Topological Simplification of Signals for Inference and Approximate Reconstruction

Gary Koplik, Nathan Borggren, Sam Voisin,
Gabrielle Angeloro, Jay Hineman, and Tessa Johnson
Geometric Data Analytics
gary.koplik@geomdata.com

Paul Bendich
Geometric Data Analytics
Department of Mathematics
Duke University
paul.bendich@geomdata.com

Abstract— **As Internet of Things (IoT) devices become both cheaper and more powerful, researchers are increasingly finding solutions to their scientific curiosities both financially and computationally feasible. When operating with restricted power or communications budgets, however, devices can only send highly-compressed data. Such circumstances are common for devices placed away from electric grids that can only communicate via satellite, a situation particularly plausible for environmental sensor networks. These restrictions can be further complicated by potential variability in the communications budget, for example a solar-powered device needing to expend less energy when transmitting data on a cloudy day. We propose a novel, topology-based, lossy compression method well-equipped for these restrictive yet variable circumstances. This technique, *Topological Signal Compression*, allows sending compressed signals that utilize the entirety of a variable communications budget. To demonstrate our algorithm's capabilities, we perform entropy calculations as well as a classification exercise on increasingly topologically simplified signals from the Free-Spoken Digit Dataset and explore the stability of the resulting performance against common baselines.**

Table of Contents

1. Introduction

Time series arise whenever numerical values are collected over time. Time series classification involves training a model on a set of labeled time series signals, and then using the model to predict those labels on a test set. Applications of time series classification abound: for example, using signals of labeled EEG time series to predict [1] epileptic brain activity. Multiple methods [2] for time series classification exist in the literature, including a variety [3] of deep learning techniques.

This paper explores a novel lossy compression technique, *Topological Signal Compression* (TSC). Figures 2 and 3 demonstrate this technique. Illustrative results are shown using the Free-Spoken Digit Dataset (FSDD)[4]. The key findings of this paper appear in Figures 5, 6, and 8, which show that TSC both preserves information content and can maintain classifier performance while significantly reducing the size of signal needed to achieve said performance. Furthermore, TSC achieves this capability in an interpretable way at arbitrarily high compression levels and with compression on the margin having only a local effect on the reconstructed signal.

Motivation

We are motivated by the following abstraction of a common paradigm: we imagine that the time series signals to be classified are collected by any one of potentially many edge devices, and that the classification itself must happen at a central device.[2] In addition to classification accuracy, we will also judge success based on the amount of transmission between the edge devices and the central device.

Examples of this paradigm include: a) the Internet-of-things (IoT), where on-device power constraints or a low-bandwidth communications network can preclude the transmission of full signals to the central node; b) surveillance applications, where excessive transmission between a drone and a central computer increases the chances of counter-surveillance measures detecting and thus disrupting the classification process. In general, we call these *constrained communications* (CC) scenarios.

The Topological Signal Compression (TSC) algorithm proposed in this paper serves as a generic lossy compression step useful in any time series compression and / or classification task that must take place under constrained communications. TSC is adaptable to levels of CC, as the algorithm permits transmission at exactly the level of transmission permitted by the scenario. Furthermore, TSC is stable to changing levels of CC. On a theoretical level, TSC performs a localized and thus more stable compression when removing more points—if one were to remove an additional point from a signal already compressed via TSC, then the resulting signal reconstruction would only be affected around the removed point. This stability is further demonstrated experimentally in this paper with the graceful degradations shown in both classifier performance and entropy levels as the CC level increases as well as with a Dynamic Time Warping (DTW) analysis comparing compressed then reconstructed signals to the original signals.

We contrasted TSC with several additional lossy compression methodologies in this paper—the Opus [18] codec, the Discrete Fourier Transform (DFT, [19]), and Piecewise

[2] Although there are plenty of cases where one would be inclined to resolve classification on the edge as opposed to at a central device, this is not always possible, for example, if a task required information from *multiple* devices before classification. One may not even be interested in classification at all, instead focusing on returning as much relevant raw signal data as possible.

