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

Johan Vik Mathisen

May 14, 2024

TODO: relate our work to TimeVQVAE, Neural Representation, Barlow, MaskGIT,

TODO: Include something on time series generation / representation learning

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 TSG problem. It leverages a two stage approach similar to VQVAE and uses a bidirectional transformer akin to MaskGIT for prior learning. Additionally, they propose VQ modeling in time-frequency domain, separating data into high and low frequency components to better retain temporal consistencies and generate higher quality samples.

TimeVQVAE provides class-guided conditional sampling.

Our work in this thesis can be seen as a tangent of the paper "Vector Quantized Time Series Generation with a Bidirectional Prior Model" [1]. We simplify the architecture by omitting the HF/LF split. This reduces prior learning to resemble MaskGIT to a larger degree.

0.1.1 Tokenization

0.1.2 Prior learning

0.2 MaskGIT

The Masked Generative Image Transformer (MaskGIT) is a generative transformer model for image synthesis developed by Google Research. The novelty of the model lies in the token generation. Unlike popular autoregressive generative transformers, who treat images as a sequence of tokens, MaskGIT introduces an image synthesis paradigm using a bi-directional transformer decoder. This means that during training MaskGIT learns to predict tokens in all directions, an intuitively more natural way to consider images. At inference time MaskGIT starts out with a blank canvas and predicts the entire image, and iteratively keeps and conditions on the most confident pixels.

The model assumes a tokenization procedure for stage 1, and in the original paper they used VQGAN [2]. As MaskGIT only focuses on improving stage 2, present only that part.

0.2.1 Masked Visual Token Modeling (Prior learning)

For some image X in the dataset $\mathcal{D}$, let $Y = \{y_i\}_{i=1}^{N}$ denote the latent tokens obtained by passing X through the VQ-Encoder and denote the corresponding binary mask by $M = \{m_i\}_{i=1}^{N}$. During training a subset of Y is replaced by a special masking token we denote by $\mathtt{M}$ according to the binary mask M. This is done by

$$Y_{\text{Mask}} = Y \odot (1_N - M) + M \cdot \mathtt{M}, \tag{1}$$

where $\odot$ is the Hadamard product, i.e point wise multiplication, and 1_N is a vector with the same shape as M and Y.

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.

The training objective is to minimize the negative log likelihood of the masked tokes, conditional on the unmasked

$$\mathcal{L}_{\text{Mask}} = -\mathbb{E}_{Y \in \mathcal{D}} \left[\sum_{i \in I} p(y_i | Y_{\text{Mask}}) \right] \tag{2}$$

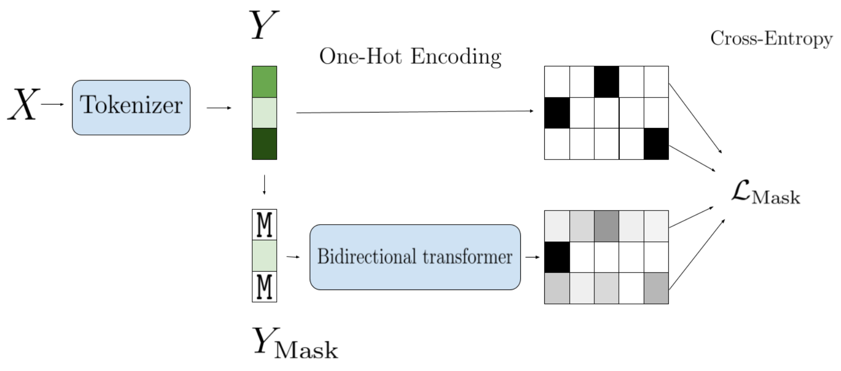

Figure 1: MaskGIT forward computation.

A bidirectional transformer is used to predict the probabilities $p(y_i|Y_{\text{Mask}})$ of each masked token, and $\mathcal{L}_{\text{Mask}}$ is computed as the cross entropy between the ground truth one-hot token and the predicted token.

