# 進捗報告

## 1 今週やったこと

- Transformers で漢字の画生成モデルによる改善
- 漢字ベクトルの生成手法による調査

## 2 Transformers による灯謎から漢字の画の生成実験

GRU での実験の過学習問題は解決できないため, Transformers で実験を行ってみました.

数回の実験を行った結果,Epoch 数が 30 の時 Train Loss は 0.468 に収束し,Valid Loss は 0.816 に収束し, 過学習問題を改善しました. Test Data について,Precision, Recall, F1 は各自 0.76, 0.74, 0.72 に収束しました. 図 1 に Train Loss と Valid Loss の変化曲線を示します.

図 2 に Transformers の実験用パラメータを示します.

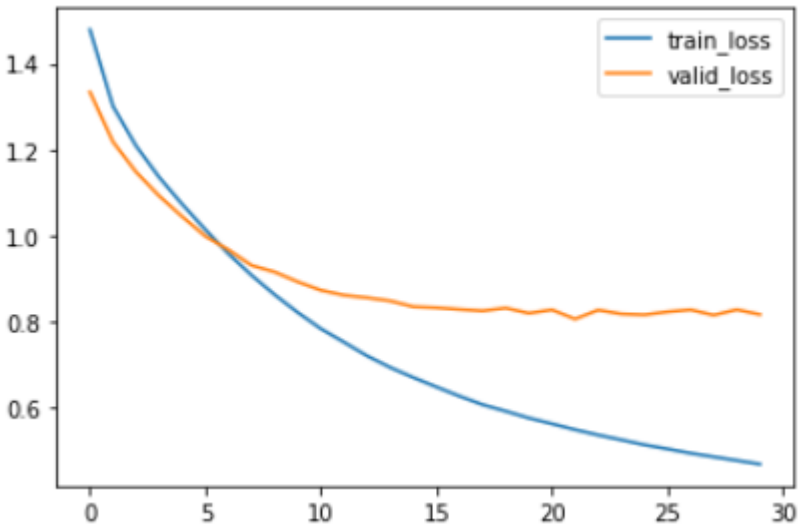


図 1: Transformers による Train Loss と Valid Loss の変化曲線

## 3 漢字ベクトルの生成手法

### 3.1 chinese-roberta-wwm-ext

灯謎問題の自動解答システムの OOV 問題を解決するため, 灯謎の答えの漢字ベクトルと「漢字の画」で生成した漢字ベクトルの類似度の計算で比較します. 故に「Chinese Word Vector」を利用しましたが,「Chinese Word Vector」も低頻度漢字による OOV 問題があります.

```
INPUT_DIM  =  3687
OUTPUT_DIM  =  9
HID_DIM  =  256
ENC_LAYERS  =  3
DEC_LAYERS  =  3
ENC_HEADS  =  8
DEC_HEADS  =  8
ENC_PF_DIM  =  512
DEC_PF_DIM  =  512
ENC_DROPOUT  =  0.1
DEC_DROPOUT  =  0.1
LEARNING_RATE  =  0.0005
optimizer  =  torch.optim.Adam(model.parameters(),  lr  =  LEARNING_RATE)
criterion  =  nn.CrossEntropyLoss(ignore_index  =  TRG_PAD_IDX)
```

図 2: Transformers のパラメータ

### 3.2 BERT

次には学習済み BERT モデル「hfl/chinese-roberta-wwm-ext」を利用しました。このモデルにより, 低頻度単語は全部「UNK」として扱い, かつ問題文で計算すれば, 違う漢字ベクトルを生成できます.

しかし, 続きの「画から漢字ベクトル生成モデル」の Train Loss は 0.555 に止まっています.Learning Rate などのパラメータセッティング問題と想定します.

図 3 に訓練の過程を示します.

```
Epoch: 25 | Epoch Time: 5m 12s
        Train Loss: 0.555
         Val. Loss: 0.555
Epoch: 26 | Epoch Time: 5m 13s
        Train Loss: 0.555
         Val. Loss: 0.555
Epoch: 27 | Epoch Time: 5m 12s
        Train Loss: 0.555
         Val. Loss: 0.555
Epoch: 28 | Epoch Time: 5m 9s
        Train Loss: 0.555
         Val. Loss: 0.555
Epoch: 29 | Epoch Time: 5m 9s
        Train Loss: 0.555
         Val. Loss: 0.555
Epoch: 30 | Epoch Time: 5m 8s
        Train Loss: 0.555
         Val. Loss: 0.555
```

図 3: 漢字ベクトル Train Loss の一部

ChineseBERT を使っていない理由として、プログラムにエラーが発生しましたです. 現在プログラムを改善しています.

