You're your own best teacher: A Self-Supervised Learning Approach For Expressive Representations in Time Series Generation

Johan Vik Mathisen

June 13, 2024

Our work involves building a framework that leverage non-contrastive self-supervised learning (SSL) to improve representation learning in a Vector Quantized Variational Autoencoder (VQVAE) type model. This section presents the works related to ours. To the best of our knowledge, enhancing VQ-based tokenization models with SSL methods has not yet been investigated in the time series domain. Therefore, the related works consist of models used as different components in our model.

We base our model on a simplified version of TimeVQVAE, which utilize a bidirectional transformer model, MaskGIT, for prior learning. Although our non-contrastive SSL extension could use any model with a siamese architecture, we experiment with the proven methods Barlow Twins and VIbCReg.

0.1 TimeVQVAE

TimeVQVAE is a time series generation model based on VQVAE and MaskGIT. It is the first to our and the authors knowledge that utilizes vector quantization (VQ) to address the time series generation problem. TimeVQVAE employs a two-stage approach similar to VQVAE, with a bidirectional transformer, akin to MaskGIT, for prior learning. The authors introduce VQ modeling in the time-frequency domain, separating data into high and low-frequency components to better retain temporal consistencies and generate higher quality samples.

In addition to VQ modeling in the time-frequency domain, TimeVQVAE presents a process for sampling jointly from high and low-frequency latent spaces and enables guided class-conditional sampling. By appending a class token, similar to [**dosovitskiy2021image**], the model can generate synthetic samples both conditionally and unconditionally.

Our work extends a variation of the TimeVQVAE model without the high-low frequency split, reducing the prior learning method to MaskGIT with the addition of guided class-conditional sampling. Here, we present the tokenization stage and refer the reader to [**VQVAE**] for details on the prior model training.

0.1.1 Tokenization

The tokenization stage of TimeVQVAE follows a structure similar to VQVAE, with the key difference being the frequency split. An overview of the model is presented in Figure **??**. Initially, a time series is mapped to the time-frequency domain using the Short-time Fourier Transform (STFT). The time-frequency representation is then separated into two branches: one zero-padding the high-frequency (HF) region and the other zero-padding the low-frequency (LF) region. Each branch follows the VQVAE architecture, with separate encoders, decoders and codebooks denoted by E_{LF}, E_{HF}, D_{HF}, D_{HF} and $\mathcal{Z}_{\mathrm{LF}}$, $\mathcal{Z}_{\mathrm{HF}}$ respectively. The output of the decoders are again zero-padded giving $\widehat{u}_{\mathrm{LF}}$ and $\widehat{u}_{\mathrm{HF}}$, which are then mapped back to the time domain using the Inverse Short-time Fourier Transform (ISTFT) to produce the reconstructed HF and LF components, $\widehat{x}_{\mathrm{LF}}$ and $\widehat{x}_{\mathrm{HF}}$, of the time series.

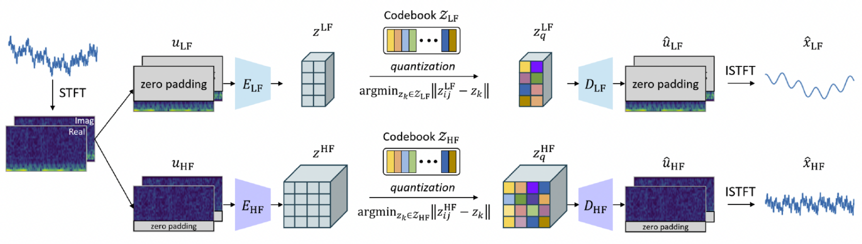

Figure 1: Stage 1: Tokenization. Figure taken with permission from [1]

