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

Johan Vik Mathisen

May 20, 2024

0.1 Implementation details

We follow [**TimeVQVAE**] closely in the encoder/decoder/codebook implementation, and

Time Frequency Modelling

The short time fourier transform (STFT) and its inverse ISTFT are implemented with `torch.stft` and `torch.istft` respectively. We follow [**TimeVQVAE**] and set the main parameter `nfft` to 8, and use default parameters for the rest.

Encoder and decoder

Appendix C.3 from [**TimeVQVAE**]
The same encoder and decoder architecture as in [**VQVAE**] is used and further use the implementation from [**nadavbh12**] with adaptations from [**TimeVQVAE**].
The encoder consists of n downsampling convolutional blocks (Conv2d - BatchNorm2d - LeakuReLU), followed by m residual blocks (LeakuReLU - Conv2d - BatchNorm2d - LeakuReLU - Conv2d). The downsampling convolutional layers are implemented with parameters kernel size=(3,4), stride=(1,2), padding=(1,1), while the residual convolutional layers have parameters kernel size=(3,3),stride=(1,1), padding=(1,1).
The decoder is implemented similarity with m resudial followed n upsampling layers using transposed convolutional layers with same parameters as in the encoder.
The downsampling rate is determined by 2^n where n is set such that z has width of 32.

VQ

Implementation from lucidrains/vector-quantize-pytorch. https://github.com/lucidrains/vector-quantize-pytorch
Codebook size = 32 and dimension = 64.
Use exponential moving average with decay 0.9, and commitment loss with weight $\beta = 1$.

Augmentations

window warp: window ratio: 0.4,min window warp: 0.9, max windowwarp: 2.0
amplitude resize: Amp Rrate: 0.2
gaussian noise: gaus mean: 0, gaus std: 0.05
slice and shuffle:

Projector

We follow the implementation from [**lee2024computer**] for both Barlow Twins and VIbCReg.
Barlow Twins: Projector's dimension size is set to 4096. λ is set to $5 \cdot 10^3$.
VIbCReg: The dimension of the the projector is set to 4096. λ and μ are both set to 25, while ν is set to 100.

Prior learning

The number of iterations in the iterative decoding algorithm T, is set to 10, following [**chang2022maskgit**]. We too use the cosine as mask scheduling function γ. The implementation is adopted from [**TimeVQVAE**].

Optimizer

The AdamW optimizer with batch sizes for stage1: 128 and stage2: 256, initial learning rate 10^{-3}, cosine learning rate scheduler and weight decay of 10^{-5}. We run 1000 epochs for both stage 1 and 2.

Evaluation

KNN and SVM are implemented using scikit-learn. $K = 5$ in KNN and linear kernel in SVM.
FCN for IS, FID and CAS

0.2 Initial Experimentation and Model Development

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/improved learning of the generative model.

During development we encountered several problems:
When we attempted a siamese architecture, with quantization in the augmented branch, and to derive the SSL loss from the discrete representations there were a correlation problem. The codewords were very highly correlated, which resulted from the passing both views through the VQ. $SSL(z_q, z_q')$
In an attempt to solve this we attempted to derive the SSL loss from the continuous latent representations, but the resulting discrete latent representations performed poorly on the downstream classification task. Separability problem: $SSL(z, z')$
The solution was to remove the VQ in the augmented branch and rather derive

the SSL loss from z_q and z'. Solution: $SSL(z_q, z')$

Overfitting problem: Using $SG()$ on augmented branch / Not using augRecons

0.3 Main Experiments

We are primarily interested in two things. For stage 1, if NC-VQVAE learns more expressive representations, i.e are we able to reconstruct on par with VQVAE while simultaneously improve on downstream classification. For stage 2 we are interested in the effect NC-VQVAE has on prior learning and time series generation.

We evaluate our model NC-VQVAE with both Barlow Twins and VIbCReg as SSL method against the naive VQVAE as described in [**TimeVQVAE**]. Firstly we look at the tokenization models, evaluating the reconstruction capability and performance on downstream classification. Then we train a prior model on top of the different tokenization models and evaluate the performance of the generative models by IS, FID, CAS and visual inspection.

0.4 Stage 1

0.4.1 Augmentations

In our experiments we consider three sets of augmentations with different characteristics. They are

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

Amplitude Resizing + Window Warp scales in both x and y direction. The window warp has similar qualities to phase shift, but not uniformly and keeps endpoints fixed. They were considered as the observed conditional distribution in some datasets, such as ShapesALL, had similar overall shape, but peaked with different amplitude and at different locations. Thus the augmented view had similar characteristics as the conditional distribution of the original view.

Slice and Shuffle crops the time series into three sections and permutes them. For datasets with sharp modularity and few peaks, such as ElectricDevices, the augmentation provides a view with peaks occurring at timestamps not seen in the training data, which could improve the reconstruction on unseen data, as well as encouraging the model to focus more on the existence of a peak rather than its specific location. For some datasets such as FordA, the semantics of the dataset is preserved under this augmentation, despite their continuous nature.

Gaussian noise adds a nose $\epsilon \sim N(0, 0.05)$ to each datapoint in the time series. This introduces, in many cases, a substantial high frequency component. As the naive VQVAE described in [**TimeVQVAE**] had trouble with reconstruction of HF components, this augmentation could provide more emphasis on these. The reconstruction of the augmented views can too provide more information regarding HF components for the decoder.

