CLUIT を用いた Realtime Textless VC

Tosaka

2022/01/30

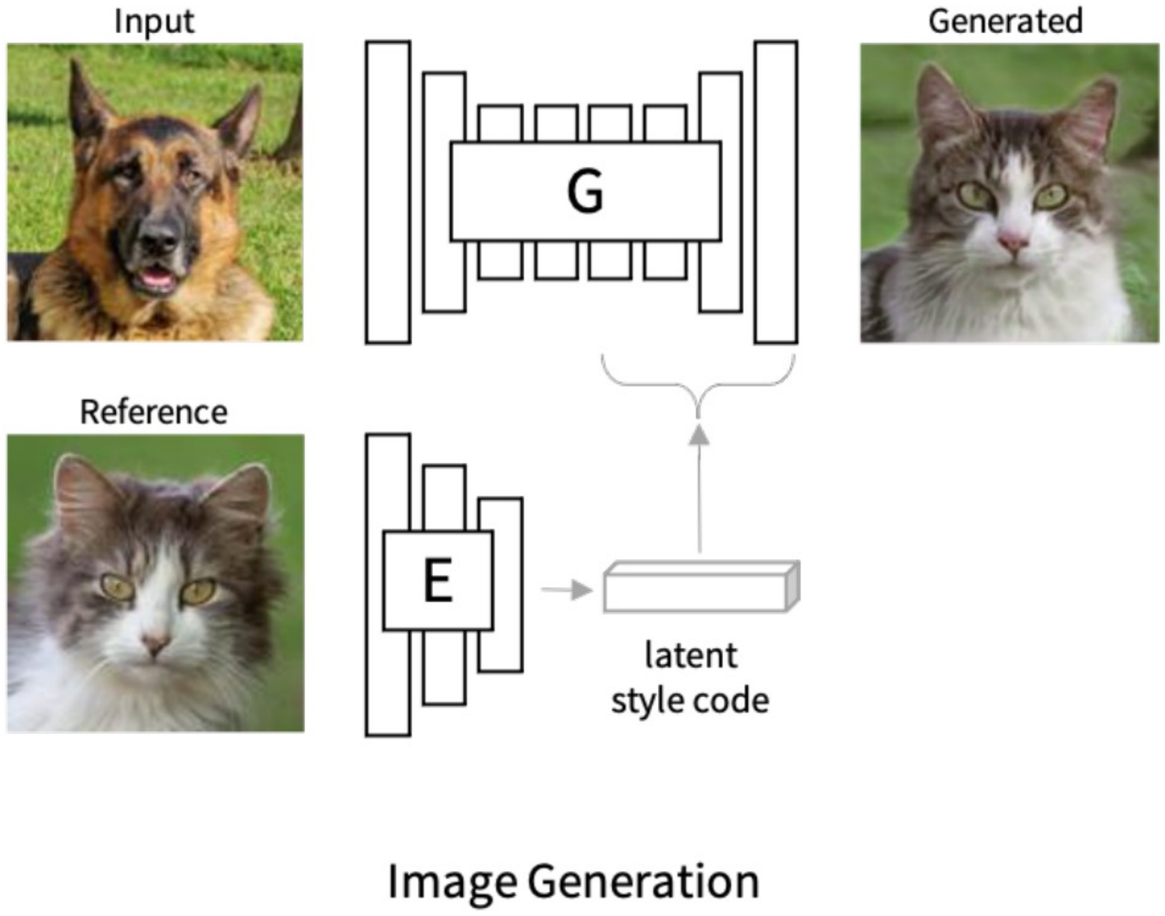
目次

1. CLUITの論文紹介

1.1. 論文概要

タイトル	Contrastive Learning for Unsupervised Image-to-Image Translation
著者	Hanbit Lee, Jinseok Seol, Sang-goo Lee
サマリ	**Contrastive learning** を使用して unsupervised で高精度な image to image を実現 Stargan v2 は domain label を使用しているかつ domain label が粗い場合 reference を無視する問題があるが、こちらでは起きない

