HTRやってみたver4

7-22 個別ゼミ 工藤滉青

アウトライン

- コードの理解を進める
- いろんな人が書いた筆記スタイルで精度評価
- 生成したテキストが正しいかどうかの処理をしているのか(入っていたらオフにする)

1.コードの理解を進める

実行コマンド：python -m src train

```python
from .cli import cli

main = cli

if __name__ == "__main__":
    main()
```

__main__.py

```python
import typer
from pathlib import Path

from .main import TrocrPredictor, main_train, main_validate

cli = typer.Typer(name="TrOCR")


@cli.command()
def train(local_model: bool = False):
    main_train(local_model)


@cli.command()
def validate(local_model: bool = True):
    main_validate(local_model)


@cli.command()
def predict(image_paths: list[str], local_model: bool = True):
    # コマンドライン引数から画像パスを受け取り、予測を行う
    predictions = TrocrPredictor(local_model).predict_for_image_paths(image_paths)
    for path, (prediction, confidence) in zip(image_paths, predictions):
        print(f"Path:\t\t{path}\nPrediction:\t{prediction}\nConfidence:\t{confidence}\n")
```

cli.py

1.コードの理解を進める

```python
def main_train(use_local_model: bool = False):
    processor = load_processor()
    train_dataset = HCRDataset(paths.train_dir, processor)
    train_dataloader = DataLoader(train_dataset, constants.batch_size, shuffle=True, num_workers=constants.num_workers)

    val_dataset = HCRDataset(paths.val_dir, processor)
    val_dataloader = DataLoader(val_dataset, constants.batch_size, num_workers=constants.num_workers)

    model = load_model(use_local_model)
    init_model_for_training(model, processor)

    context = Context(model, processor, train_dataset, train_dataloader, val_dataset, val_dataloader)
    train(context, constants.train_epochs)
    debug_print(f"Saving model to {paths.model_path}...")
    model.save_pretrained(paths.model_path)
```

main.py(一部抜粋)

1.コードの理解を進める

processor中身

```
[
  ViTImageProcessor {
    "do_normalize": true,
    "do_rescale": true,
    "do_resize": true,
    "image_mean": [0.5, 0.5, 0.5],
    "image_processor_type": "ViTImageProcessor",
    "image_std": [0.5, 0.5, 0.5],
    "resample": 2,
    "rescale_factor": 0.00392156862745098,
    "size": {
      "height": 384,
      "width": 384
    }
  },
  RobertaTokenizerFast {
    "name_or_path": "microsoft/trocr-base-handwritten",
    "vocab_size": 50265,
    "model_max_length": 512,
    "is_fast": true,
    "padding_side": "right",
    "truncation_side": "right",
    "special_tokens": {
      "bos_token": "<s>",
      "eos_token": "</s>",
      "unk_token": "<unk>",
      "sep_token": "</s>",
      "pad_token": "<pad>",
      "cls_token": "<s>",
      "mask_token": "<mask>"
    },
    "clean_up_tokenization_spaces": true
  }
]
```