978-1-6654-9032-0/23/$31.00 ©2023 IEEE

Aggregate Approximation (PAA, [20]). Opus is an audio-specific codec whereas TSC generalizes to compressing any time series data. Given our choice of an audio dataset tailored to the narrow strengths of Opus for classification tasks, we found in our experimental analysis that Opus better-maintains performance over increased compression levels and noise than TSC. Unlike Opus, however, TSC is also flexible to any precise compression cutoff, whereas Opus is harder to use to compress a signal to a specific byte size. Furthermore, when using Opus in practice on the Free-Spoken Digit Dataset, we were unable to compress beyond roughly 90% of the original size of a signal, whereas TSC can be run at arbitrarily high compression percentages. Finally, in our machine learning exercises, we found only TSC and Opus maintained both accuracy and stability with respect to higher levels of compression as well as when noise was added to the data. TSC was thus the only algorithm considered in this paper that had the union of generalizability, flexibility, and stability that we believe to be important in a highly-compressed data transmission scenario with a variable communications budget.

Compression With a Variable Communications Budget

Consider a sensor network where a given edge device can send no more than a small but ever-changing quantity of bytes of collected information to the central device at a moment in time. As a more concrete example, consider the use case of IoT over the ocean in the *Ocean of Things (OoT)* project [5]. Since the raw data itself is of high value on this project, one would not want to only send summary statistics or any sort of classification results alone, as the time series data itself is of great value. For example, sending high-resolution spatiotemporal ocean environmental data would be valuable to oceanographers in evaluating and improving ocean models. Additionally, for floats to be able to send data from anywhere in the ocean, the only viable means of data transmission is via satellite, which drastically shrinks the maximum possible communications budget to only a few hundred bytes.[3]

Although there may be a fixed bandwidth constraint, the possibility of floats clustering would require at times having a more restrictive communications budgets for each float.[4] Moreover, since these floats are capable of collecting and reporting a range of multimodal data products of variable relevance for different use cases, the triage to prioritize which data to send will lead to situations where even for a fixed communications budget, there will be a variable *remaining* budget to send information from a given modality.

TSC addresses these circumstances for the OoT use case. For any spare space in a given communications budget that would be wasted otherwise, one could simply run TSC to return more points from a signal that use the exact number of remaining available bytes, a task that cannot easily be achieved by Opus or Piecewise Aggregate Approximation. As for triaging which data to send, TSC could update dynamically to requests by a human or an automated anomaly detection algorithm to prioritize sending more of one signal at the expense of another. Since TSC generalizes to any modality of signal data, one can simply revise each modality's byte constraint and then run TSC for all the modalities. One could even factor in environmental constraints. For example, if devices were solar-powered, and battery levels were low on a cloudy day, signals could be compressed more than normal to save power by transmitting fewer bytes, preventing the devices from temporarily running out of battery power. Though one would have the same flexibility with the Discrete Fourier Transform, the instability of the compressed data, particularly at higher levels of compression, would make inference between signals variably-compressed with DFT difficult.

Outline

The rest of this paper proceeds as follows. Persistent homology and its use in Topological Signal Compression is discussed in Section 2. Then Section 3 introduces the dataset used for illustration, with machine-learning and entropy experiments described in Section 4. The paper discusses how TSC can be used in practice relative to several competing compression methodologies in Section 5, and concludes in Section 6.

2. Topological Signal Compression

The typical mathematical model of a one-dimensional signal is a real-valued function f on a closed interval $[a, b]$, but for this work we imagine that the interval has been sampled at a discrete set of time points $a = t_0 < t_1 < t_2 < \ldots < t_n = b$, and so f is given by its values at each of these time points. This section outlines the *persistence diagram* summary of f, and then describes our proposed scheme for using persistence diagrams to transmit parsimonious approximations of f.

Persistent Homology

The signal f is summarized by its *zero-dimensional persistence diagram* ([7] [8]) $D_0(f)$. Intuitively, one need only understand the following in the context of a signal. Here, persistent homology tracks connected components as we sweep a horizontal line vertically from negative infinity to positive infinity. Components are *born* at local minima, and *die* at local maxima, destroying the more recently born component of the two components merging. The diagram $D_0(f)$ plots the births and deaths of components as dots in the plane. The vertical distance of a dot above the 45 degree line birth = death represents the *persistence* of a component within the filtration. See Figure 1 for an example of a signal and its corresponding persistence diagram.

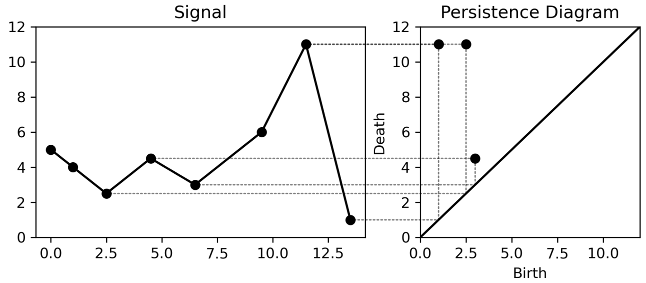

Figure 1. On the left a signal with 8 points is shown. The Persistence Diagram is shown on the right and formed by sweeping upwards. Components are born at local minima and die at local maxima, destroying the more recently born component. We follow the convention of pairing the global minimum and maximum.