1.2. CLUITモデル概要

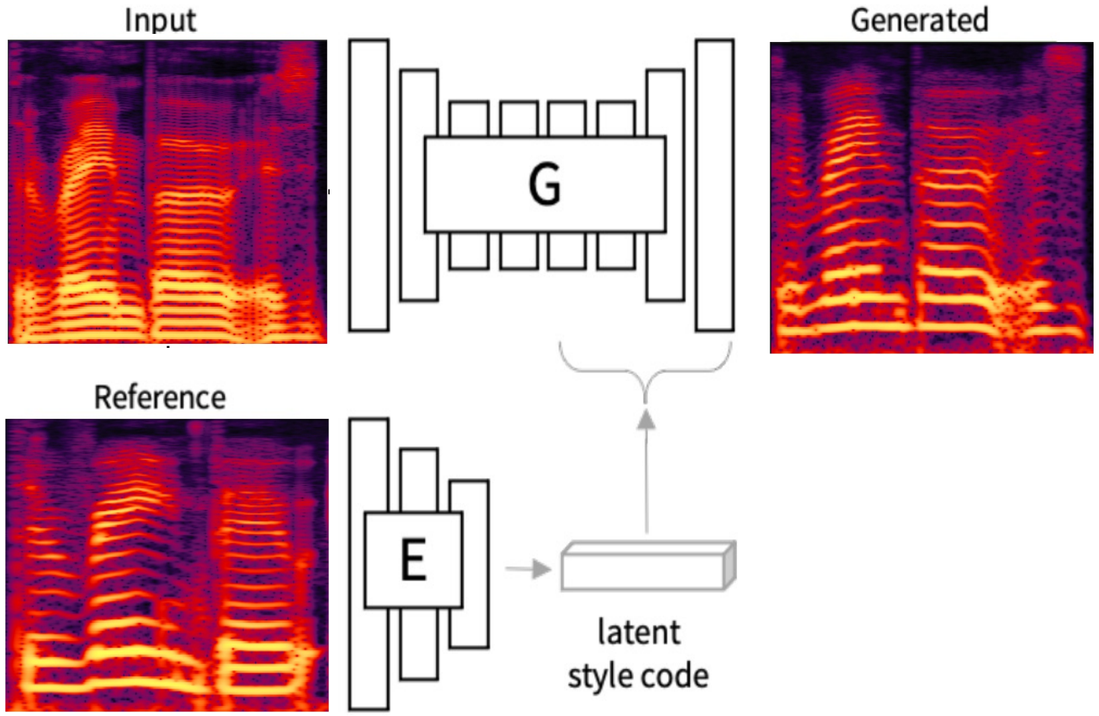


- Encoder-Decoder モデル
- Decoder で reference の style vector を混ぜる
- Stargan v2 とほぼ一緒の構造

