Corpora Composition for Unsupervised Timeseries Representation Learning

Zion Steiner

***Abstract*—Timeseries representation learning is a method of automated feature engineering made possible by advances in deep learning architecture. These techniques aim to distill important information and reduce the dimensionality of timeseries, enabling more efficient downstream data-mining. There are several traits that make learned representation spaces valuable, among them transferability. Ideally, representations learned from one dataset will transfer well to other datasets and tasks. Our research assesses the quality and transferability of representations produced by two deep learning architectures: RNN-based Timenet and an unnamed dilated causal CNN model. We also explore the effect of training dataset composition on transferability.**

I. Introduction

Timeseries data poses a challenge to traditional data-mining tasks, such as classification. Timeseries are high-dimensional and contain complex patterns, making it difficult to craft universal heuristics for analyzing the data. Further, full-length timeseries suffer from the curse of dimensionality, as concepts like similarity lose their meaning in many-dimensional space.

This is the motivation for developing methods for extracting representative information about timeseries. Condensed timeseries representations are more efficient to manipulate. Good representations may also convey information such as class membership more clearly than raw timeseries. These benefits increase the applicability of traditional classification and clustering algorithms to timeseries data.

There have been several deep learning architectures proposed for learning representation spaces. Networks that learn data representation mappings are called encoders. Encoder networks are trained to take a timeseries as input and output a fixed-dimensional representation.

Although all neural networks perform feature engineering to some extent, these are typically task specific and often require learned parameters just to interpret the representations. Conversely, representation encoders decouple the learning of the representations and the task algorithm. While this decoupling may forfeit gains from fine-tuning representation spaces jointly with classifiers, it promotes the universality of the representations.

Both supervised and unsupervised methods have been proposed for training encoders. Unsupervised methods have the advantage of being label-agnostic. Rather than being restricted by an imposed structure mapping labels to data, unsupervised methods are designed to learn more generic data representations. Methods can be distinguished by their architecture and the formulation of their loss function.

Recurrent and convolutional architectures have shown promising results for unsupervised representation learning. Among these are RNN-based Timenet [1] and an unnamed CNN-based encoder model [2].

In this work, we study two related but disparate questions. First, we compare the quality of representations produced by Timenet and the CNN encoder. By comparing the strengths of each, we can make more informed decisions about which to use/work to improve in the future. Second, we measure the effect of training corpus size and diversity on the representation quality. Our hypothesis is that representations including more data and highly diverse data will help the encoder generalize the representation space.

There are several motivations for knowing how to construct a well-formed corpus for representation learning. A well-generalized representation space would allow us to train a single encoder on a well-formed training corpus and use this model to generate representations for many datasets and tasks. This would avoid the expensive need to train a new encoder for each domain. Further, it is possible more complex latent patterns may be learned from diverse data than homogenous data. To understand why, imagine that to learn timeseries representations, the encoder tries to learn the ”shape” of the data. Stock price timeseries may have one shape while temperature timeseries may have another. By training an encoder to learn both, it is possible patterns can be learned from the intersection of these ”shapes” that would not have been possible if learned in isolation. This overlap may lead the way to a more general understanding of timeseries by the encoder, giving its representations more context and power.

We begin by summarizing the algorithms we test in section II. Section III details our experimental procedure, with results following in section IV. We end by summarizing our results and providing direction for future work in sections V and VI.

II. Algorithm Overview

Timenet

Timenet is an encoder-decoder pair network, known as an autoencoder. It was introduced in 2017. It is called an autoencoder because it attempts to take an input sequence, shrink its dimension, and then reconstruct the inputs. If the network can be trained to do this well, then the encoder is capable of information distillation.

Architecture: The encoder is made up of one or more layers of recurrent cells, GRU in this implementation. RNN cells work well for this task because they are designed specifically to handle sequential data. The recurrent cell is able to ”remember” information from prior timesteps while processing each new timestep. The downside of this approach is that it requires

sequential computing, as opposed to parallelization. Dropout layers are placed between the individual GRU layers during training to encourage regularization and the robustness of the learned representations. The decoder is the same architecture as its encoder, but flipped around such that the outputs of the encoder are the inputs of the decoder. A linear transformation is applied to the outputs of the decoder to reconstruct the input timeseries.

