Visual Question Answering (VQA) モデルの実装と改良

1. はじめに

本レポートでは、Visual Question Answering (VQA) タスクのためのモデルの実装と、その性能向上のために行った一連の改良について詳述する。

1.1 実行環境

今回はゲーム用に所持していた、RTX3070を使って学習を行った。また、ベースラインから追加で実装した部分はsrc_v2に集約している。

2. モデルの改良点

2.1 Transformerを用いたテキストエンコーダー

テキスト処理の性能向上のため、事前学習済みのTransformerモデルをテキストエンコーダーとして採用した。具体的には、Hugging Faceのライブラリから"bert-base-uncased"モデルを使用し、以下のように実装した。

src_v2/models/base.py

```
from transformers import AutoModel, AutoTokenizer

class VQAModel(nn.Module):
    def __init__(self, n_answer: int, pretrained_model_name: str = "bert-base-
uncased"):
        super().__init__()
        self.text_encoder = AutoModel.from_pretrained(pretrained_model_name)
        self.tokenizer = AutoTokenizer.from_pretrained(pretrained_model_name)
        # ... (以下省略)
```

以上をベースとし、ファインチューニングを行う形で実装した。

2.2 class_map.csvを用いた正解ラベルの拡充

モデルの出力候補を拡張し、より多様な回答を可能にするため、外部のclass_map.csvファイルを利用した。このファイルには、可能な回答とそれに対応するクラスIDが記載されている。

src_v2/datasets.py

```
import pandas as pd

class_mapping = pd.read_csv('class_mapping.csv')
answer_space = class_mapping['answer'].tolist()
answer2idx = {answer: idx for idx, answer in enumerate(answer_space)}
```

この実装により、訓練データに存在しない回答も出力できるようになった。

2.3 画像データのData Augmentation

画像データの多様性を増やし、モデルの汎化性能を向上させるため、以下のようなData Augmentationを実装した。

```
train_transform = transforms.Compose([
    transforms.Resize((256, 256)),
    transforms.RandomCrop((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),
    transforms.ToTensor(),
])
```

2.4 Early Stoppingの実装

過学習を防ぎ、最適なモデルを選択するため、Early Stoppingを実装した。具体的には、検証損失が改善しなくなった場合に学習を早期に終了させる仕組みを導入した。この際、trainデータからさらにvalidationデータを切り出す処理も実装した。

```
patience = 10
best_val_loss = float('inf')
counter = 0
best_model = None

for epoch in range(num_epoch):
    # ... (学習処理)
    if val_loss < best_val_loss:
        best_val_loss = val_loss
        counter = 0
        best_model = model.state_dict()
    else:
        counter += 1
        if counter >= patience:
            print(f"Early stopping triggered after {epoch + 1} epochs")
            break
```

この実装により、適切なエポック数で学習を止めることができるようになった。

3. 環境設定

ローカルGPUを効率的に利用するため、以下のコマンドを用いて適切なバージョンのPyTorchをインストールした。

```
pip install torch==1.11.0+cu113 torchvision==0.12.0+cu113 torchaudio==0.11.0 --
extra-index-url https://download.pytorch.org/whl/cu113
```

GPUの種類、CUDAのバージョン、Pythonのバージョン、pytorchやその他ライブラリのバージョンをうまく揃えるのにかなりの時間を要してしまった。

4. 結論

本実装では、Transformerを用いたテキストエンコーダー、拡張された回答空間、Data Augmentation、そしてEarly Stoppingという一連の改良を加えることで、VQAモデルの性能向上を図った。これらの改良により、これらの改良により、ベースラインからは`0.03`の精度向上を達成した。（0.44 → 0.48）