Loss 関数に **Contrastive Loss** を使用（後で紹介）

2. VCに適用する方法 & 結果

2.1. VC に適用する場合



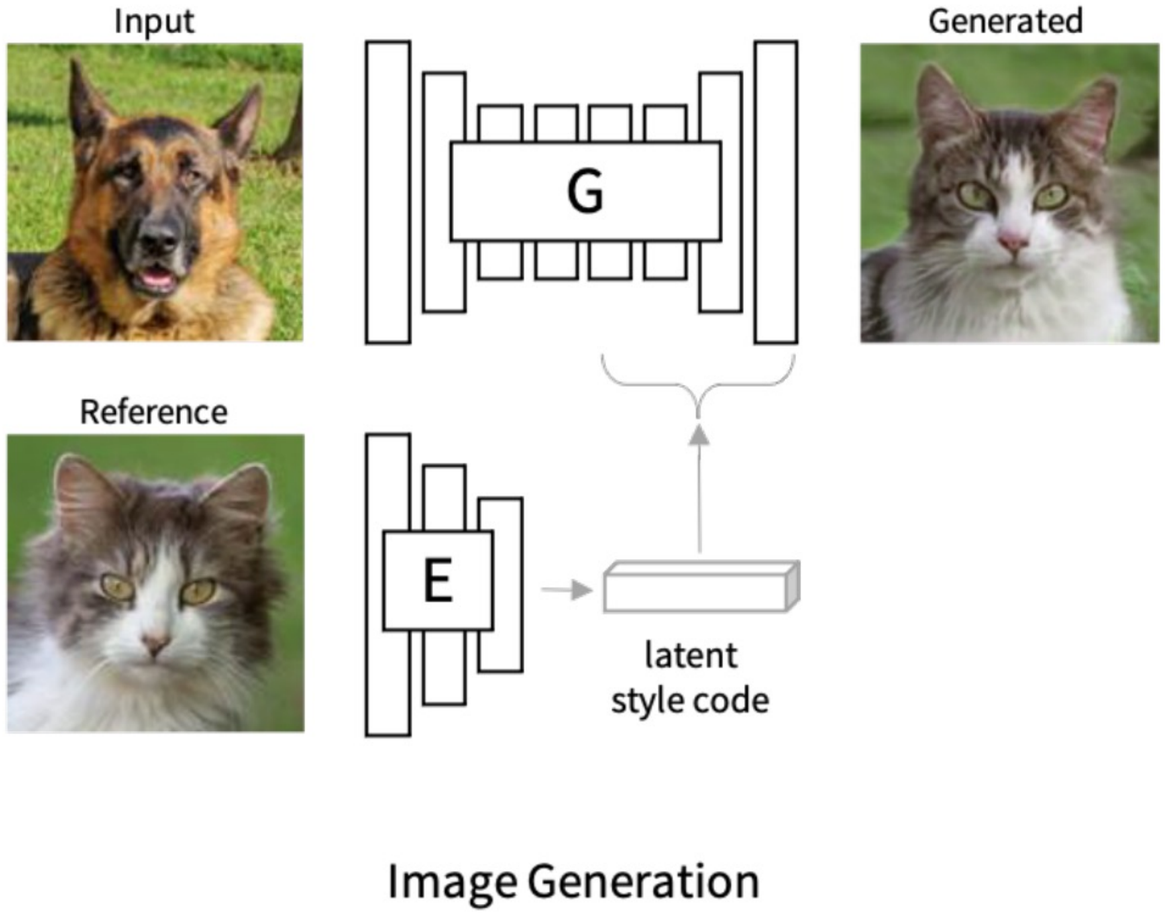
- 単純に画像のとこを mel spectrogram に置き換えるだけ
- 元論文で Conv2d のところを Conv1d に変更
- mel → wave は vocoder を使用
- データ集めで一番大変な音素ラベルが学習にいらない点が良い

2.2. 結果

- ここ https://tosaka-m.github.io/cluitvc.github.io/ にまとめてます
- 使用データ JVS (parallel, nonparallel, whisper, falset 全部)
- RTX3090 で 1.5日くらい
- domain label (speaker label) なしだとあんまりうまくいかなかったので結局入れた