Training: Timenet is trained to shrink a given timeseries into a smaller representation using the encoder, and then reconstruct the input timeseries using the decoder. This network uses mean-squared error between the input timeseries and the reconstructed series as its minimization objective and optimizes the network parameters through gradient descent.

Dilated Causal Convolutions

Introduced in 2019, this architecture uses only a CNN-based encoder. Rather than using reconstruction error to learn, this network learns using a novel adaptation of word2vec's skip-gram triplet loss.

Architecture: First, a few definitions. A 1D **convolution** layer learns a length k filter with c channels. This filter is slid across windows of the input in strides. For all windows the filter stops over, the dot product of the filter kernel and the input window is taken. The sequence of these dot products forms the output. This is repeated for c channels. Convolutions also take advantage of GPU parallelization, whereas recurrent layers cannot.

Dilation is a hyperparameter of the filter, 1 by default. A dilation of d means the individual cells of the filter are applied to inputs at indexes d apart. For example, a three cell filter will be applied at input indexes $i, i+d, i+2d$. This allows the network to learn sequence dependencies over a wider receptive field.

These dilated 1D convolutions are also **causal**. In this context, causal means the i^{th} output is computed using only inputs up to the i^{th} element. Because CNNs are not sequential like RNNs, the causal computation ensures each output is computed using only past values, analogous to the RNN mechanism for remembering past values.

This network is a stack of 1D causal convolution layers that are exponentially dilated. If the network has 4 layers, its layers will have dilations of 2^0, 2^1, 2^2, 2^3, respectively. This allows the encoder to learn both short and long-term temporal dependencies. The output of the last causal layer is aggregated using a max pooling layer, which squeezes the channel dimension down to 1, yielding shape $[output\_length, 1]$. This output is fed through a linear layer to get the final output representation.

Training: This network does not use a decoder, as its learning objective is not to minimize reconstruction error like Timenet. Instead, this network uses a novel adaptation of the skipgram with negative sampling triplet loss for timeseries. Triplet losses were first popularized by word2vec [3] in 2013, but had not been used for unsupervised timeseries representation learning until this network.

Triplet loss aims to minimize the representational differences between contextually similar timeseries and maximize the difference between out-of-context series. We do this by sampling triplets from the training dataset. Each triplet consists of a reference sample, a positive sample (in-context), and K negative samples (out-of-context). These are chosen such that the positive sample is a subsection of the reference, and the negative samples are from different timeseries instances. This loss function is then applied to the triplet:

$$-log(\sigma(f(x^{ref},\theta)^T f(x^{pos},\theta))) - \sum_{k=1}^{K} log(\sigma(-f(x^{ref},\theta)^T f(x_k^{neg},\theta)))$$

,

. In the notation, σ is the sigmoid function, $f(x,\theta)$ is the encoded output given a timeseries, and x is timeseries. Superscripts on x notate which triplet role that sample corresponds to. When x^{ref} and x^{pos} are similar, the first loss term will be small, minimizing the loss. When x^{ref} and x^{neg} are dissimilar, the second loss term will be large, also minimizing the loss. Similarities are computed as the dot product of the representation vectors, which corresponds to unnormalized cosine similarity.

III. Methodology

Source code for these architectures was found on Github. The CNN encoder source code was provided by the paper authors. While no official implementation of Timenet was provided by the authors, a suitable implementation written by someone else was used instead.

For each encoder architecture, we test the effect of training corpus size and diversity on representation quality. Because the representations are learned independent of a specific task, "quality" should be assessed over a variety of tasks. In this research, we focus on the quality of the representations for classification. Measuring quality for clustering tasks is also desirable, but will be left for future work.

All datasets used are from the UCR Timeseries Classification Archive [4].

