust(15), pd.__version__)
print("Transformers:".rjust(15), transformers.__version__)
print("Torch:".rjust(15), torch.__version__)
```

```
         Python: 3.10.1
         Pandas: 2.2.2
   Transformers: 4.46.3
          Torch: 2.5.1+cu121
```

```
# Mount Google Drive
drive.mount('/content/drive', force_remount=True)
path = '/content/drive/My Drive/CMPE 252 Project/whiteplate_normal/'
```

```
Mounted at /content/drive
```

```
# Dataset Preparation
file_names, texts = [], []
for file in os.listdir(path):
    if file.endswith(('.jpg', '.png')):
```

```
    if file.endswith(('.jpg', '.png')):
        file_names.append(file)
        texts.append(os.path.splitext(file)[0])

dataset = pd.DataFrame({'file_name': file_names, 'text': texts})
train_dataset, test_dataset = train_test_split(dataset, train_size=0.80, random_s
train_dataset.reset_index(drop=True, inplace=True)
test_dataset.reset_index(drop=True, inplace=True)
```

```
class License_Plates_OCR_Dataset(Dataset):
    def __init__(self, root_dir, df, processor, max_target_length=128):
        self.root_dir = root_dir
        self.df = df
        self.processor = processor
        self.max_target_length = max_target_length

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        file_name = self.df['file_name'][idx]
        text = self.df['text'][idx]
        image = Image.open(self.root_dir + file_name).convert("RGB")
        pixel_values = self.processor(image, return_tensors="pt").pixel_values
        labels = self.processor.tokenizer(text, padding="max_length", max_length=
        labels = [label if label != self.processor.tokenizer.pad_token_id else -1
        return {"pixel_values": pixel_values.squeeze(), "labels": torch.tensor(la
```

```
!huggingface-cli login
```

```
    _|    _|  _|    _|    _|_|_|    _|_|_|  _|_|_|  _|      _|    _|_|_|
    _|    _|  _|    _|  _|        _|          _|    _|_|    _|  _|
    _|_|_|_|  _|    _|  _|  _|_|  _|  _|_|    _|    _|  _|  _|  _|  _|_|
    _|    _|  _|    _|  _|    _|  _|    _|    _|    _|    _|_|  _|    _|
    _|    _|    _|_|      _|_|_|    _|_|_|  _|_|_|  _|      _|    _|_|_|

    To log in, `huggingface_hub` requires a token generated from https://hugg
Enter your token (input will not be visible):
Add token as git credential? (Y/n) n
Token is valid (permission: fineGrained).
The token `testingLicensePlate` has been saved to /root/.cache/huggingface/st
Your token has been saved to /root/.cache/huggingface/token
Login successful.
The current active token is: `testingLicensePlate`
```

```
# Model Initialization
MODEL_CKPT = "microsoft/trocr-base-printed"
processor = TrOCRProcessor.from_pretrained(MODEL_CKPT)
train_ds = License_Plates_OCR_Dataset(path, train_dataset, processor)
```

```
test_ds = License_Plates_OCR_Dataset(path, test_dataset, processor)

model = VisionEncoderDecoderModel.from_pretrained(MODEL_CKPT)
model.config.decoder_start_token_id = processor.tokenizer.cls_token_id
model.config.pad_token_id = processor.tokenizer.pad_token_id
model.config.eos_token_id = processor.tokenizer.sep_token_id
model.config.max_length = 64
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: Use
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access pu
  warnings.warn(
```

preprocessor_config.json: 100% 224/224 [00:00<00:00, 18.4kB/s]

tokenizer_config.json: 100% 1.12k/1.12k [00:00<00:00, 104kB/s]

vocab.json: 100% 899k/899k [00:00<00:00, 1.39MB/s]

merges.txt: 100% 456k/456k [00:00<00:00, 1.07MB/s]

special_tokens_map.json: 100% 772/772 [00:00<00:00, 67.0kB/s]

config.json: 100% 4.13k/4.13k [00:00<00:00, 334kB/s]

model.safetensors: 100% 1.33G/1.33G [00:05<00:00, 224MB/s]