3.手法詳細とか工夫点とか

3.1. 手法詳細

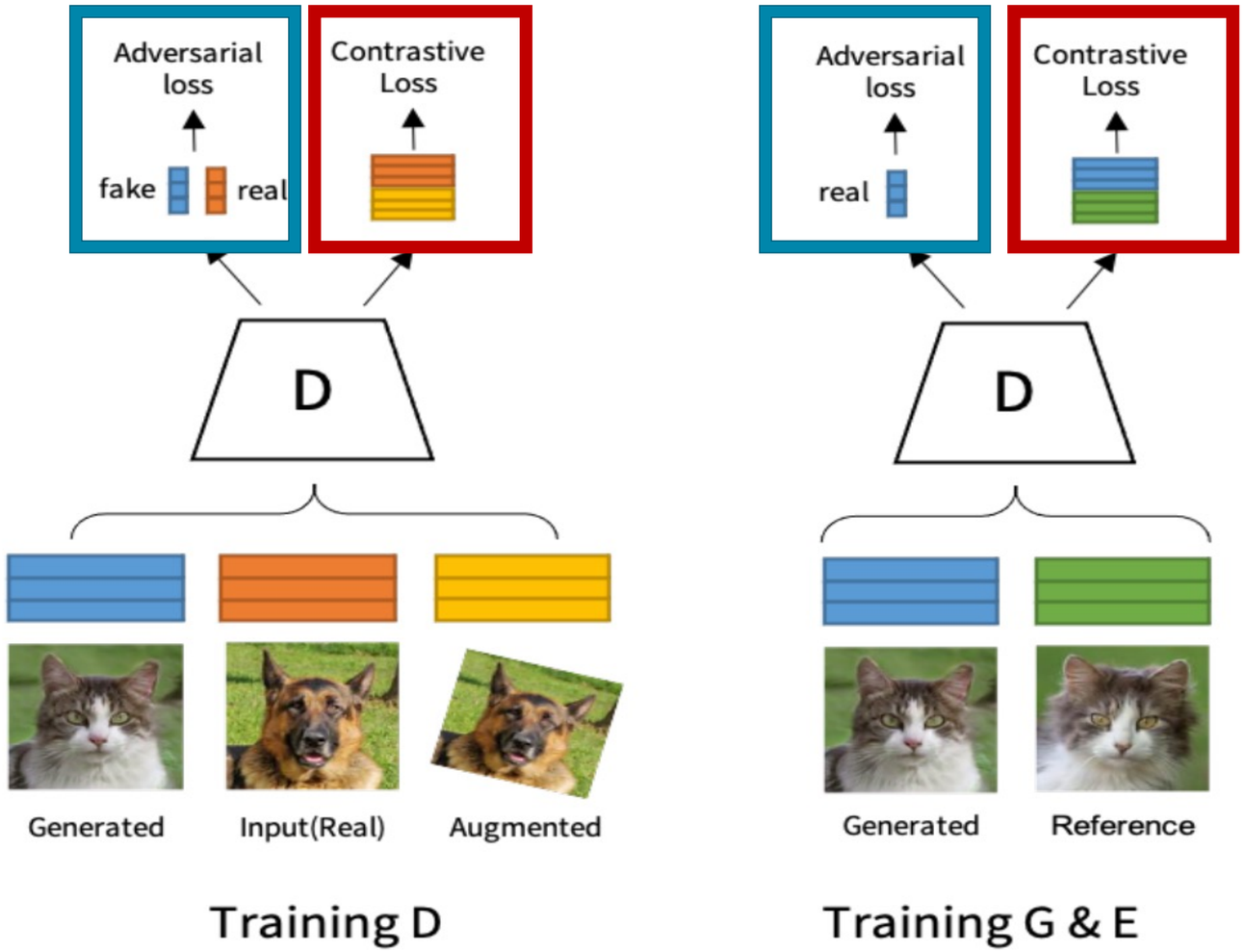


- Loss 関数

$$\mathcal{L}_D = -\mathcal{L}_{\text{adv}} + \lambda_{\text{ct}}^D \mathcal{L}_{\text{ct}}^D,$$

$$\mathcal{L}_{G,E} = \mathcal{L}_{\text{adv}} + \lambda_{\text{cyc}} \mathcal{L}_{\text{cyc}} + \lambda_{\text{ct}}^G \mathcal{L}_{\text{ct}}^G,$$