Persistent homology thus gives us an *ordering* on our connected components. Dots on the persistence diagram that

[3] Iridium Short Burst Data, for example, has a transmission budget constraint of 340 bytes [6].

[4] A single satellite can only process so many messages at one time. Thus, if 1000 floats are in a cluster trying to report to one satellite at one moment in time, they will likely strain the communications channel at that moment, even if each float is transmitting within its original communications budget.

are close to the diagonal die soon after being born. By allowing us to identify low-persistence components, persistent homology shows us the parts of the signal most likely corresponding to noise. More precisely, persistence diagrams enjoy a *stability theorem* ([9],[8]) that states, roughly, that diagrams corresponding to functions which are small perturbations of each other will differ mostly by the presence or absence of low-persistence dots. We note that some results (e.g, [10]) have shown these low-persistence dots to still have classification power in machine learning applications.

Topological Simplification

Inspired by the typical situation where dots of low-persistence correspond to noise in the signal, our simplification method enables the reconstruction of the signal as it would be without the values of least persistence, thus keeping the more prominent features of the signal at the expense of noise. Figures 2 and 3 show examples of this simplification on synthetic and real data, respectively.

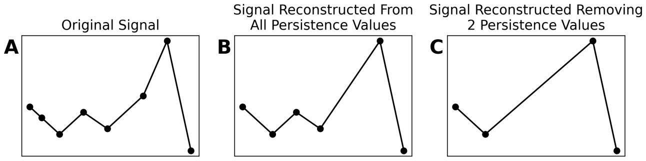

Figure 2. An example of running our topological simplification algorithm. (A) is our original signal, borrowed from Figure 1. (B) is the baseline topologically simplified compression, which keeps only critical points, thus dropping the two non-critical points in the signal. (C) drops the two further points corresponding to the smallest persistence value on the persistence diagram, which can be validated comparing against Figure 1.

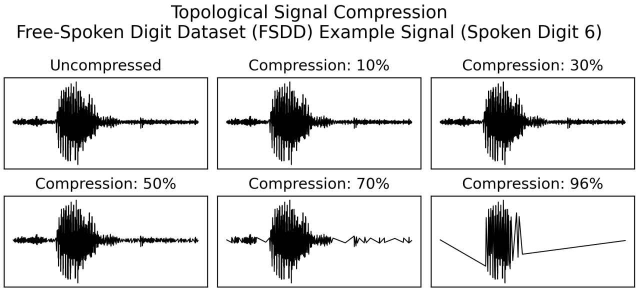

Figure 3. Example from the Free-Spoken Digit Dataset (FSDD) of a person speaking the integer 6, shown with increasing levels of topological simplification.

The theoretical force behind the algorithm is the *Morse Cancellation Lemma* (MCL, [11], and also see [12] for a self-contained proof). In the simplest version, which is all that we require here, the MCL states that, if $\mathbf{u} = (b, d)$ is the dot of lowest persistence in the diagram $D_0(f)$ of a one-dimensional signal f, then there exists a signal g defined on the same domain whose persistence diagram $D_0(g)$ is exactly the same as $D_0(f)$ except that $\mathbf{u}$ has been removed. For example, panel C of Figure 2 shows one such g corresponding to the signal f in panel B of the same figure.

In effect, g is formed from f by "un-kinking" the pair of critical points of lowest persistence. The reason that this action does not cause global change in the signal stems from the MCL[5], which guarantees that the dot of lowest persistence corresponds to a pair of *horizontally adjacent* critical points in the time domain, thus guaranteeing that un-kinking that pair of critical points will not un-kink any other pair. The MCL can of course be applied iteratively, as demonstrated in Figure 3.

Bauer et al. [13] have noted a relationship between persistence-based simplification in one dimension and total variation-based denoising. Simplification via the MCL extends as well to functions defined on two-dimensional domains (Edelsbrunner et al [14] give an algorithm for this two-dimensional simplification), but there are theoretical issues with higher-dimensional domains. More relevant to the current work, the reader may have noted that there are in fact infinitely many functions g whose persistence diagrams are the required simplifications of $D_0(f)$. For example, the long diagonal line in panel C of Figure 2 could be replaced by any monotonically increasing function defined on the same subinterval without affecting the persistence diagram. Several works address ways to choose the "right" type of g; for example, Poulenard et al [15] give a general technique for finding a g that satisfies a user-specific cost function. The perspective we take in this paper is that we simply transmit the $(t, f(t))$ values needed for the central device to make this choice. Note that all illustrations and machine learning tasks throughout this paper perform piecewise linear interpolation when reconstructing signals.