Hyperparameters

Because optimal deep learning training regiments vary based on many variables, we employ early stopping heuristics for all encoder training cycles. If a model does not improve for three epochs, training ceases. This is an easy way to prevent overfitting and limit overall training time. We use batch sizes of 32 for training. Learning rate was selected per architecture based on what the original authors used.

We use the encoder source papers for guidance on all architecture hyper-parameters. These choices are seldom one-size-fits-all, but for clarity and convenience of testing, we train all tasks using these choices. For instance, a shallow network might learn short timeseries better than a deep network. While effects like these are important to keep in mind, we focus this study on the effect of training corpus composition on architectures with these hyper-parameters.

Timenet

Learning rate	0.006
Dropout rate	0.2
Channels	60
Recurrent layers	3
Encoding length	180

CNN Encoder

Learning rate	0.001
Negative samples	5
Channels	40
Convolutional layers	10
Kernel size	3
Encoding length	160

A. Size

We tested sizes 50, 100, 500, and 1000. Note that because our chosen batch size is 32, a dataset of 50 samples will only have one full batch per epoch. While models trained on a larger size dataset will train for more batches in absolute terms, all models will still only see each sample once per epoch.

For each size, we trained the encoders on five UCR datasets, one at a time. The datasets used are *ElectricDevices*, *EthanolLevel*, *Phoneme*, *ECG5000*, and *Earthquakes*. These datasets were chosen because they vary in timeseries length and number of classes. Overall, $n\_sizes * n\_datasets * n\_architectures = 40$ models were trained. For each unique size-dataset pair, we sampled $size$ samples from $dataset$. We then train each encoder on this sample. After training, we use the model to encode each of the five datasets for testing. Next, we train an SVM to classify the encodings and record classification accuracy, precision, recall, and f1.

Because we test each model's representation quality on unseen datasets, we can try to measure gains in transferability related to corpus size. For instance, if the classification accuracy of dataset A encodings gotten from an encoder trained on dataset B increases with training corpus size, we know transferability increases with corpus size.

B. Diversity

Here, diversity refers to the number of different datasets included in a training corpus. A diversity 5 corpus will be sampled from 5 unique datasets. Diversities of 1, 3, 5, 10, 20, and 50 are tested. Each corpus uses 500 samples, such that the more diverse a corpus, the more diluted each of its constituent datasets. For each diversity d, we choose d unique datasets, then sample $\frac{500}{d}$ instances from chosen dataset. Because the dataset choosing process is done by random sampling, we repeat this process three times for each diversity level. In the case that one dataset sampling is particularly good or bad, repeated this process helps get a better idea of the average effect of diversity. Results reported for any diversity level are the average of these three trials. Using this protocol, $n\_diversities * n\_reps * n\_architectures = 36$ models are trained.

C. Holistic Models

To evaluate the full potential of these architectures, we train them without placing limits on the training corpus like we do for the size and diversity procedures. Each architecture is trained on a much larger diverse dataset to do this. This dataset is both large and diverse, with *n* samples from 15 distinct UCR datasets. Datasets were chosen such that timeseries lengths were less than 512.

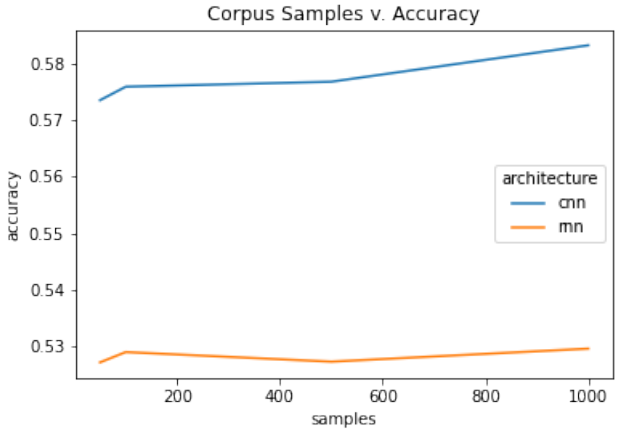

Fig. 1. Aggregated corpus size vs. classification accuracy

TABLE I
AGGREGATED CNN SIZE EXPERIMENT RESULTS

samples	accuracy	f1
50	0.57	0.51
50	0.53	0.45
100	0.58	0.51
100	0.53	0.46
500	0.58	0.52
500	0.53	0.46
1000	0.58	0.52
1000	0.53	0.46

IV. Experimental Results

We present the results of the procedures described in section III in the same order as they are described.