Loss

The codebook loss of TimeVQVAE is similar to the codebook loss presented in Section **??** but reflects the HF-LF split

$$\begin{aligned}\mathcal{L}_{\text{codebook}} = &\,||\,\mathrm{sg}[E_{LF}(\mathcal{P}_{\mathrm{LF}}(\mathrm{STFT}(x)))] - z_q^{\mathrm{LF}}||_2^2 \\ &+ ||\,\mathrm{sg}[E_{HF}(\mathcal{P}_{\mathrm{HF}}(\mathrm{STFT}(x)))] - z_q^{\mathrm{HF}}||_2^2 \\ &+ \beta||E_{LF}(\mathcal{P}_{\mathrm{LF}}(\mathrm{STFT}(x))) - \mathrm{sg}[z_q^{\mathrm{LF}}]||_2^2 \\ &+ \beta||E_{HF}(\mathcal{P}_{\mathrm{HF}}(\mathrm{STFT}(x))) - \mathrm{sg}[z_q^{\mathrm{HF}}]||_2^2,\end{aligned} \tag{1}$$

where $\mathcal{P}_{[\cdot]}$ denotes the zero-padding operation. The reconstruction loss is computed both on time and time-frequency reconstructions and is given by

$$\begin{aligned}\mathcal{L}_{\text{recons}} = &\,||x_{\mathrm{LF}} - \widehat{x}_{\mathrm{LF}}||_2^2 + ||x_{\mathrm{HF}} - \widehat{x}_{\mathrm{HF}}||_2^2 \\ &+ ||u_{\mathrm{LF}} - \widehat{u}_{\mathrm{LF}}||_2^2 + ||u_{\mathrm{HF}} - \widehat{u}_{\mathrm{HF}}||_2^2.\end{aligned} \tag{2}$$

The total loss is given by

$$\mathcal{L}_{\mathrm{VQ}} = \mathcal{L}_{\text{codebook}} + \mathcal{L}_{\text{recons}}. \tag{3}$$

To update the codebooks, TimeVQVAE uses an exponential moving average method as presented in Appendix A.1 of [**VQVAE**].

0.2 MaskGIT

The Masked Generative Image Transformer (MaskGIT), introduced in [3], is a generative transformer model for image synthesis developed by Google Research. The innovation of MaskGIT lies in its token generation method. Unlike traditional autoregressive generative transformers that treat images as a sequence of tokens, MaskGIT employs a bi-directional transformer for image synthesis. This allows MaskGIT to predict tokens in all directions during training, offering a more natural approach to image modeling. During inference, MaskGIT begins with a blank canvas, predicting the entire image iteratively by conditioning on the most confident pixels from previous predictions.

MaskGIT's architecture relies on a tokenization procedure in its first stage. In the original paper, VQGAN [2] was used for this purpose, focusing on improving the second stage. Therefore, our discussion will primarily address this aspect of the model.

0.2.1 Masked Visual Token Modeling (Prior learning)

For prior learning, the codebook learned in the tokenization procedure is provided with a masking vector, which is the embedding of the special masking token, denoted by `M`. The input embedding in the bidirectional transformer is initialized with this expanded codebook. For an image X in the dataset $\mathcal{D}$, let $z = \{z_{k_i}\}_{i=1}^N$ denote the sequence of codewords obtained by passing X through the VQ-Encoder. This sequence can equivalently be described as a sequence of indices $s = \{k_i\}_{i=1}^N$. Prior learning involves masking this sequence and training the bidirectional transformer to predict the masked indices.

Let $s = \{k_i\}_{i=1}^N$ be the sequence of indices described above, and denote the corresponding binary mask by $M = \{m_i\}_{i=1}^N$. During training, a subset of s is replaced by the masking token `M` according to the binary mask M. This is done by

$$s_{\text{Mask}} = s \odot (1_N - M) + M \cdot \mathtt{M}, \tag{4}$$

where $\odot$ is the Hadamard product (pointwise multiplication), and 1_N is a vector with the same shape as M and s.

The sampling procedure, or choice number of tokens to mask, is parameterized by a mask scheduling function γ. The sampling can be summarized as follows

- Sample $r \sim U(0, 1]$.
- Sample $\lceil \gamma(r) \cdot N \rceil$ indices I uniformly from $\{0, \dots, N-1\}$ without replacement.
- Create M by setting $m_i = 1$ if $i \in I$, and $m_i = 0$ otherwise.

A forward computation is illustrated in Figure **??**, where the bidirectional transformer predicts the probabilities $p(s_i | s_{\text{Mask}})$ of each masked token. The training objective is to minimize the negative log likelihood of the masked tokes, conditional on the unmasked

$$\mathcal{L}_{\text{Mask}} = -\mathbb{E}_{s \in \mathcal{D}} \left[\sum_{i \in I} p(s_i | s_{\text{Mask}}) \right], \tag{5}$$

and $\mathcal{L}_{\text{Mask}}$ is computed as the cross entropy between the ground truth one-hot token and the predicted token probabilities.