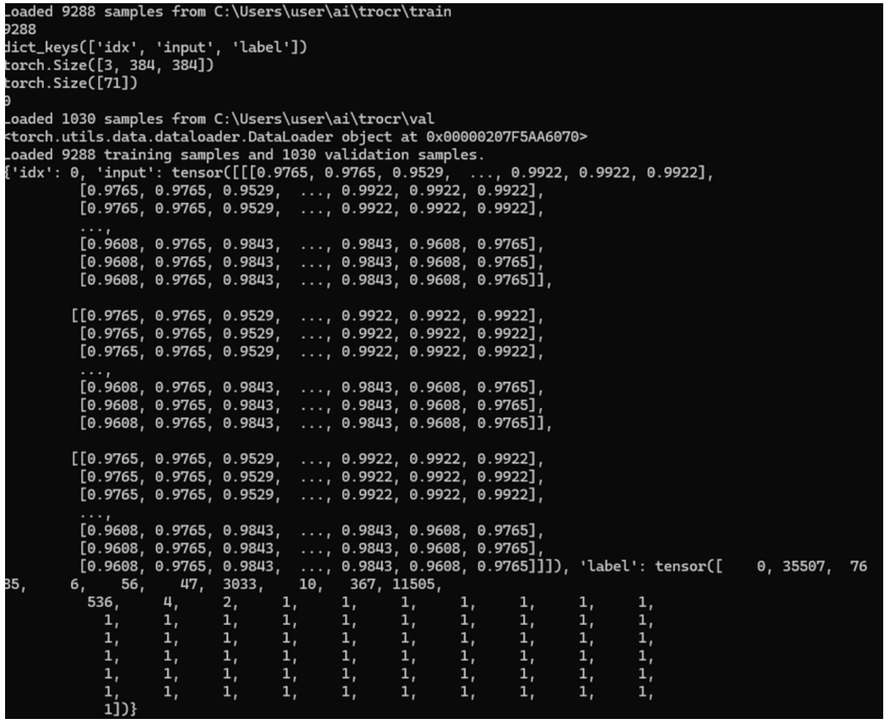
```
TrOCRProcessor:
- image_processor: ViTImageProcessor {
  "do_normalize": true,
  "do_rescale": true,
  "do_resize": true,
  "image_mean": [
    0.5,
    0.5,
    0.5
  ],
  "image_processor_type": "ViTImageProcessor",
  "image_std": [
    0.5,
    0.5,
    0.5
  ],
  "resample": 2,
  "rescale_factor": 0.00392156862745098,
  "size": {
    "height": 384,
    "width": 384
  }
}

- tokenizer: RobertaTokenizerFast(name_or_path='microsoft/trocr-base-handwritten', vocab_size=50265, m
odel_max_length=512, is_fast=True, padding_side='right', truncation_side='right', special_tokens={'bos
_token': AddedToken("<s>", rstrip=False, lstrip=False, single_word=False, normalized=True), 'eos_token
': AddedToken("</s>", rstrip=False, lstrip=False, single_word=False, normalized=True), 'unk_token': Ad
dedToken("<unk>", rstrip=False, lstrip=False, single_word=False, normalized=True), 'sep_token': AddedT
oken("</s>", rstrip=False, lstrip=False, single_word=False, normalized=True), 'pad_token': AddedToken(
"<pad>", rstrip=False, lstrip=False, single_word=False, normalized=True), 'cls_token': AddedToken("<s>
", rstrip=False, lstrip=False, single_word=False, normalized=True), 'mask_token': AddedToken("<mask>",
 rstrip=False, lstrip=True, single_word=False, normalized=True)}, clean_up_tokenization_spaces=True)
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

1.コードの理解を進める

data中身

train_datasetはHCRDatasetのインスタンス化でディレクトリtrain/,val,内の画像読み込みetc

train:9288,val:1030



python コピーする 編集する

```python
train_dataloader = DataLoader(
    train_dataset,                  # → HCRDataset（__getitem__あり）
    constants.batch_size,           # → 1回に取り出すサンプル数（例: 16, 32）
    shuffle=True,                   # → 各エポックでサンプルをランダムに並び替える
    num_workers=constants.num_workers  # → 並列でデータを読み込むワーカープロセス数
)
```

1.コードの理解を進める

model中身

```
VisionEncoderDecoderModel(
  (encoder): ViTModel(
    (embeddings): ViTEmbeddings(
      (patch_embeddings): ViTPatchEmbeddings(
        (projection): Conv2d(3, 768, kernel_size=(16, 16), stride=(16, 16))
      )
      (dropout): Dropout(p=0.0, inplace=False)
    )
    (encoder): ViTEncoder(
      (layer): ModuleList(
        (0): ViTLayer(
```

• • •

```
          )
          (layernorm_before): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (layernorm_after): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        )
      )
    )
    (layernorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (pooler): ViTPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
```

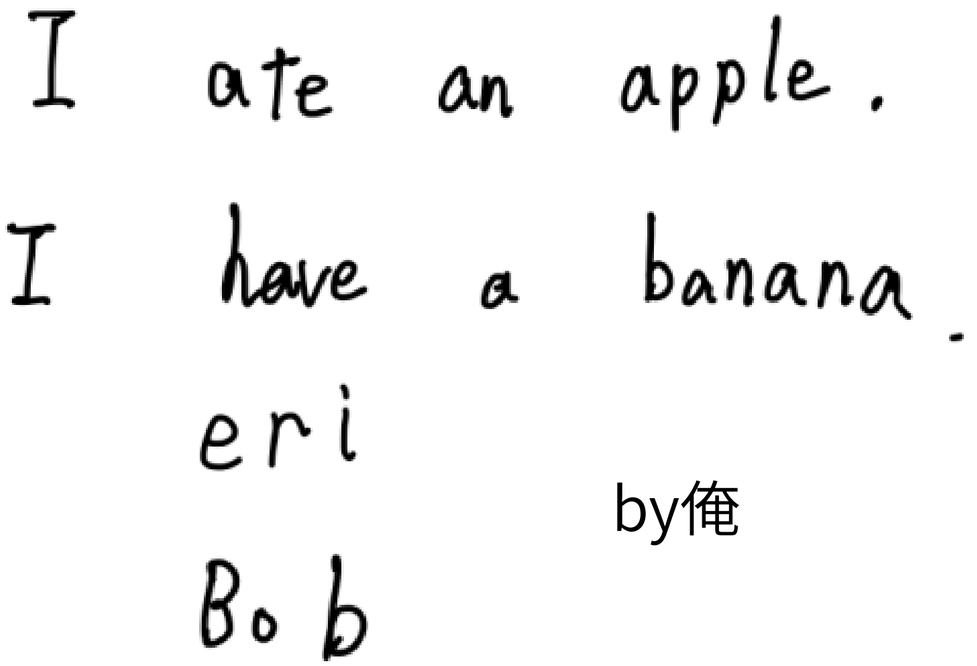
• • •