A. Size

Figs 1 and 2 show how classification accuracy and F1 relate to different corpus sizes, respectively. These results are aggregated over all test datasets. Although CNN encoders show a very small increase in accuracy and F1 as corpus size increases, Timenet shows no such improvement. On the dataset-aggregate level, there appears to be no significant relation between corpus size and classification performance.

This effect remains when we analyze this relationship by training dataset, although there are some cases where the correlation between corpus size and performance is slightly positive or even negative. Tables I and II show the full results. The aggregated results seem low because they are averaged over the results from diverse datsets, some with high classification scores and others with low.

We also analyze the classification performance difference between encoders trained on the test dataset and those that

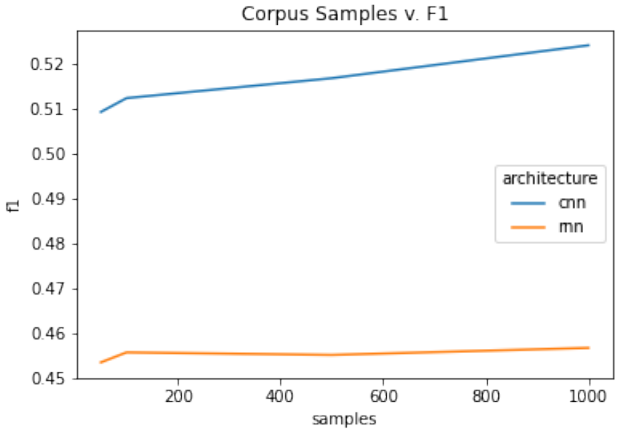

Fig. 2. Aggregated corpus size vs. classification F1

TABLE II
AGGREGATED TIMENET SIZE EXPERIMENT RESULTS

samples	accuracy	f1
50	0.53	0.45
100	0.53	0.46
500	0.53	0.46
1000	0.53	0.46

have not seen the test dataset. There was variation in these results between architectures and datasets. We refer to encoders trained on the test dataset as "own-encoders" and those that were not as "other-encoders." Table III shows the ratio between performance values, where the values of own-encoders are the numerator. Values greater than 1 mean the own-encoders performed better than other-encoders, and vice-versa. There was no difference between performance on *ECG5000* and *Earthquakes*. We also observe that CNN own-encoders did better than other-encoders on *ElectricDevices* and worse on *EthanolLevel*. The results for Timenet were the opposite of the CNN encoder, with own-encoders doing better on *EthanolLevel* and worse on *ElectricDevices*.

These results show that other-encoders can produce representations on unseen data almost as high-quality as own-encoders. This indicates that these encoders are able to learn dataset-agnostic timeseries features and generalize.

The exception to this is the performance of the CNN own-encoder on the *Phoneme* dataset. Phoneme is a harder dataset to classify, so the CNN own-encoder benefited immensely from having seen examples of it before.

B. Diversity

The results from the diversity experiments were also unexpected. Figs 3 and 4 show the relationship between diversity and classification performance after aggregating by test dataset results. Performance peaks at diversity 1 for both architectures. CNN models trained on diversity 5 datasets did better than diversities 3, 10, and 20. Timenet performance did not differ for all diversities above 1. Full aggregated results can be found

TABLE III
OWN VS. OTHER ENCODER CLASSIFICATION PERFORMANCE

architecture	test_dataset	accuracy_ratio	f1_ratio
cnn	ElectricDevices	1.09	1.12
cnn	EthanolLevel	0.96	0.96
cnn	Phoneme	1.53	2.33
cnn	ECG5000	1.00	1.00
cnn	Earthquakes	1.00	1.00
rnn	ElectricDevices	0.98	0.95
rnn	EthanolLevel	1.04	1.40
rnn	Phoneme	0.96	0.89
rnn	ECG5000	1.02	1.02
rnn	Earthquakes	1.02	1.03

