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

Johan Vik Mathisen

June 7, 2024

0.1 Main Experiments

TODO: This section i need som assistance with. Find it difficult to write anything smart.

To evaluate our model NC-VQVAE and address the research questions, we compare it to the naive VQVAE using both Barlow Twins and VIbCReg as self-supervised learning methods. Additionally, as we are interested in the effect of augmentations on representation learning, for each SSL method we train three distinct models using different sets of augmentations. Furthermore, each configuration is trained using four different seeds, resulting in a total of 364 models trained at each stage. All experiments were run on the Idun HPC cluster [**Idun**], using several compute nodes. The total training time of all 728 models was approximately 700 hours.

0.1.1 Implementation details

In our implementation, we closely follow to the methodology outlined in [**TimeVQVAE**], particularly in the design and deployment of the encoder, decoder, and codebook. The code base for the project is found at `https://github.com/erlendlokna/Generative-SSL-VQVAE-modelling`

Time Frequency Modelling

We utilize the Short Time Fourier Transform (STFT) and its inverse (ISTFT) for time-frequency modeling, using the functions `torch.stft` and `torch.istft`, respectively. Consistent with [**TimeVQVAE**], we set the primary parameter $\mathtt{n}_{\mathtt{fft}}$ to 8 and use default parameters for others. This configuration yields a frequency axis spanning $[1, 2, 3, 4, 5]$ and a time axis half the length of the original.

Encoder and decoder

The encoder and decoder architectures closely resemble those described in [**nadavbh12**], with further adaptations from [**TimeVQVAE**].

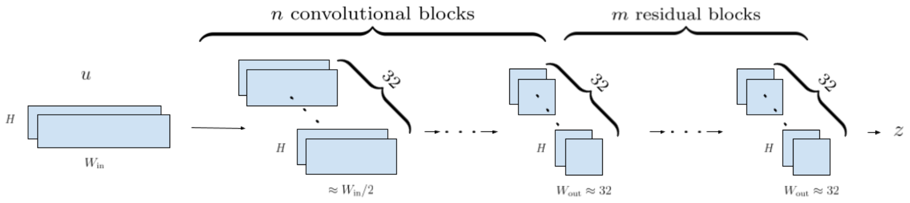


Figure 1: Overview of the encoder architecture. The decoder architecture is simply obtained by reversing the arrows and switching out the convolutional block for transposed convolutional blocks.

As illustrated in Figure **??**, the encoder consists of n downsampling convolutional blocks (`Conv2d - BatchNorm2d - LeakyReLU`), followed by m residual blocks (`LeakyReLU - Conv2d - BatchNorm2d - LeakyReLU - Conv2d`). Downsampling convolutional layers are implemented with parameters:

```
kernel size=(3,4),stride=(1,2),padding=(1,1),
```

downsampling the temporal axis by a factor of 2 per block. Residual convolutional layers have parameters:

```
kernel size=(3,3),stride=(1,1),padding=(1,1).
```

The decoder mirrors this architecture, featuring m residual and n upsampling layers using transposed convolutional blocks with identical parameters.

The downsampling rate is determined by 2^n, ensuring the resulting latent representation (z) has a width of 32. While we set the number of residual blocks $m = 4$. For an in depth walkthrough of the TimeVQVAE implementation details, we direct readers to Appendix C.3 of [**TimeVQVAE**].

VQ

We adopt a codebook size of 32 and a dimension of 64, utilizing exponential moving average with a decay of 0.9 and a commitment loss weight $\beta = 1$.

SSL

For both Barlow Twins and VIbCReg, we implement the projector following the guidelines in [**lee2024computer**]. We use identical projectors for both methods, consisting of 3 linear layers. The first two are normalized using `BatchNorm1d`. Both hidden and output layer has dimension size set to 4096.

For Barlow Twins the weight of the redundancy reduction term, λ, is set to 0.005. The weight of the Barlow Twins loss in the NC-VQVAE is set to $\eta = 1/D$ where D is the projector output dimension.