```
(decoder): TrOCRForCausalLM(
  (model): TrOCRDecoderWrapper(
    (decoder): TrOCRDecoder(
      (embed_tokens): Embedding(50265, 1024, padding_idx=1)
      (embed_positions): TrOCRLearnedPositionalEmbedding(514, 1024)
      (layernorm_embedding): LayerNorm((1024,), eps=1e-05, elementwise_affine=True
      (layers): ModuleList(
        (0): TrOCRDecoderLayer(
          (self_attn): TrOCRAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
```

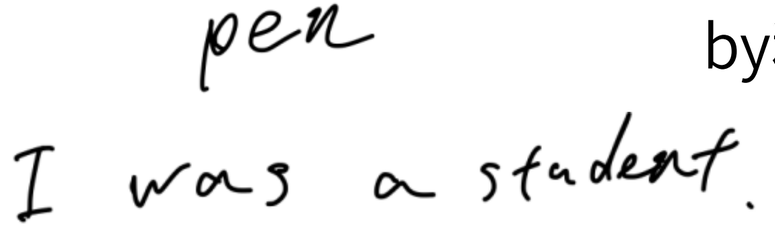
• • •

```
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
      )
    )
    (output_projection): Linear(in_features=1024, out_features=50265, bias=False)
  )
)
```

2.いろんな人が書いた筆記スタイルで精度評価

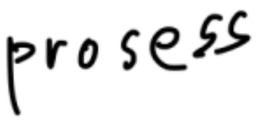


by俺



by和住さん

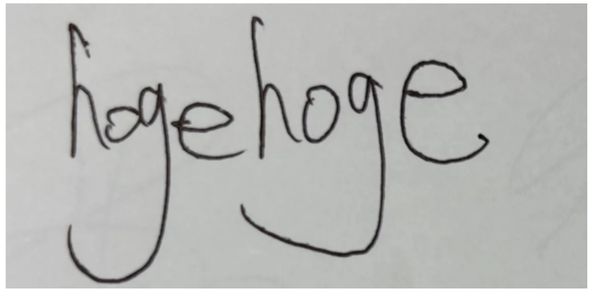
byきよや

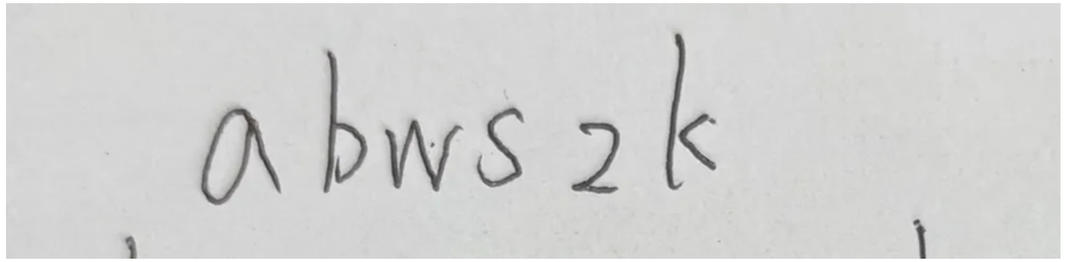


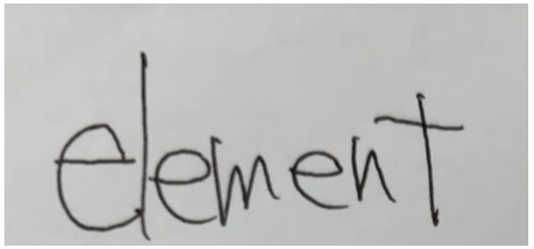
by岡本さん

14	ata\test\image3.png	I ate an apple.
15	ata\test\image4.png	I. have a banana,
16	ata\test\image5.png	er I
17	ata\test\image6.png	Boe'_br_br_br_br_gr_gr_gr_
18	ata\test\image7.png	pen
19	ata\test\image8.png	I was a student.
20	ata\test\image9.png	hope
21	ata\test\imagee.png	I
22	ata\test\imageeee.png	I like industry

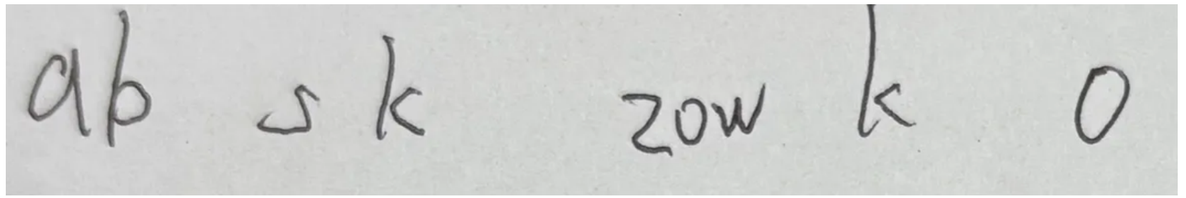
2.いろんな人が書いた筆記スタイルで精度評価







byきよや



by俺

	column 1	column 2	column 3
1	image_path	prediction	confidence
2	data\test\text_htr\imagee1.png	lugehope	0.5140115618705 75
3	data\test\text_htr\imagee2.png	element	0.910437822341919
4	data\test\text_htr\imagee3.png	, abwszk,,,,	0.5754343867301941
5	data\test\text_htr\imagee4.png	Ab.s k-zow k-O	0.26034966111183167

3.生成したテキストが正しいかどうかの処理をしているのか(入っていたらオフにする)