0.4.2 Evaluation

Reconstruction, Classification and Visual inspection

0.5 Stage 2

0.5.1 Evaluation

- **IS**:
- **FID**:
- **CAS**:We evaluate the CAS for TSTR by using the Supervised FCN on all our models considered and compare against the baseline model to investigate the relative performance.
- **Visual inspection**:
- **Token usage**:

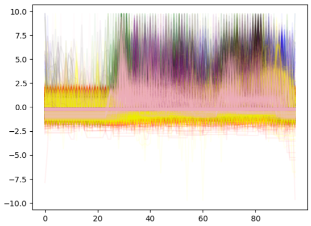
0.6 UCR Time Series Classification Archive

The evaluation of our model NC-VQVAE is done on a subset of the UCR Time Series Archive [**UCRArchive2018**]. The UCR archive is a collection of 128 datasets of univariate time series for time series classification. The different datasets in the archive span a wide range characteristics and include among others sensor, device, image-derived and simulated data. Each dataset has a predefined training and test split.

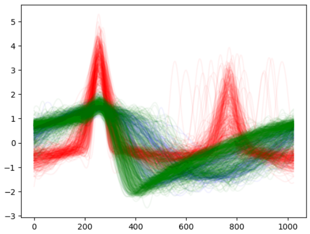
Our subset of the UCR archive is

Type	Name	Train	Test	Class	Length
Device	ElectricDevices	8926	7711	7	96
Sensor	FordB	3636	810	2	500
Sensor	FordA	3601	1320	2	500
Sensor	Wafer	1000	6164	2	152
Simulated	TwoPatterns	1000	4000	4	128
Sensor	StarLightCurves	1000	8236	3	1024
Motion	UWaveGestureLibraryAll	896	3582	8	945
ECG	ECG5000	500	4500	5	140
Image	ShapesAll	600	600	60	512
Simulated	Mallat	55	2345	8	1024
Image	Symbols	25	995	6	398
Sensor	SonyAIBORobotSurface2	27	953	2	65
Sensor	SonyAIBORobotSurface1	20	601	2	70

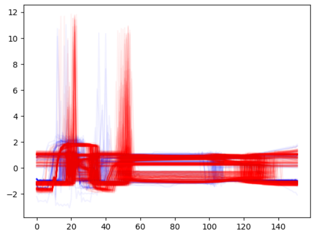
Table 1: The subset of the UCR Archive considered for our experiments.



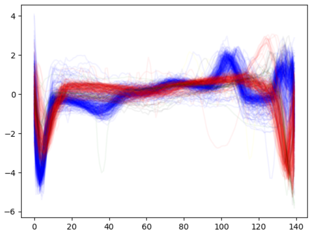
ElectricDevices



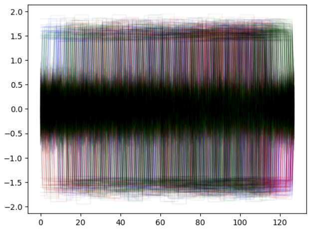
StarLightCurves



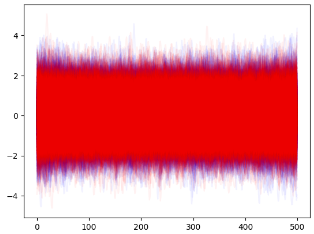
Wafer



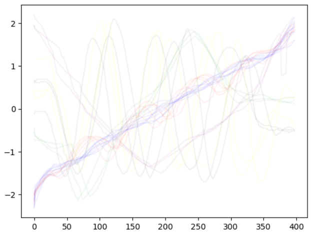
ECG5000



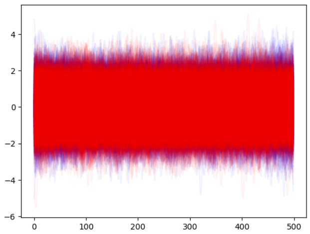
TwoPatterns



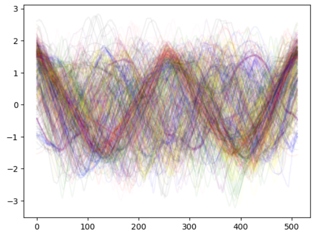
FordA



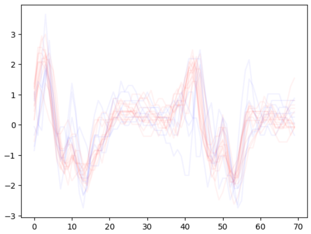
Symbols



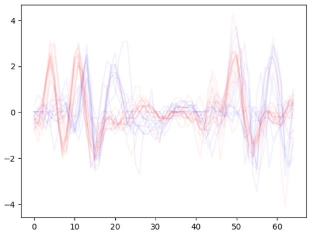
FordB



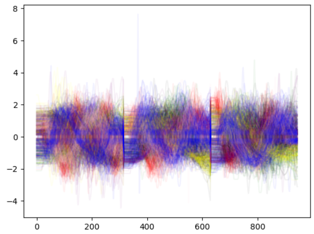
ShapesAll



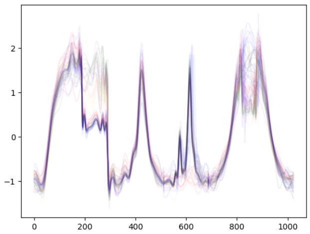
SonyAIBORobot-
Surface1



SonyAIBORobot-
Surface2



UWaveGestureLib-
raryAll



Mallat