3. Data

We used the Free-Spoken Digit Dataset (FSDD) created by Zohar Jackson, César Souza, Jason Flaks, Yuxin Pan, Hereman Nicolas, and Adhish Thite [4]. This audio dataset was produced by six male speakers and consists of recordings of spoken digits 0 through 9. The 3000 examples were recorded at 8kHz, with each person recording 50 samples of each digit. Finally, the samples are trimmed to have minimal silence at both the beginning and end of each audio clip.

4. Machine Learning and Entropy

For our baseline machine learning exercise, we classified FSDD spoken digits using Mel-Frequency Cepstrum Coefficient (MFCC) featurizations run through a Convolutional Neural Network (CNN) with 5-fold cross-validation. The MFCC featurization performs a short-time Fourier analysis with a binning scheme motivated by the anatomy of the human ear. For more on MFCC, see [16] and [17]. By this process, each sample was transformed into a 20 x 16 feature vector. For this initial exercise, we achieved a mean cross-validated accuracy of approximately 97%. The confusion matrix for these baseline results is reported in Figure 4.

We then repeated this cross-validated classification pipeline, only we built the MFCC features using increasingly topologically simplified signals from FSDD. We demonstrate various levels of topological simplification on an example signal from FSDD in Figure 3.

In addition to TSC, we considered four competing forms of

[5] Technically, this lemma requires that the function values of the two neighbors of any given critical point be distinct. If this assumption fails, we could create non-unique solutions; however, if we sort first by persistence and then, for example, by index value of the points as input into the algorithm, the output will still be both correct and consistent, even though it may not be unique.

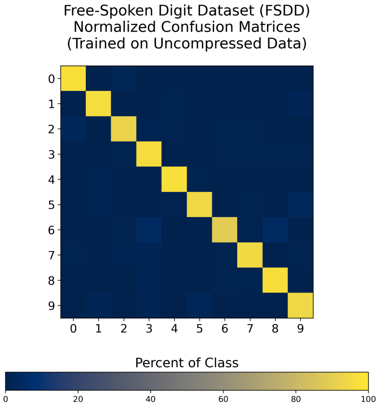

Figure 4. Confusion matrix for classification of Free-Spoken Digit Dataset (FSDD) digits with no compression of the data before featurization.

lossy compression to act as counterfactuals. We first considered Opus, a codec designed exclusively for lossy audio compression [18]. We then compressed signals using the Discrete Fourier Transform (DFT) by representing a signal with its Fourier coefficients and then sending only a subset of those coefficients to reconstruct the signal [19]. Next, we compressed signals using Piecewise Aggregate Approximation (PAA), which uniformly partitions a signal using a fixed window size and returns the mean function value within each window [20]. Finally, as the most naive control, we considered "Random Compression," where a random subset of $(t, f(t))$ pairs are returned as a lossy compression. For a comparison of how these competing compression methods affect FSDD data relative to TSC as shown in Figure 3, see Figure 12.

Our cross-validated accuracy results are reported in Figure 5. As a reference point for if we were solely concerned with machine learning accuracy as opposed to the ability to reconstruct a signal, we also included "Uncompressed" results in black, which considers the "compression" to be sending only MFCC featurizations, where we varied the number of MFCC coefficients returned. As expected, when we sacrifice returning any sort of signal to only return features, we are able to maximize machine learning performance at higher compression levels.

As for the lossy compression schemes considered in Figure 5, we see that Opus dominates in machine learning accuracy over increasing levels of compression, but this success requires a couple of qualifications. First, given the sole focus of Opus on audio compression, it is unsurprisingly the most successful compression method of the five methods considered here given FSDD is an audio dataset. The other four compression methods considered, including TSC, are agnostic to the type of signal data. Second, using the minimum level of bitrate compression possible, Opus was unable to achieve an average compression on FSDD data greater than 90%, thus limiting its potential ability to be used in highly constrained communications scenarios.

Once we reach a compression of greater than 90%, at which point Opus is no longer an option for this dataset, TSC-compressed data maintains a mean cross-validated classification accuracy of roughly 15-25 percentage points better than the competing compression methods.