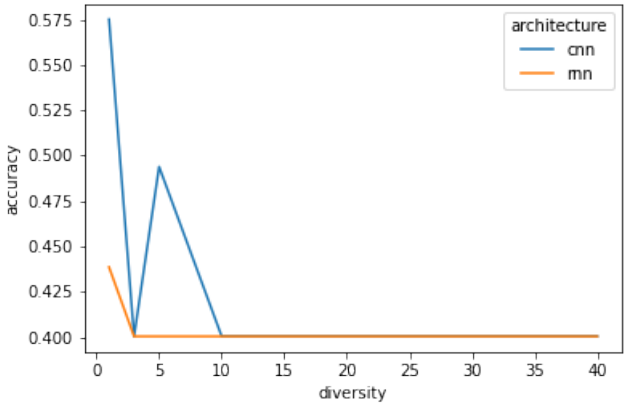

Fig. 3. Corpus Diversity v. Classification Accuracy

in tables IV and V. Like in the size experiment results, these aggregated results seem low because they are averaged across datasets of varying classification difficulty.

Looking at the unaggregated results from the three corpus selection trials for each diversity level, we see why there are peaks at diversities 1 and 5 for CNN encoders. The peak at diversity 1 is caused by all three trials having above average

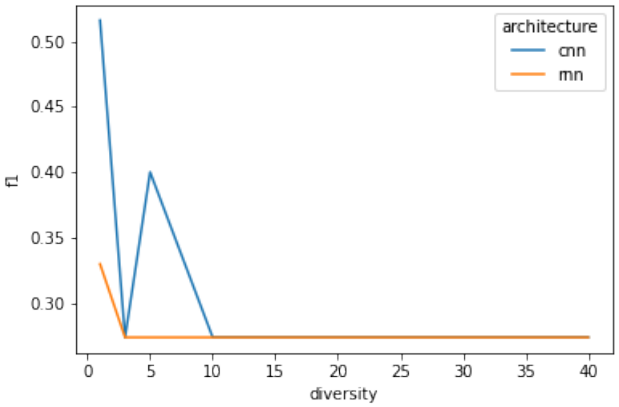

Fig. 4. Corpus Diversity v. Classification F1

TABLE IV
AGGREGATED CNN DIVERSITY EXPERIMENT RESULTS

diversity	accuracy	f1
1	0.58	0.52
3	0.40	0.27
5	0.49	0.40
10	0.40	0.27
20	0.40	0.27
40	0.40	0.27

TABLE V
AGGREGATED TIMENET DIVERSITY EXPERIMENT RESULTS

diversity	accuracy	f1
1	0.44	0.33
3	0.40	0.27
5	0.40	0.27
10	0.40	0.27
20	0.40	0.27
40	0.40	0.27

results. However, the peak at diversity 5 can be explained by a single model that did better than average. This implies large variation in corpus qualities, even within the same diversity level. This means that the datasets included are more important than the number of datasets sampled from.

However, no such pattern is seen in the Timenet results, so this effect speculative. These results are also in contradiction to the Timenet authors' claim that Timenet's performance increased with corpus diversity.

Interestingly, certain performance scores were shared exactly between several models of different diversity. For example, 0.58 was the accuracy score of the *ECG5000* dataset for several CNN and Timenet models, across diversity levels. These discretized accuracy values indicate a clear division between timeseries instances based on classification difficulty. This was observed for multiple datasets and also applies to precision and recall.

C. Holistic Models

Among the encoders trained on the large and diverse dataset, SVMs trained on encodings from the CNN-based encoder outperformed Timenet on 91.5% of 117 test datasets. The CNN model also trained in 15 minutes, only 2.9% of the 8 hours it took to train Timenet. Both of these encoders outperformed the average scores of their respective diversity-leveled counterparts. Tables VI and VII show the results for the datasets included in the size and diversity experiments.