### 3.3 AutoEncoder

ChineseBERT で利用された漢字画像データ (サイズ $24 \times 24$) と AutoEncoder で漢字ベクトルを抽出実験を行っています.

図 4 に AutoEncoder モデルの構造を示します.

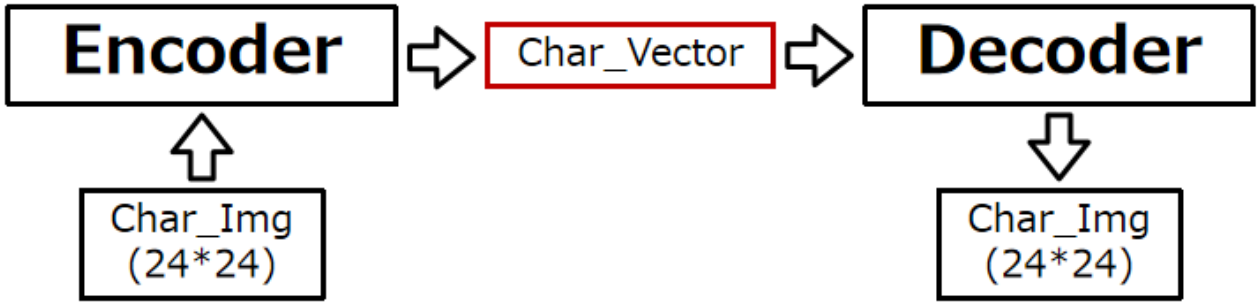


図 4: AutoEncoder モデルの構造

実験結果として,200 Epoch で Train Loss は 0.602 に収集しました. 図 5 に Train Loss の変化曲線を示します.
図 6 に 各 Epoch の出力の例を示します.
図 7 に AutoEncoder の実験用パラメータを示します.

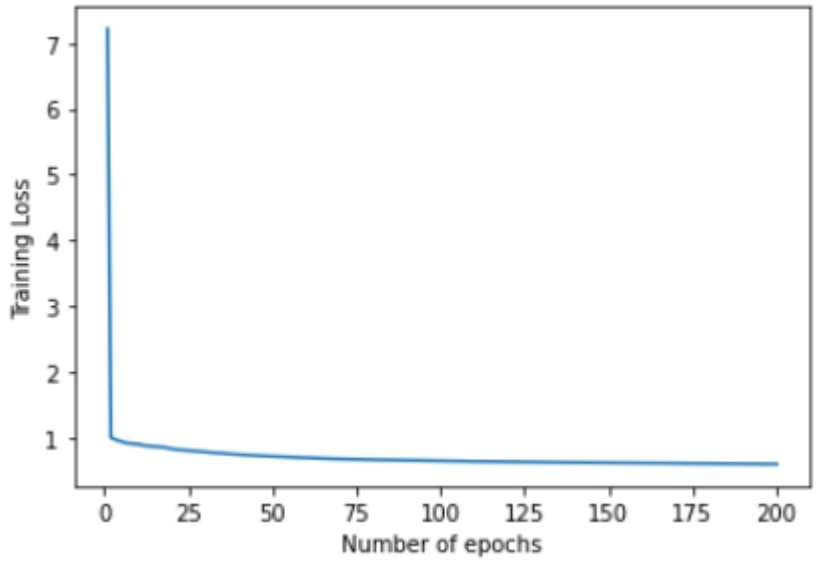


図 5: Train Loss の変化曲線

## 4 対照実験

問題を ChineseBERT に入力し, 答え Token の出力で漢字ベクトルを生成します.