For VIbCReg, the weights of the similarity loss, λ, and variance loss, μ, are both set to 25, while for the covariance loss, ν is set to 100. The weight of the VIbCReg loss in the NC-VQVAE is set to $\eta = 0.01$.

Augmentations

For the reconstruction loss on the augmented brach we set the weight ζ to 0.1.

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 transforms in both x and y direction. The window warp augmentation randomly selects a section of the time series and speeds it up or down, and the factor is choose randomly from the interval $[0.9, 2.0]$. We interpolate in order to obtain a time series of equal length as the original. It has similar qualities to phase shift, but not uniformly and keeps endpoints fixed. The amplitude resize multiplies the time series by a factor of $1 + N(0, 0.2)$. This set was 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 as seen in Figure **??**. In many cases we will refer to this set of augmentations simply as warp.

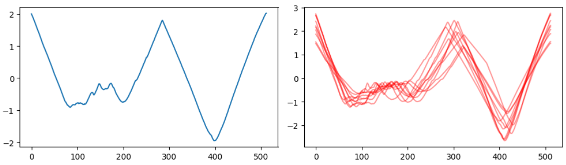


Figure 2: ShapesAll: Original (left), augmented (right). 15 instances of Amplitude Resizing + Window Warp applied to the original sample.

Slice and Shuffle crops the time series into four randomly selected 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 is illustrated in Figure **??**. This 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. In many cases we will refer to this augmentation simply as slice.

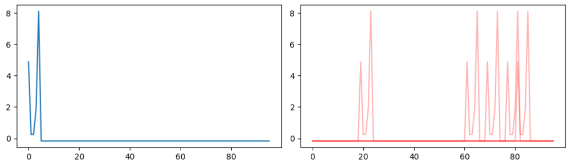


Figure 3: ElectricDevices: Original (left), augmented (right). 5 instances of Slice and Shuffle applied to the original sample.

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 seen in Figure **??**. 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. Of the three augmentations, gaussian noise provides the most predictable augmented views from a numerical standpoint, which might result in a SSL loss which is easier to minimize. In many cases we will refer to this augmentation simply as Gaussian or gauss.

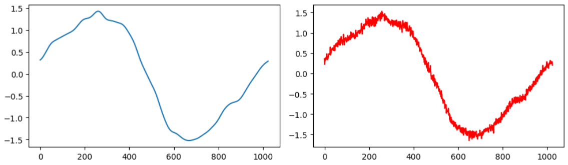


Figure 4: StarLightCurves: Original (left), augmented (right). One instance of Gaussian noise applied to the original sample.

Prior learning

We adopt the implementation from [**chang2022maskgit**]. In the iterative decoding algorithm, the number of iterations T is set to 10, and use cosine as mask scheduling function (γ).

Training

We utilize the AdamW optimizer with batch sizes set to 128 for stage 1 and 256 for stage 2, an initial learning rate of 10^{-3}, cosine learning rate scheduler, and a weight decay of 10^{-5}. Both stage 1 and stage 2 training procedures run for 1000 epochs.

0.1.2 Evaluation

For downstream classification, K-nearest neighbors (KNN) and Support Vector Machines (SVM) are implemented using scikit-learn, with $K = 5$ for KNN and a linear kernel for SVM.

SupervisedFCN is employed for calculating Inception Score (IS), Fréchet Inception Distance (FID), and Classification Accuracy Score (CAS). See Appendix B and C.2 of [**VQVAE**] for detailed information.