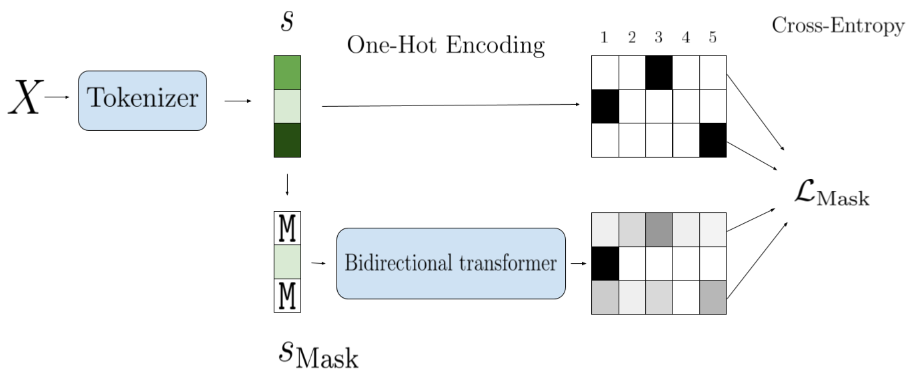

Figure 2: MaskGIT forward computation.

0.2.2 Iterative Decoding (Sample generation)

The bi-directional transformer could in principle predict all masked tokens and generate a sample in a single pass by sampling according to the predicted probabilites $p(\widehat{s}_i|s_{\text{Mask}})$ from a forward pass of an all masked sequence. However, this approach presents certain challenges. To address these, [3] proposes a novel non-autoregressive decoding method to synthesize samples in a constant number of steps.

The decoding process goes from $t = 0$ to T. To generate a sample during inference, the process starts with an all masked sequence, denote as $s^{(0)}_{\text{Mask}}$. At iteration t, the model predicts the probabilities for all the mask tokens, $p(\widehat{s}_i|s^{(t)}_{\text{Mask}})$, in parallel. At each masked index i, a token $s^{(t)}_i$ is sampled according to the predicted distribution, and the corresponding probability $c^{(t)}_i$ is used as a measure of the confidence in the sample. For unmasked tokens, a confidence of 1 is assigned to the true position.

The number of tokens $s^{(t)}_i$ with highest confidence that are kept for the next iteration is determined by the mask scheduling function. Specifically, $n = \lceil \gamma(t/T) \cdot N \rceil$ of the lower confidence tokens are masked again by calculating $M^{(t+1)}$ as follows

$$m^{(t+1)}_i = \begin{cases} 1, \text{ if } c_i < \text{Sort}([c^{(t)}_1, \dots, c^{(t)}_N])[n] \\ 0, \text{ otherwise} \end{cases} \tag{6}$$

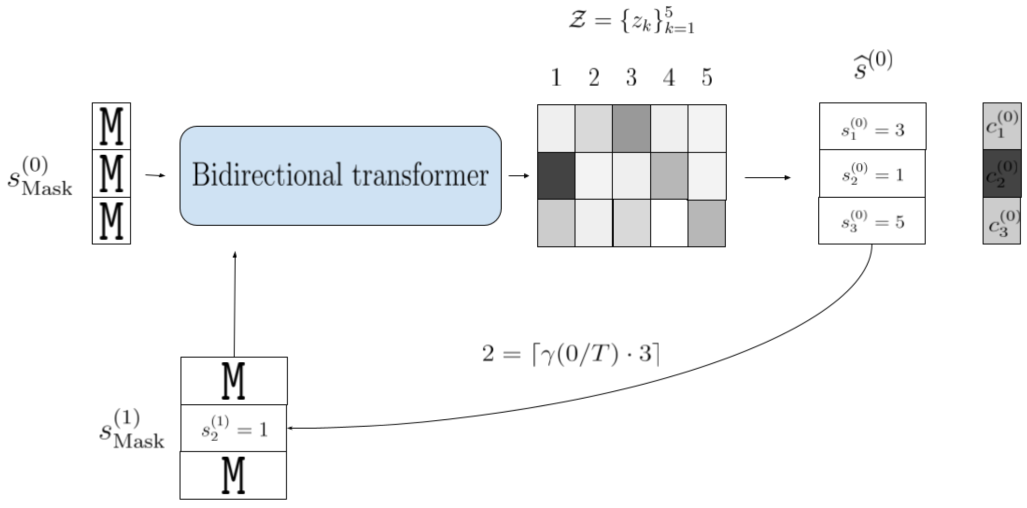

Figure 3: Illustration of first pass of the iterative decoding algorithm.