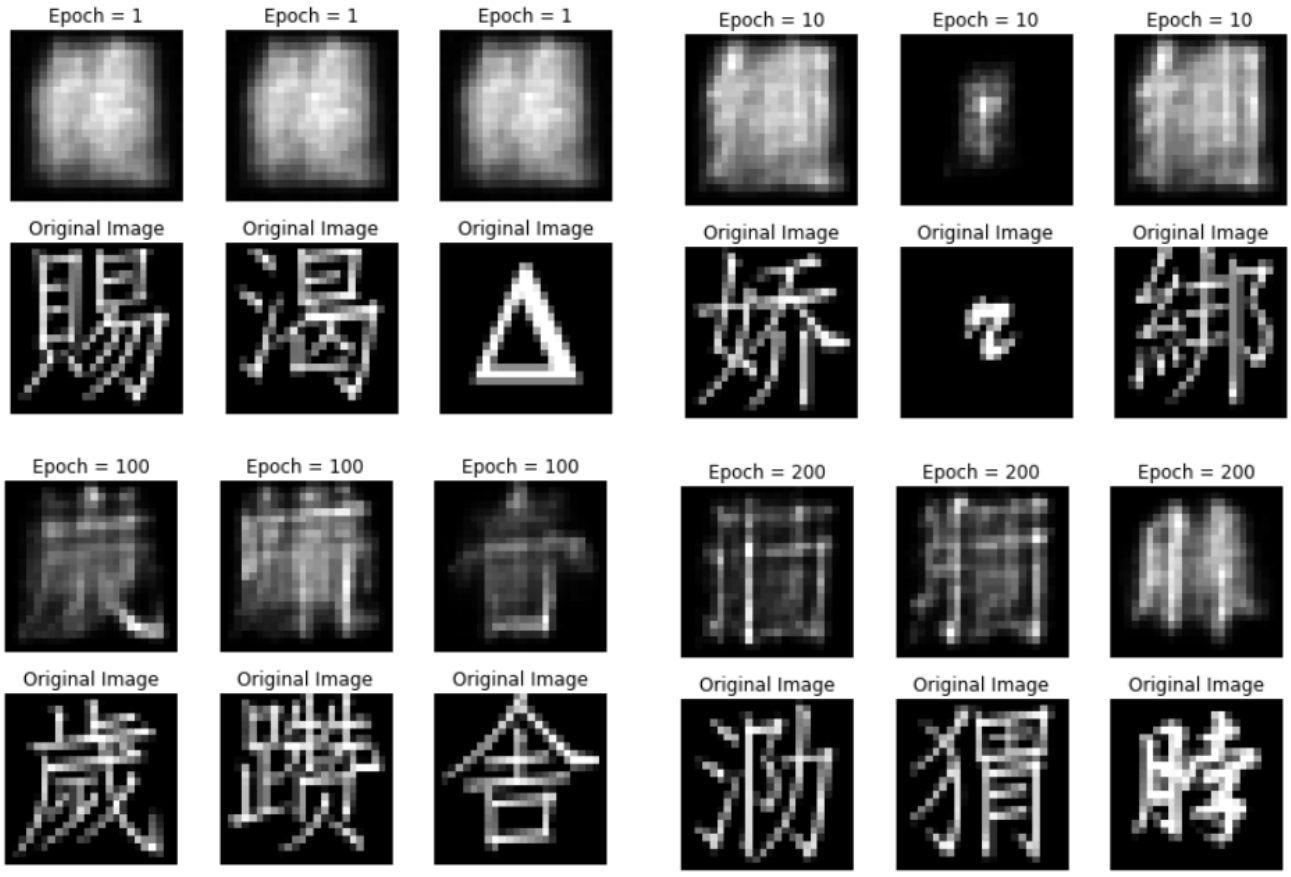


図 6: 各 Epoch の出力の例

```
class DeepAutoencoder(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = torch.nn.Sequential(
            torch.nn.Linear(24 * 24, 256),
            torch.nn.ReLU(),
            torch.nn.Linear(256, 128),
            torch.nn.ReLU(),
            torch.nn.Linear(128, 64),
            torch.nn.ReLU(),
            torch.nn.Linear(64, 10)
        )

        self.decoder = torch.nn.Sequential(
            torch.nn.Linear(10, 64),
            torch.nn.ReLU(),
            torch.nn.Linear(64, 128),
            torch.nn.ReLU(),
            torch.nn.Linear(128, 256),
            torch.nn.ReLU(),
            torch.nn.Linear(256, 24 * 24),
            torch.nn.Sigmoid()
        )

    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded

model = DeepAutoencoder().to(device)
criterion = torch.nn.MSELoss()
num_epochs = 200
optimizer = torch.optim.Adam(model.parameters(), lr=0.0001)
```

図 7: AutoEncoder のパラメータ

この漢字ベクトルと画で生成した漢字ベクトル各自 AutoEncoder で抽出した漢字ベクトルで類似度を計算します.
図 6 に モデル構造を示します.

