You’re your own best teacher: A Self-Supervised Learning Approach For Expressive Representations

Johan Vik Mathisen

May 16, 2024

Our work in this thesis can be seen as a tangent of the paper "Vector Quantized Time Series Generation with a Bidirectional Prior Model" [**TimeVQVAE**]. We simplify the model architecture by omitting the high-low frequency split, which reduces the model to what they refer to as "naive TimeVQVAE" in their paper. We expand on naive TimeVQVAE with a self-supervised extension.

The overarching objective in creating our model is to learn more expressive latent representations for better time series generation. We want to improve the reconstruction capabilities of the tokenization model. The rationality is that if the tokenization model reconstructs well the latent representations contains all relevant information of the input. We simultaneously want enforce better class separability in the latent representations, as we hypothesize that such additional structure eases learning of the generative model, both unconditional and conditional generation.

To improve on the reconstruction we add a regularizing term by reconstructing augmented views. We hypothesize that the model generalizes better to unseen data by letting the decoder "see" the augmented views.
To separate classes better we introduce a non contrastive self supervised loss. The intuition being that the representation of original and augmented views are pushed closer together by the SSL loss. We further enforce this hypothesis by using augmentations that preserve the overall semantics of the class conditional distributions.

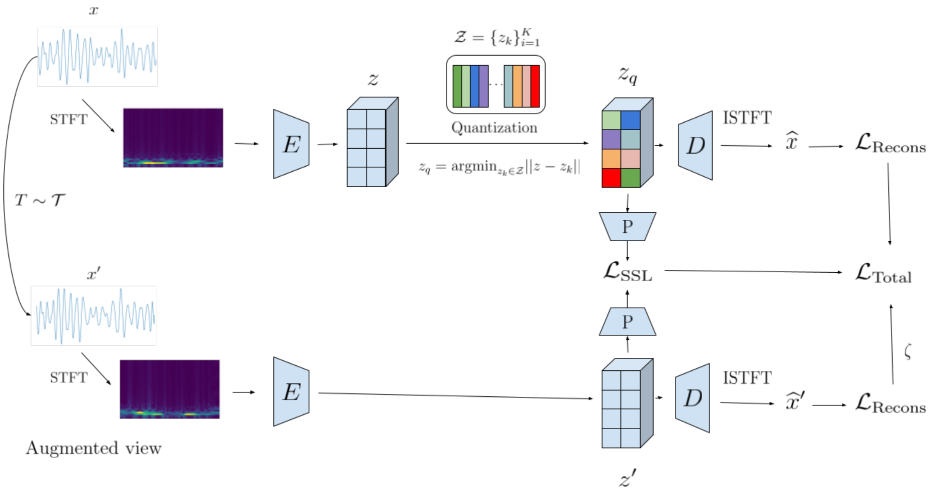
x
STFT
E
z
$\mathcal{Z} = \{z_k\}_{i=1}^{K}$
Quantization
$z_q = \text{argmin}_{z_k \in \mathcal{Z}} ||z - z_k||$
z_q
D
ISTFT
$\widehat{x}$
$\mathcal{L}_{\text{Recons}}$
$T \sim \mathcal{T}$
P
$\mathcal{L}_{\text{SSL}}$
$\mathcal{L}_{\text{Total}}$
P
ζ
x'
STFT
E
D
ISTFT
$\widehat{x}'$
$\mathcal{L}_{\text{Recons}}$
Augmented view
z'

Figure 1: Overview of proposed model.

0.1 Stage 1: Tokenization

0.1.1 VQVAE

The the naive VQVAE model as described in [**TimeVQVAE**] is baseline for our experiments.

An encoder, decoder, and codebook are to be optimized by compressing the input into discrete latent space, minimizing information loss by comparing input to the output, which ideally are equal. We follow [**TimeVQVAE**] and augment time-series into time-frquency domain, but leave the high-low frequency split for future work.

Method

A schematic overview of the our VQ-VAE model is presented in "Figure here"

A time series is first augmented into time-frequency domain using the Short-time Fourier Transform (cite pytorch stft). Then it is encoded into the continuous latent space, and is discretized by the codebook via the argmin process. In the argmin process the continuous token is compared to every discrete token in the codebook, and replaced by the closes discrete token in terms of euclidean distance. Then, the decoder maps the discrete token back to time-frequency domain, before finally being mapped back to time domain using the ISTFT.

Implementation details

0.1.2 NC-VQ-VAE

We have a common framework for the two SSL methods.

As our overarching objective is to learn tokens that eases the prior learning and train a better generative model, we are primarily interested in lowering the reconstruction loss. This lead us to only provide augmented views to one branch. We denote the latent variables by z and quantized latent variables z_q. All augmented values are denoted by as asterix, i.e z' is a latent variable in the augmented branch.

The framework is a siamese architecture with upper/original branch identical to the VQ-VAE model presented above. The the lower/augmented branch is identical, except for a lack of quantization layer.

We compute a SSL loss between projected values of z_q and z'.

We compute reconstruction losses $\mathcal{L}_{\text{Rec}}(\hat{x})$ and $\mathcal{L}_{\text{Rec}}(\hat{x}')$, of both original and augmented view.

Method

Loss

Augmentations

TODO: Talk about soft vs hard augmentations. Temporal perserving vs not etc.

We used the following collection of augmentation techniques.

- Amplitude Resizing
- Window Warp
- Slice and Shuffle
- Gaussian noise

Implementation details

0.1.3 Barlow Twins VQ-VAE

Model Architecture

An encoder, decoder, codebook, and projector are to be optimized. Produce two augmented views of the time-series, augment views into time-frequency domain and encode into latent space. Choose one view for quantization, decoding and comparison to original time series (VQVAE loss). Project both latent embeddings and calculate Barlow loss. Update using both VQVAE and Barlow loss.

A schematic overview of the BT-VQ-VAE model is presented in "Figure here"

0.1.4 VIbCReg VQ-VAE

Method

Implementation details

Training

0.2 Stage 2: Prior Learning

0.2.1 MaskGIT

Regular MaskGIT, token context MaskGIT, learnable codebook MaskGIT

0.3 UCR Archive