```
Config of the encoder: <class 'transformers.models.vit.modeling_vit.ViTModel'>
  "attention_probs_dropout_prob": 0.0,
  "encoder_stride": 16,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.0,
  "hidden_size": 768,
  "image_size": 384,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "model_type": "vit",
  "num_attention_heads": 12,
  "num_channels": 3,
  "num_hidden_layers": 12,
  "patch_size": 16,
  "qkv_bias": false,
  "transformers_version": "4.46.3"
}

Config of the decoder: <class 'transformers.models.trocr.modeling_trocr.TrOCR
```

```
      "activation_dropout": 0.0,
      "activation_function": "gelu",
      "add_cross_attention": true,
      "attention_dropout": 0.0,
      "bos_token_id": 0,
      "classifier_dropout": 0.0,
      "cross_attention_hidden_size": 768,
      "d_model": 1024,
      "decoder_attention_heads": 16,
      "decoder_ffn_dim": 4096,
      "decoder_layerdrop": 0.0,
      "decoder_layers": 12,
      "decoder_start_token_id": 2,
      "dropout": 0.1,
      "eos_token_id": 2,
      "init_std": 0.02,
      "is_decoder": true,
      "layernorm_embedding": true,
      "max_position_embeddings": 512,
      "model_type": "trocr",
      "pad_token_id": 1,
      "scale_embedding": false,
      "transformers_version": "4.46.3",
      "use_cache": false,
      "use_learned_position_embeddings": true,
      "vocab_size": 50265
    }
```


```
# Metrics
cer_metric = evaluate.load("cer")

# Adversarial Patch Attack to Maximize CER
def overlay_patch(image_tensor, patch):
    """Overlay patch on center of the image."""
    patched_image = image_tensor.clone()
    patch_height, patch_width = patch.shape[1:]
    center_y = (patched_image.shape[1] - patch_height) // 2
    center_x = (patched_image.shape[2] - patch_width) // 2
    patched_image[:, center_y:center_y + patch_height, center_x:center_x + patch_
    return patched_image

def fast_gradient_sign_patch(image_tensor, true_label, processor, model, cer_metr
                             epochs=10, lr=0.01):
    """
    Generate an adversarial patch to fool the OCR model, ensuring device consiste
    """
    # Ensure all computations happen on the same device
    device = model.device
    image_tensor = image_tensor.to(device)
    patch = torch.rand((3, *patch_size), device=device, requires_grad=True)
    optimizer = torch.optim.Adam([patch], lr=lr)
```

```
    # Tokenize the true label and move it to the same device
    true_ids = processor.tokenizer(true_label, return_tensors="pt").input_ids.to(

    for epoch in range(epochs):
        optimizer.zero_grad()

        # Overlay the patch on the image
        patched_image = overlay_patch(image_tensor, patch)

        # Forward pass through the model
        outputs = model(patched_image.unsqueeze(0), labels=true_ids)

        # Use cross-entropy loss
        loss = outputs.loss
        loss.backward()
        optimizer.step()

        # Clamp patch values to ensure valid pixel range
        patch.data = torch.clamp(patch.data, 0, 1)

        # Compute CER (for logging purposes)
        predictions = processor.batch_decode(outputs.logits.argmax(dim=-1), skip_
        cer = cer_metric.compute(predictions=predictions, references=[true_label]

        print(f"Epoch {epoch + 1}/{epochs}, Loss: {loss.item():.4f}, CER: {cer:.4

    return patch
```

Downloading builder script: 100%  5.60k/5.60k [00:00<00:00, 456kB/

```
# Evaluate on Test Image
true_label = test_dataset.iloc[0]['text']
image_path = path + test_dataset.iloc[0]['file_name']

# Load and preprocess the image
image = Image.open(image_path).convert("RGB")
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Move processor-preprocessed tensor to the same device
image_tensor = processor(image, return_tensors="pt").pixel_values[0].to(device)

# Ensure the model is on the same device
model = model.to(device)

# Generate Adversarial Patch
adversarial_patch = fast_gradient_sign_patch(
```

```
    image_tensor, true_label, processor, model, cer_metric,
    patch_size=(100, 100), epochs=10, lr=0.01
)

# Apply the adversarial patch to the image
patched_image = overlay_patch(image_tensor, adversarial_patch)

# Ensure patched_image is on the correct device and has batch dimension
patched_image = patched_image.unsqueeze(0).to(device)