This process is illustrated in Figure **??**, which shows the first pass of the iterative decoding algorithm. The algorithm synthesizes a complete image in T steps. For image generation, the cosine scheduling function proved to be the most effective across all experiments in the original paper.

0.3 SSL

Our model leverages self-supervised learning algorithms to learn more expressive latent representations. In this section, we present the relevant SSL algorithms for our work: Barlow Twins and VIbCReg.

0.3.1 Barlow Twins

Barlow Twins is a non-contrastive SSL method that applies the *redundancy-reduction principle* [5] from neuroscientist H. Barlow to a pair of identical networks. The key idea is to encourage representations of similar samples to be alike while reducing redundancy between the components of the vectors.

The model produces two augmented views of each sample and projects their representations onto a feature space such that their cross-correlation is close to the identity matrix. This approach enforces both distortion invariance and decorrelated features in the representations.

The Barlow Twins algorithm, illustrated in Figure **??**, starts out by creating

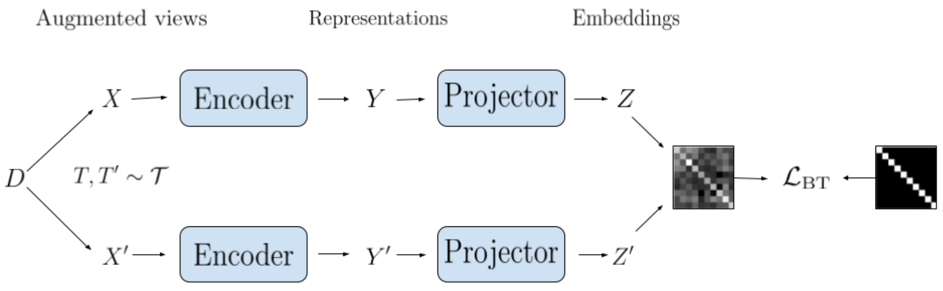

Figure 4: Overview of the Barlow Twins architecture. Figure inspired by [6]

two augmented views for each datapoint in a batch D. These augmentations are sampled from a collection of augmentations $\mathcal{T}$. The batches of augmented views $T(D) = X$ and $T'(D) = X'$, are then passed through an encoder to give representations Y and Y', which are further projected to produce embeddings Z and Z'. The embeddings are assumed to be mean centered across the batch dimension. The loss function is calculated using the cross-correlation matrix $\mathcal{C}$ between Z and Z', measuring its deviation from the identity. The Barlow Twins loss is defined as

$$\mathcal{L}_{\mathrm{BT}} = \overbrace{\sum_i (1 - \mathcal{C}_{ii})^2}^{\text{Invariance}} + \lambda \overbrace{\sum_i \sum_{j \neq i} \mathcal{C}_{ij}^2}^{\text{Redundancy reduction}} , \tag{7}$$

where

$$\mathcal{C}_{ij} = \frac{\sum_b z_{b,i} z'_{b,j}}{\sqrt{\sum_b (z_{b,i})^2} \sqrt{\sum_b (z'_{b,j})^2}} . \tag{8}$$

The *invariance term* helps make the embeddings invariant to the distortions introduced by the augmentations, pushing the representations closer together. The *redundancy reduction term* decorrelates the different vector components, reducing the information redundancy.

0.3.2 VIbCReg

VIbCReg [**lee2024computer**] is a non-contrastive SSL model with a siamese architecture based on VICReg [8]. It improves upon VICReg by incorporating better covariance regularization and IterNorm [9]. A key difference from Barlow Twins is that variance/covariance regularization is applied to each branch individually.

As with Barlow Twins, a batch D is augmented to create two views and passed through an encoder and projector. The embedding Z and Z' are then *whitened* using IterNorm [9] before calculating the similarity loss.