3.1. 手法詳細

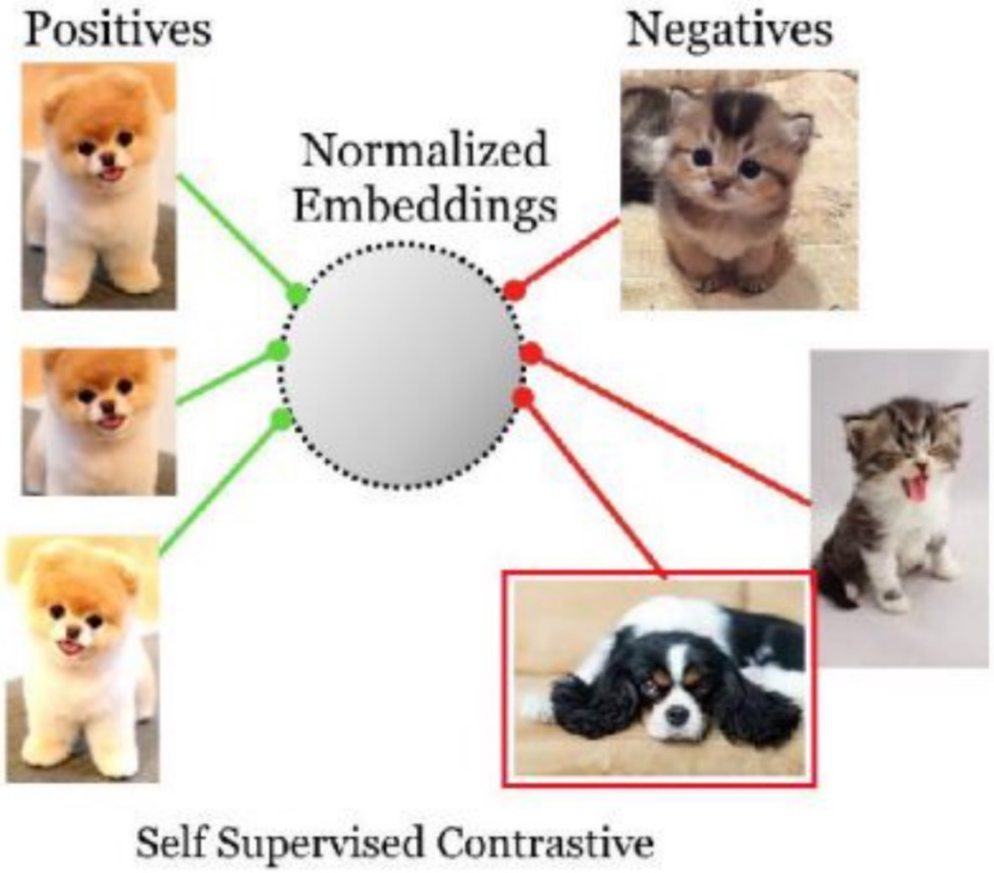


- Loss 関数

$$\mathcal{L}_D = -\mathcal{L}_{\text{adv}} + \lambda_{\text{ct}}^{D}\mathcal{L}_{\text{ct}}^{D},$$

$$\mathcal{L}_{G,E} = \mathcal{L}_{\text{adv}} + \lambda_{\text{cyc}}\mathcal{L}_{\text{cyc}} + \lambda_{\text{ct}}^{G}\mathcal{L}_{\text{ct}}^{G},$$

3.1. 手法詳細



画像: https://towardsdatascience.com/weakly-and-self-supervised-learning-part-4-2fbfd10280b3