# Evaluate the model on the patched image
model.eval()  # Set model to evaluation mode
with torch.no_grad():  # Disable gradient computation
    outputs = model.generate(patched_image)  # Use generate for inference

# Decode predictions and compute CER
patched_predictions = processor.batch_decode(outputs, skip_special_tokens=True)
patched_cer = cer_metric.compute(predictions=patched_predictions, references=[tru

# Print results
print(f"True Label: {true_label}")
print(f"Predicted Label (Patched): {patched_predictions}")
print(f"CER on Patched Image: {patched_cer:.4f}")
```

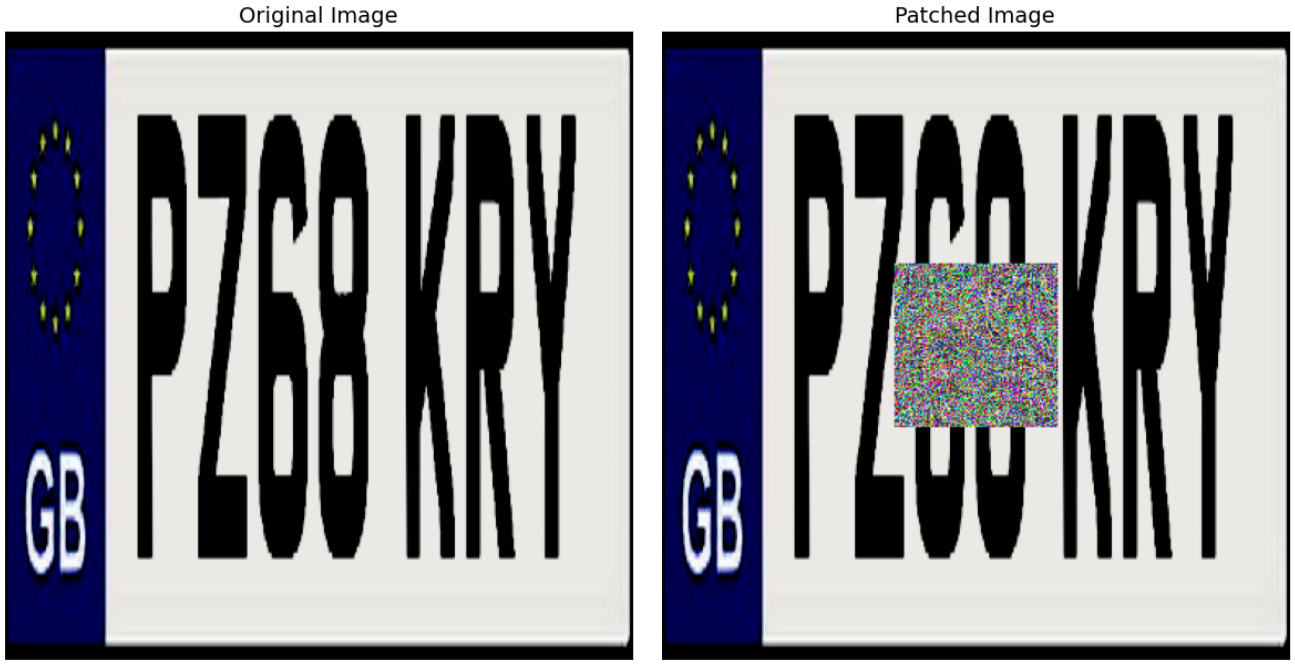
```
Epoch 1/10, Loss: 6.1174, CER: 0.8571
Epoch 2/10, Loss: 5.8741, CER: 0.8571
Epoch 3/10, Loss: 5.7155, CER: 0.7143
Epoch 4/10, Loss: 5.6127, CER: 0.7143
Epoch 5/10, Loss: 5.5154, CER: 0.7143
Epoch 6/10, Loss: 5.4110, CER: 0.5714
Epoch 7/10, Loss: 5.3069, CER: 0.5714
Epoch 8/10, Loss: 5.2190, CER: 0.5714
Epoch 9/10, Loss: 5.1339, CER: 0.5714
Epoch 10/10, Loss: 5.0390, CER: 0.4286
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1493
  warnings.warn(
True Label: PZ68KRY
Predicted Label (Patched): ['@8 PZCSKRY']
CER on Patched Image: 0.7143
```