Surprisingly, the training time for the CNN holistic encoder was less than several of the CNN encoders from the diversity experiment, despite the training corpus size being much larger for the holistic encoder. Upon closer inspection, we found this was caused by the length of the datasets chosen in the diversity section. In the holistic model training corpus, timeseries were capped by a length of 512. In the diversity experiment, timeseries of any length could be chosen, including those that are over 10,000 measurements long.

TABLE VI
CNN HOLISTIC ENCODER CLASSIFICATION PERFORMANCE

test_dataset	accuracy	f1
EthanolLevel	0.24	0.16
Phoneme	0.22	0.14
ECG5000	0.93	0.91
ElectricDevices	0.73	0.71
Earthquakes	0.80	0.71

TABLE VII
TIMENET HOLISTIC ENCODER CLASSIFICATION PERFORMANCE

test_dataset	accuracy	f1
EthanolLevel	0.26	0.16
Phoneme	0.11	0.02
ECG5000	0.88	0.85
ElectricDevices	0.27	0.15
Earthquakes	0.80	0.71

This observation may explain the low classification scores from the diversity section. Very long timeseries may hinder encoder performance, especially if learned in conjunction with shorter timeseries. Because an encoder trained on both short and very long timeseries must learn to transform both to a fixed-length vector, the encoder may not be doing either as well as it could.

We also visualize the CNN encodings of various datasets in 2D space using the t-SNE method. This gives us an idea of what the representation space of a particular dataset looks like. Fig 5 visualizes six datasets in t-SNE transformed representation space, where each dot represents a timeseries instance and colors represent dataset class.

V. CONCLUSION

Confusingly, encoders trained on 50 samples did about as well on average as those trained on 1000. This result holds for both encoder architectures. The initial conclusion is that the encoder learns the most important information early in training, but it is usually expected further training will still yield marginal improvement. This was not observed to be the case here. This would indicate that corpus size does not have an impact on encoder representation quality.

Similarly, models trained on a corpus comprised of a single dataset did better than those trained on a mixture of datasets. This shows that there is slight negative relationship between corpus diversity and representation quality, which was the opposite of our hypothesis. A potential explanation for this trend is that increasing corpus diversity without increasing sample size confuses the encoder, leading to a decrease in performance. A more likely explanation is the observation discussed in section IV.C, that very long timeseries confuse encoders being jointly trained on timeseries of various lengths. Because of this discovery, we can make no conclusions about the effect of corpus diversity on representation quality. To remedy this, the diversity experiment should be redone such that timeseries lengths are capped at 512.