Contrastive Loss とは？

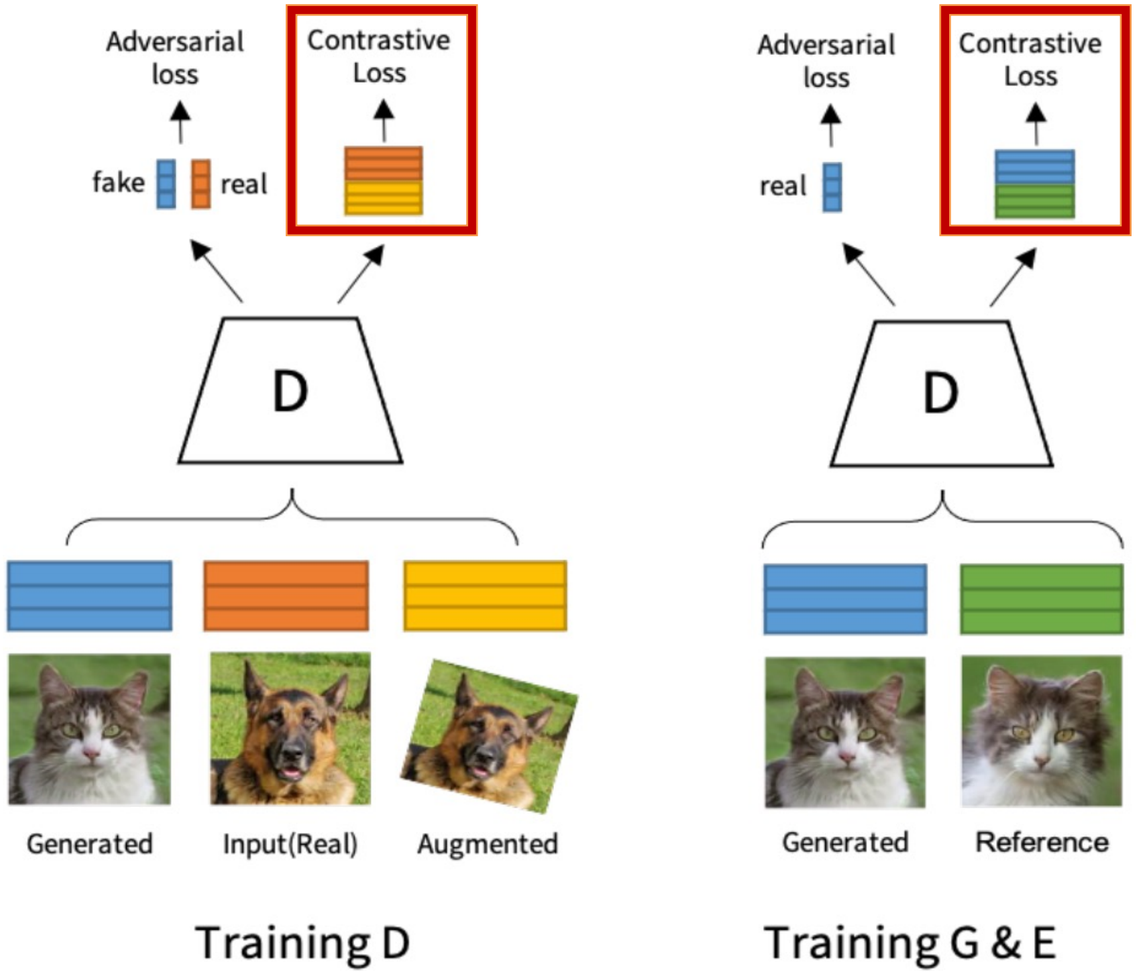
ラベルなしで特徴量を学習させる手法の一種

似ているデータを近づけ異なるデータを遠ざける

似ているデータ = 同じデータの異なる Augmentation

がよく使用される。

3.1. 手法詳細



- Contrastive Learning 部分の loss関数

$$\mathcal{L}_{\mathrm{ct}}^{D} = -\log \frac{\exp(v \cdot v^{+}/\tau)}{\exp(v \cdot v^{+}/\tau) + \sum_{i=1}^{N} \exp(v \cdot v_i^{-}/\tau)}$$

$$\mathcal{L}_{\mathrm{ct}}^{G} = \mathbb{E}\left[-\log \frac{\exp(v_g \cdot v_r/\tau)}{\exp(v_g \cdot v_r/\tau) + \sum_{i=1}^{N} \exp(v_g \cdot v_i^{-}/\tau)} \right]$$

- v_i^- には memory bank (過去使用したデータを保存したもの) からサンプリング
- 途中まで Discriminator の NN を利用している点が特徴的

3.2. VC 適用時に追加した要素

- Norm loss を入れた
  - Norm は mel spec の spec 方向の和(みたいなやつ)で定義
- 結局 domain label (speaker label) は入れた….
  - ラベル一切いらない良さが…

3.3.リアルタイム化するなら

CPU 推論速度

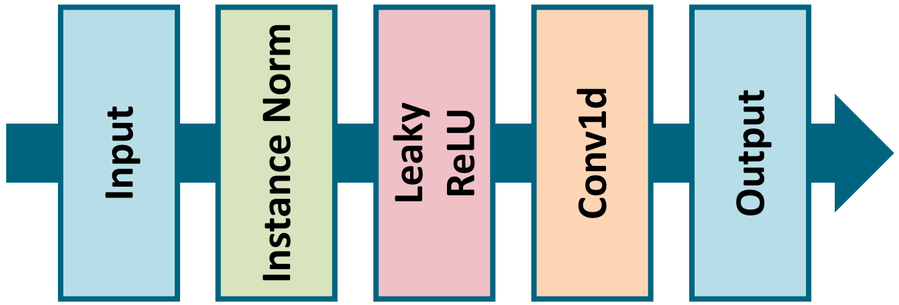
	Total	CLUIT	HifiGANv2
RTF	0.0553	0.0127	0.0426

※CPU=AMD Ryzen 7 3700X 8-Core Processor

- 速度上はリアルタイムっぽいけど…?

3.3. リアルタイム化するなら

Network 構造の一部



- Instance Normalization が全体の情報を使用する
- Convolution も未来の情報を見る
- リアルタイム推論時は細かく切って推論するので基本的にはこれらの情報を使用できない

推論時

4. まとめと今後の発展

4.1. CLUIT を VC に適用してみて

CLUIT の良い点

- Starganv2 よりモデルや学習方法がシンプル（拡張しやすい）
- Contrastive Learning の発展によりこちらもよくなる可能性がある

CLUIT の微妙な点

- Stargan v2 より特別精度は良くない。
- VCに適用した場合だとチューニングが難しい
- 音素も保つ保証がないので更なる工夫が必要 (ASR Loss 入れるとか)

4.2. 今後の発展

1. 別の Contrastive Learning 手法の適用
    1. meta社の の data2vec とか
2. 大規模データで学習
    1. ラベルがいらないので大規模化が容易
    2. ラベルがあるデータはラベルを使う Semi-supervised も良い
3. Wave to Wave にする
    1. mel spec などの中間表現がいらないので精度向上の余地が出る