```python
def predict(
    processor: TrOCRProcessor, model: VisionEncoderDecoderModel, dataloader: DataLoader
) -> tuple[list[tuple[int, str]], list[float]]:
    output: list[tuple[int, str]] = []
    confidence_scores: list[tuple[int, float]] = []

    with torch.no_grad():
        model.eval()
        for i, batch in enumerate(dataloader):
            debug_print(f"Predicting batch {i+1}")
            inputs: torch.Tensor = batch["input"].to(constants.device)

            generated_ids = model.generate(inputs, return_dict_in_generate=True, output_scores = True)
            generated_text = processor.batch_decode(generated_ids.sequences, skip_special_tokens=True)

            ids = [t.item() for t in batch["idx"]]
            output.extend(zip(ids, generated_text))

            # Compute confidence scores
            batch_confidence_scores = get_confidence_scores(generated_ids)
            confidence_scores.extend(zip(ids, batch_confidence_scores))

    return output, confidence_scores
```

```python
def get_confidence_scores(generated_ids) -> list[float]:
    # Get raw logits, with shape (examples,tokens,token_vals)
    logits = generated_ids.scores
    logits = torch.stack(list(logits),dim=1)

    # Transform logits to softmax and keep only the highest (chosen) p for each token
    logit_probs = F.softmax(logits, dim=2)
    char_probs = logit_probs.max(dim=2)[0]

    # Only tokens of val>2 should influence the confidence. Thus, set probabilities to 1 for tokens 0-2
    mask = generated_ids.sequences[:,:-1] > 2
    char_probs[mask] = 1

    # Confidence of each example is cumulative product of token probs
    batch_confidence_scores = char_probs.cumprod(dim=1)[:, -1]
    return [v.item() for v in batch_confidence_scores]
```

- 文法的正確性
- 単語の妥当性
- 文脈の一貫性

上記のような処理はしてなさそう

単純にモデルの出力ロジットを単語に置き換えてるっぽい

4.今後の展望

OCRなどのファインチューニング用に、日本語文が入ったテキストファイル（例：1行1文 × 10,000行）を入れてみて日本語htrモデルにしてみる？？できるのか

画像が中央に来ていないときが精度悪いのかも？？
もっと回りきたない場合は？？

二行以降もできるようにした方がよき？？

trocrの論文読んでみた

TrOCRモデルでは、エンコーダの初期化に事前学習済みのDeiT（Touvronら、2021）およびBEiT
（Bao、Dong、Wei、2021）モデルが使用