We then explored accuracy for the compression methods on each of the 10 labels in FSDD. The confusion matrices in Figure 6 illustrate that machine learning on TSC-compressed data better maintains within-label classification accuracy at 90%+ compression relative to the non-Opus compression methods, as demonstrated by the more pronounced diagonal structure in the TSC confusion matrices.

To abstract away from the potential variation in performance due to the machine learning training methodology, we also explored changes in entropy, which allowed us to measure the average amount of information lost over increasing levels of lossy compression for each method. We utilized the approximate entropy algorithm developed in [21] and expanded upon in [22] to measure average entropy across samples of FSDD for increasing compression levels, with the results shown in Figure 8. Relative "performance" here is mostly consistent with machine learning classification accuracy, with the most notable exception being TSC maintaining comparable entropy to Opus at up to 70% compression.

Robustness to Noise

To test these compression methodologies' capabilities on noisier data, we first mean-centered and standardized each signal in the dataset. We then added increasing levels of Gaussian noise to the standardized dataset. Finally, we compressed the noised signals with each compression methodology, after which we ran them through the featurization and machine learning pipeline the same as before. The resulting machine learning classification accuracies are shown in Figure 7.

Although classification accuracy declines as more noise is added to the dataset, our results show consistent relative accuracy between the 5 compression methods for the unaltered signals and the noisy signals with up to 2.5 times the noise of the standardized signals added to the dataset, with one major exception being DFT compression performing noticeably worse with more noise. Although TSC's performance suffers more than Opus with increasing levels of noise, TSC still outperforms the other compression methodologies at greater than 90% compression.

5. Discussion

In addition to its superior machine learning performance at higher levels of compression with FSDD, Topological Signal Compression offers several additional benefits over the informed[6] competing compression methods.

Interpretability When Changing the Compression Parameter

We find TSC's underlying parameter for increasing compression, *persistence*, to be more interpretable in its effect on the compressed signal than the other competing compression methodologies.

For Opus, changing the bitrate appears to have ambiguous effects on the $(t, f(t))$ pairs at higher levels of compression, though it does track well with the signal overall, as demonstrated in Figure 9. TSC, on the other hand, precisely recovers the critical points it keeps, which may be important for downstream use and interpretation of compressed signals.

[6] The Random Compression scheme is highly interpretable, but uninformed by design as it compresses. With Random Compression serving mainly as a baseline control, it will be excluded from further discussion in this section.