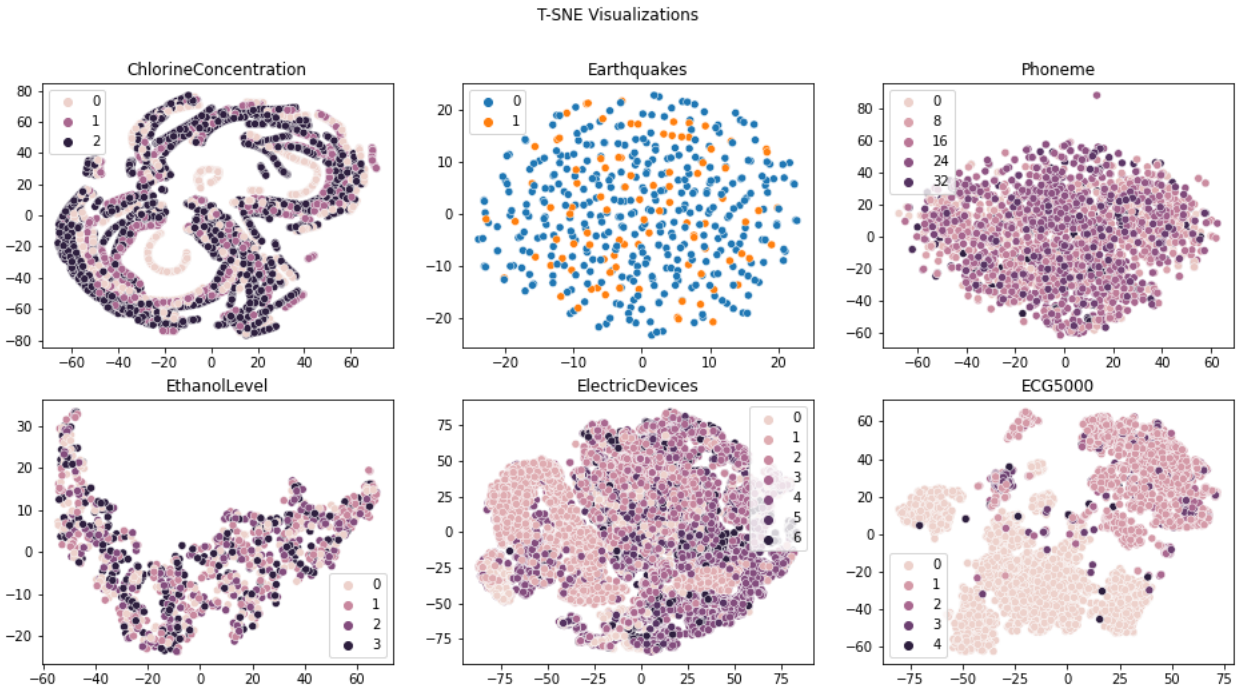

Fig. 5. t-SNE representation visualizations

We are also able to corroborate the authors of the CNN encoder paper's claim that their encoder outperforms Timenet in both terms of time and classification efficiency, making it the better representation learning architecture using these measures. We also made a bug fix contribution to the CNN author's source code [5]. We were also able to contribute a change to the Timenet Github repository used [6]. This change makes the decoder learn to reconstruct the input timeseries backwards, per the Timenet paper.

Overall, our research represents an exploratory first step in the domain of corpus composition for unsupervised timeseries representation learning.

VI. Future Work

This work tested the effect of size and diversity on training corpus effectiveness, but there is more that can be studied relating to corpus composition. Here, we considered datasets "diverse" if they contain different constituent datasets. However, this is a limited view as different datasets may be more similar than others. Another measure of diversity would include dataset type. The UCR timeseries repository categorizes datasets by image, spectro, sensor, simulated, device, motion, among others. It is plausible that these categories have higher intracategorical similarity than intercategorical similarity. If so, it is worth experimenting with forming corpora on a categorical basis rather than dataset basis. Further research could determine the relative effectiveness of dataset diverse corpora compared to dataset-specific corpora and categorically diverse corpora compared to categorically-specific corpora. Other potential proxies for diversity include timeseries length and number of classes.

Another avenue of exploration is the effect of dataset "difficulty" in training corpora. We use state-of-the-art classification accuracy here for our notion of difficulty. An "easy" dataset would be one that is easily classifiable, and a "hard" dataset is one that is difficult to achieve a high classification accuracy. It is plausible that balancing corpus construction between "easy" and "hard" datasets would increase representation generality. Conversely, it is also possible that "hard" datasets are hard because they are noisy, which may hinder corpus effectiveness if included.

Plotting encoder learning curves may provide useful insight for the results presented in this paper. Measuring the marginal benefit of training for an additional epoch may show some corpus treatments peak early, while others peak late.

This research would also benefit from the context of existing literature on curriculum learning for deep neural networks [7].

References

[1] Pankaj Malhotra, Vishnu TV, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Timenet: Pre-trained deep recurrent neural network for time series classification. 2017.

[2] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *CoRR*, abs/1901.10738, 2019.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[4] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. https://www.cs.ucr.edu/ eamonn/time$_{s}eries_{d}ata_{2}018/$.

[5] Franceschi. Unsupervisedscalablerepresentation-learningtimeseries. https://github.com/White-Link/UnsupervisedScalableRepresentationLearningTimeSeries, 2019.

[6] Danenas. Timenet. https://github.com/paudan/TimeNet, 2020.

[7] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML 09*, 2009.