0.2.2 Iterative decoding (Image generation)

The bi-directional transformer could in principle predict all [MASK] tokens and generate a sample in a single pass by simply sampling from the logits obtained from a forward pass of an all masked sequence. However, there are challenges with this approach. In their original article [3] proposes a novel non-autoregressive decoding method to sythesize samples in a constant number of steps.

The decoding process goes from $t = 0$ to T. To genereate a sample at inference time one starts out with a all masked sequence which we denote by $s_M^{(0)}$. At iteration t the model predicts the probabilities for all the [MASK] tokens, $p(\hat{s}_{ij}^{(t)}|s_M^{(t)})$, in parallell. Then at each masked entry ij we sample a token index based on its predicted distribution.

0.2.3 Masking design

For image generation, cosine schedulig function proved best across all experiments in the original paper. Start out by selecing just a few

0.3 SSL

Our model leverages SSL algorithms in order to learn more expressive latent representations. Here we present the relevant algorithms for our work.

0.3.1 Barlow Twins

What is it?

Barlow Twins is a non-constrastive SSL method based on applying the *redundancy-reduction principle* (or efficient coding hypothesis) [5] from the neroscientist H. Barlow to a pair of identical networks.

In essence the models encurage representations of similar samples to be similar, while simultaneously reducing the amount of redundancy between the components of the vectors. This is done by producing two distorted views of each sample and embedding these in a vast feature space, in such a way that their cross-correlation is close to the identity.

How does it work?

Start out with a sample X and creates two augmented (distorted) views X_1 and X_2. The views are then mapped to a latent space by two identical encoders, giving Y_1 and Y_2. Then the projector embeds the latent representations in a vast space, giving Z_1 and Z_2. Finally the similarity of the two embeddings are measured by the empirical cross-corelation.

TODO: Ask for premission?? to use this or make own

The loss function is calculated as the difference of the empirical cross-correlations of Z_1 and Z_2 is then calculated and the identiry matrix.

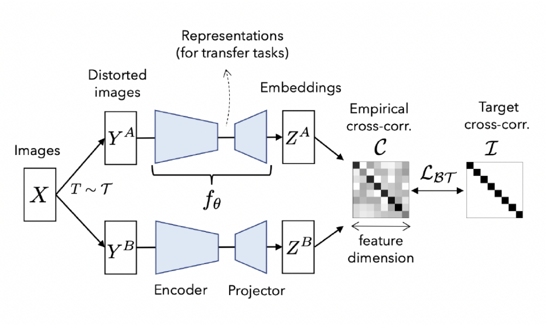

Figure 2: [6]

0.3.2 VIbCReg

TODO: What is it?

VIbCReg [7] is a non-contrastive SSL model based on VICReg [8], but with better covariance regularization. It has a joint embedding architecture.

TODO: How it works

Two different views of the input data is encoded into representations Y Y'. The representations are further mapped to a larger space by a *projector* with an IterNorm [9] layer. The loss is computed using the projected values Z and Z'.
The loss consists of a similarity loss between the branches, and feature decoration (FD) loss together with a feature component expressiveness (FcE) term at each branch.

TODO: Loss

Input data is processed in batches. Denote $Z = [z_1, ..., z_B]^T \in \mathbb{R}^{B\times F}$, and similarly for Z', where B and F denotes the batch and feature sizes respectively. Var() is a variance estimator, γ is a target value for the standard deviation, which both in VIbCReg and VICReg is set to 1. ϵ is a small scalar preventing numerical instabilities.
Similarity loss

$$s(Z, Z') = \frac{1}{B}\sum_{b=1}^{B} ||Z_b - Z'_b||_2^2 \tag{3}$$

FcE/Variance term
FD/covariance term

$$C(Z) = \frac{1}{B-1}\left(\frac{Z-\bar{Z}}{||Z-\bar{Z}||_2}\right)^T\left(\frac{Z-\bar{Z}}{||Z-\bar{Z}||_2}\right) \text{ where } \bar{Z} = \sum_{b=1}^{B} Z_b \tag{4}$$