```
import matplotlib.pyplot as plt

def show_original_and_patched(original_image_tensor, patched_image_tensor, title1
    """
    Displaying the original and patched images side by side
    """
    #Remove batch dimension if it exists
    if len(patched_image_tensor.shape) == 4:  # Shape [1, C, H, W]
        patched_image_tensor = patched_image_tensor.squeeze(0)
    #Tensors to NumPy arrays
```

```
    #Tensors to Numpy arrays
    original_image_np = original_image_tensor.permute(1, 2, 0).cpu().detach().num
    patched_image_np = patched_image_tensor.permute(1, 2, 0).cpu().detach().numpy
    #Plots
    fig, axes = plt.subplots(1, 2, figsize=(12, 6))
    #Original
    axes[0].imshow(original_image_np)
    axes[0].set_title(title1, fontsize=14)
    axes[0].axis("off") #No axes
    #Patched
    axes[1].imshow(patched_image_np)
    axes[1].set_title(title2, fontsize=14)
    axes[1].axis("off") #No axes
    plt.tight_layout()
    plt.show()

show_original_and_patched(image_tensor, patched_image)
```



Api key: 610b65cafc807a2520e2754f9248364c728ef52b

```
# Compute metrics for training and evaluation
def compute_metrics(pred):
    label_ids = pred.label_ids
    pred_ids = pred.predictions

    # Decode predictions and labels
    pred_str = processor.batch_decode(pred_ids, skip_special_tokens=True)
```

```
    pred_str = processor.batch_decode(pred_ids, skip_special_tokens=True)
    label_ids[label_ids == -100] = processor.tokenizer.pad_token_id
    label_str = processor.batch_decode(label_ids, skip_special_tokens=True)

    cer = cer_metric.compute(predictions=pred_str, references=label_str)
    return {"cer": cer}

# Seq2Seq Training Arguments
MODEL_NAME = "trocr_license_plate_model"
NUM_OF_EPOCHS = 2

args = Seq2SeqTrainingArguments(
    output_dir=MODEL_NAME,
    num_train_epochs=NUM_OF_EPOCHS,
    predict_with_generate=True,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    logging_first_step=True,
    hub_private_repo=True,
    push_to_hub=True
)

# Initialize Trainer
trainer = Seq2SeqTrainer(
    model=model,
    tokenizer=processor.feature_extractor,
    args=args,
    compute_metrics=compute_metrics,
    train_dataset=train_ds,
    eval_dataset=test_ds,
    data_collator=default_data_collator
)

# Train the model
print("\n-- Training the Model --")
train_results = trainer.train()
trainer.save_model()
trainer.log_metrics("train", train_results.metrics)
trainer.save_metrics("train", train_results.metrics)
trainer.save_state()

# Evaluate on Clean Test Set
print("\n-- Evaluating on Clean Test Set --")
metrics = trainer.evaluate()
trainer.log_metrics("eval", metrics)
trainer.save_metrics("eval", metrics)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1568: Fu
  warnings.warn(
```

```
/usr/local/lib/python3.10/dist-packages/transformers/models/trocr/processing_
  warnings.warn(
<ipython-input-11-9165111f6ba1>:32: FutureWarning: `tokenizer` is deprecated 
  trainer = Seq2SeqTrainer(
wandb: WARNING The `run_name` is currently set to the same value as `Training

-- Training the Model --
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: http
wandb: You can find your API key in your browser here: https://wandb.ai/autho
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to q
```

Tracking run with wandb version 0.18.7

Run data is saved locally in `/content/wandb/run-20241218_011038-dxr12syx`

Syncing run **trocr_license_plate_model** to Weights & Biases (docs)

View project at https://wandb.ai/sofia-silva-san-jose-state-university/huggingface

View run at https://wandb.ai/sofia-silva-san-jose-state-university/huggingface/runs/dxr12syx

[2406/2406 3:10:55, Epoch 2/2]

Epoch	Training Loss	Validation Loss	Cer
1	0.165700	0.090956	0.005099
2	0.037100	0.033695	0.001601

```
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
Trainer.tokenizer is now deprecated. You should use Trainer.processing_class 
```