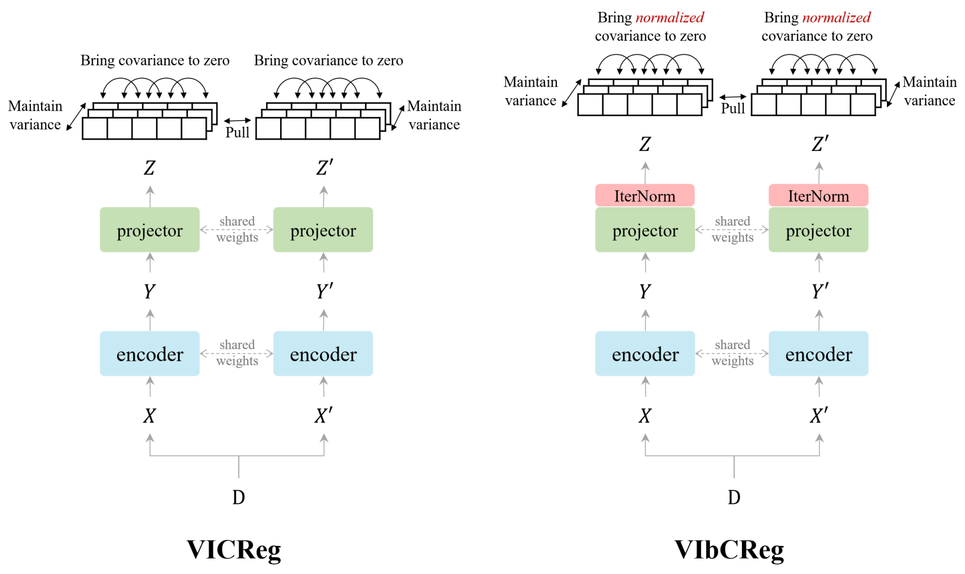

Figure 5: Overview of VIbCReg, and comparison with VICReg. Taken with permission from [**lee2024computer**]

The loss consists of a similarity loss between the branches, a feature decoration (FD) term and a feature component expressiveness (FcE) term at each branch. Input data is processed in batches, with $Z \in \mathbb{R}^{B\times F}$ where B and F denotes the batch and feature sizes, respectively. We denote a row in Z by Z_b and column by Z_f, and similarly for Z'.

The similarity loss is defined as the mean square error between the two embeddings

$$s(Z,Z') = \frac{1}{B}\sum_{b=1}^{B} ||Z_b - Z'_b||_2^2, \tag{9}$$

which encourages the embeddings to be similar. The FcE term, which encourages the variation across a batch to stay at a specified level γ, is defined as

$$v(Z) = \frac{1}{F}\sum_{f=1}^{F} \max(0, \gamma - \sqrt{\mathrm{Var}(Z_f)+\epsilon}), \tag{10}$$

where Var() is a variance estimator, γ is a target value for the standard deviation, which both in VIbCReg and VICReg is set to 1, and ϵ is a small scalar to prevent numerical instabilities.

For the FD loss, the embeddings are mean-shifted and normalized along the batch dimension

$$\widehat{Z}_b = \frac{Z_b - \bar{Z}}{||Z_b - \bar{Z}||_2} \text{ where } \bar{Z} = \frac{1}{B}\sum_{b=1}^{B} Z_b, \tag{11}$$

giving

$$\widehat{Z} = [\widehat{Z}_1, ..., \widehat{Z}_B]^T. \tag{12}$$

The normalized covariance matrix is computed as

$$C(Z) = \frac{1}{B-1}\widehat{Z}^T\widehat{Z}, \tag{13}$$

before calculating the mean square across all off-diagonal elements to obtain the FD loss

$$c(Z) = \frac{1}{F^2}\sum_{i \neq j} C(Z)_{ij}^2. \tag{14}$$

The total loss is then given by

$$\mathcal{L}_{\text{VIbCReg}} = \lambda s(Z, Z') + \mu[v(Z) + v(Z')] + \nu[c(Z) + c(Z')] \tag{15}$$

where λ, μ and ν are hyperparameters that determine the importance of each term. The normalization of the covariance matrix keep the range of the FD loss small, independent of data, and eases hyperparameter tuning across datasets.