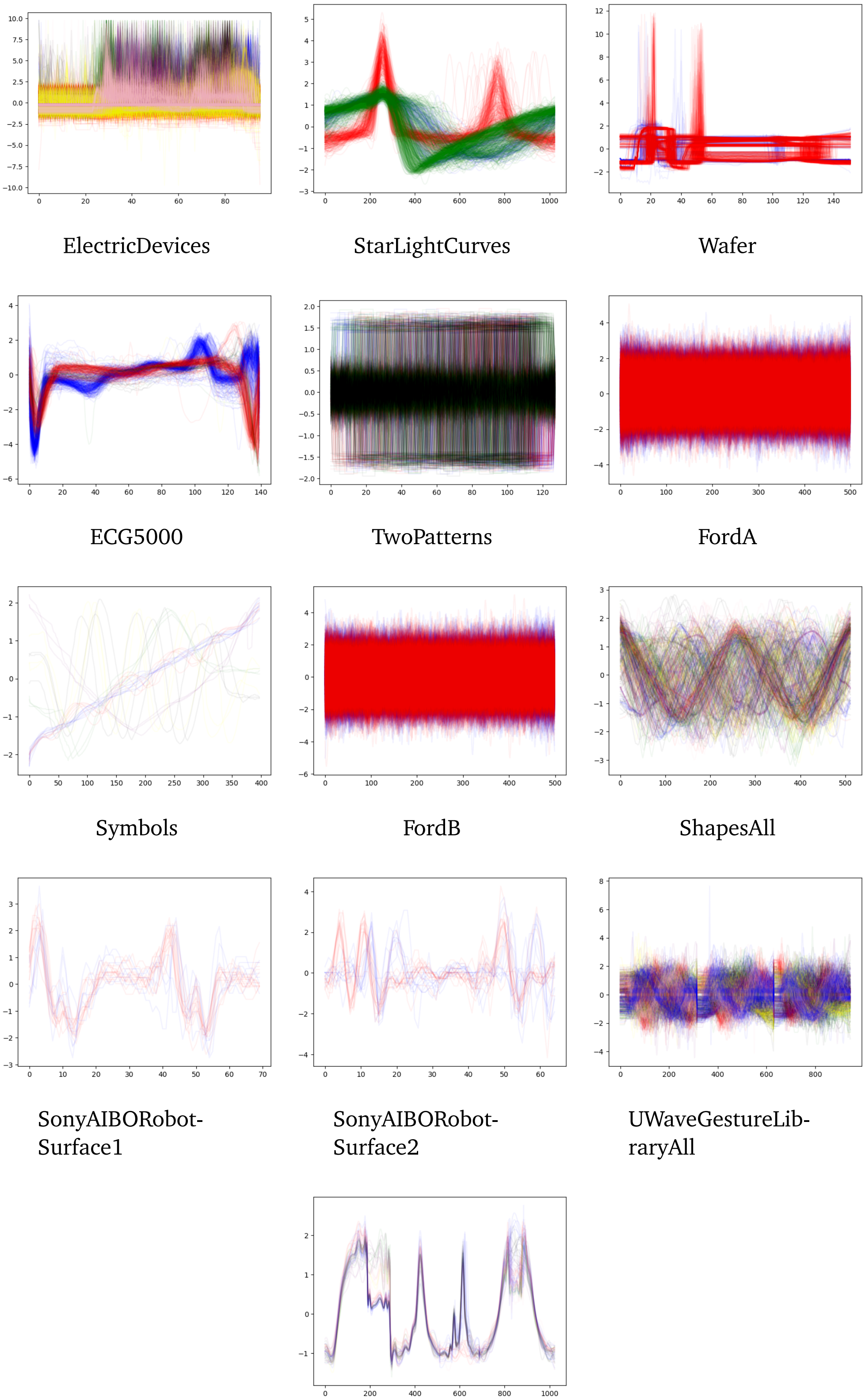
0.2 UCR Time Series Classification Archive

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 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.

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.

The subset selected for our experiments is presented in Table **??**. We choose to test on a subset, rather than on the entire UCR Archive, due to computational limitations as well as to more thoroughly investigate the effect of our models and the role of augmentations. The subset is chosen such that they span a wide range of train set sizes, lengths, classes and type, while the class distributions have visually different characteristics which can be seen from table **??** and figure **??**.



ElectricDevices

StarLightCurves

Wafer

ECG5000

TwoPatterns

FordA

Symbols

FordB

ShapesAll

SonyAIBORobot-Surface1

SonyAIBORobot-Surface2

UWaveGestureLib-raryAll

Mallat

Figure 5: Our selected subset of the UCR Archive. All time series in the training set are plotted and color coded according to label.