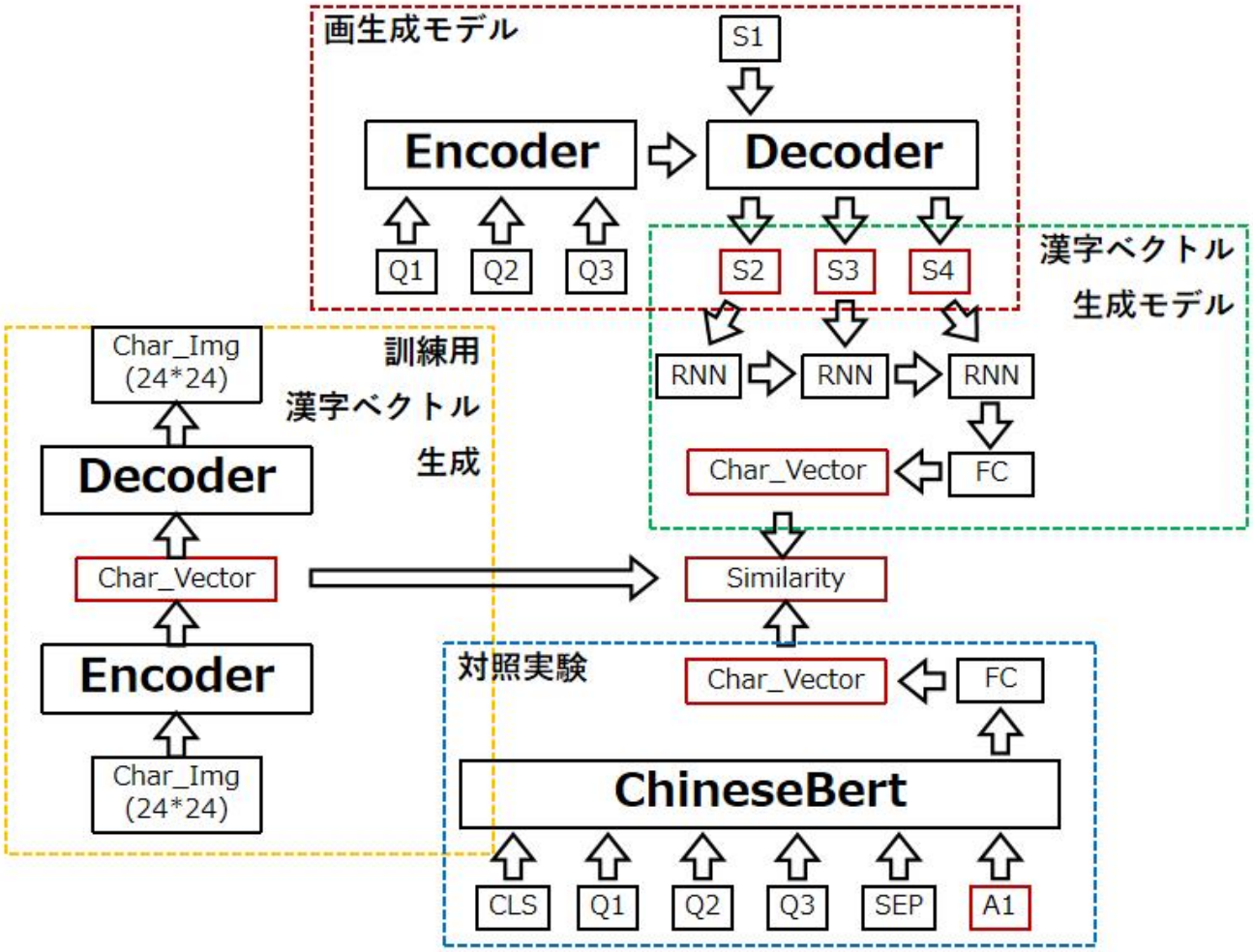


図 8: 対照実験のモデル構造

## 5 来週目標

- Transformers による漢字の画生成モデルの精度向上
- chinese-roberta-wwm-ext 漢字ベクトルによる「画から漢字ベクトル生成モデル」の精度向上
- AutoEncoder で抽出された漢字ベクトルで「画から漢字ベクトル生成実験」の実行
- 対照実験の実施