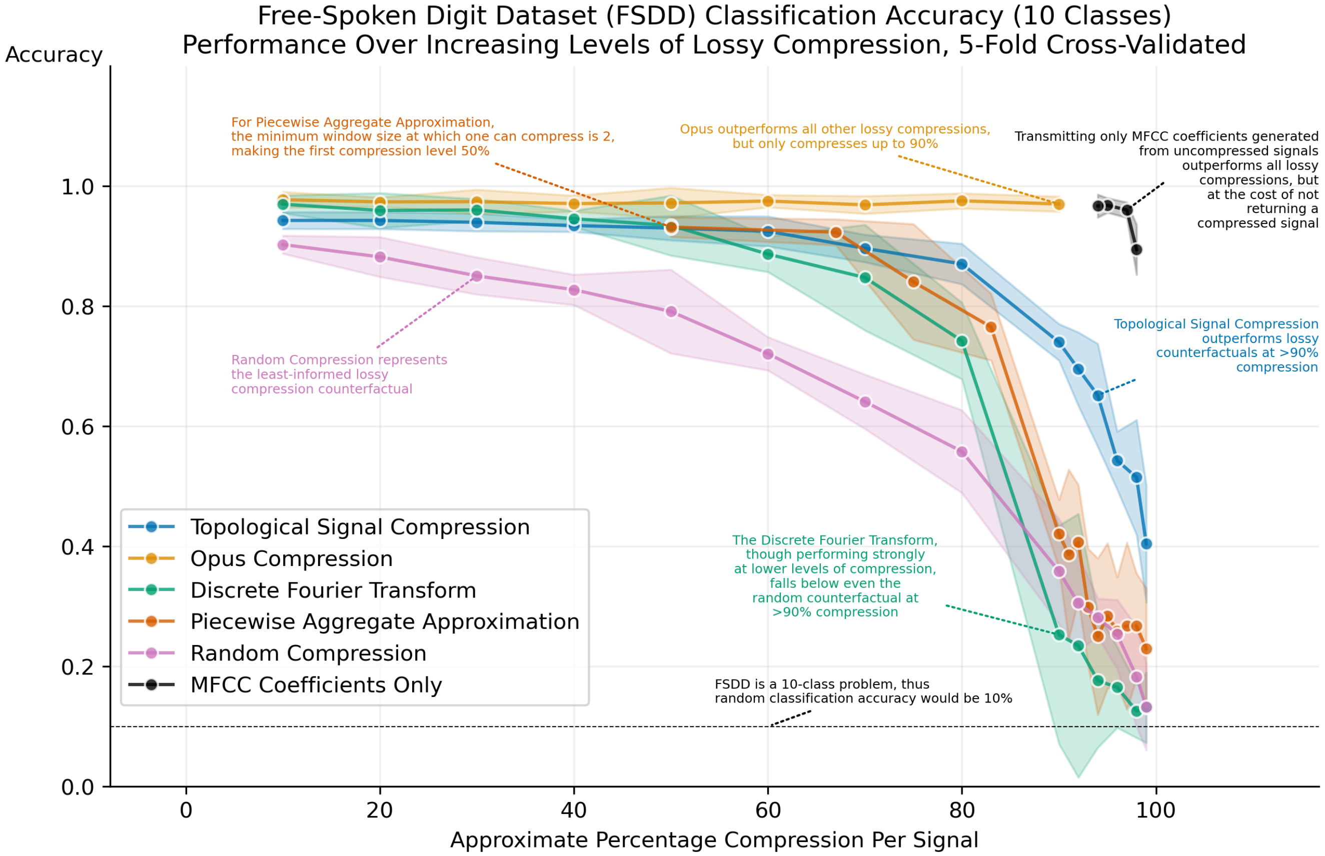

Figure 5. Accuracy for 10-label classification task with Convolutional Neural Network using MFCC featurizations generated from increasingly compressed signals. Error bars represent 2 times the standard deviation of accuracy over the 5-fold cross-validated results at each compression level.

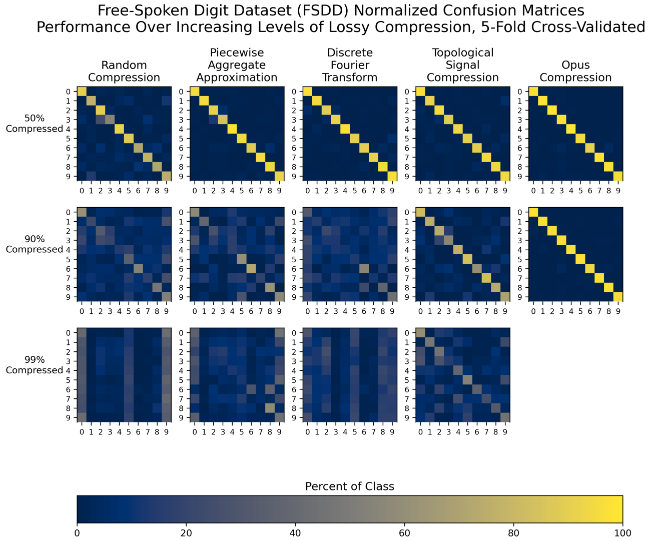

Figure 6. Confusion matrix for classification of 10 digits of selected compression percentages for our 5 compression methodologies.

For PAA, summarizing a partitioning of windows over the signal sacrifices within-window distributional information. For example, skewed data in a window would result in taking a skewed mean value, and even if we take the median instead, we would remove the skew information content in the window. Furthermore, any repeating pattern risks washing out in a windowed summary statistic even at relatively low levels of compression. For example, a regularly sampled sine curve with window size of 2π would result in a compressed signal of only 0 values. Although TSC will remove distributional information contained in non-critical points, it will otherwise preserve critical point distributional information. Additionally, repeating patterns will be preserved with TSC as long as they are sufficiently persistent.

For DFT, though it is excellent at preserving repeating patterns by its design, removing a Fourier coefficient does not have a strong intuitive implication for the resulting reconstructed signal, whereas removing a persistence pair with TSC has a localized, marginal effect on the compression, discussed further in the next sub-section.

Localized "On-the-Margin" Compression

For all of our competing compression methods, tweaking the main parameter (bitrate for Opus, window size for PAA, and number of Fourier Coefficients for DFT) results in a *global* change to the reconstructed signal. For TSC, however, we have an obvious means to make a *marginal* change to the reconstructed signal. Starting from a given reconstruction,

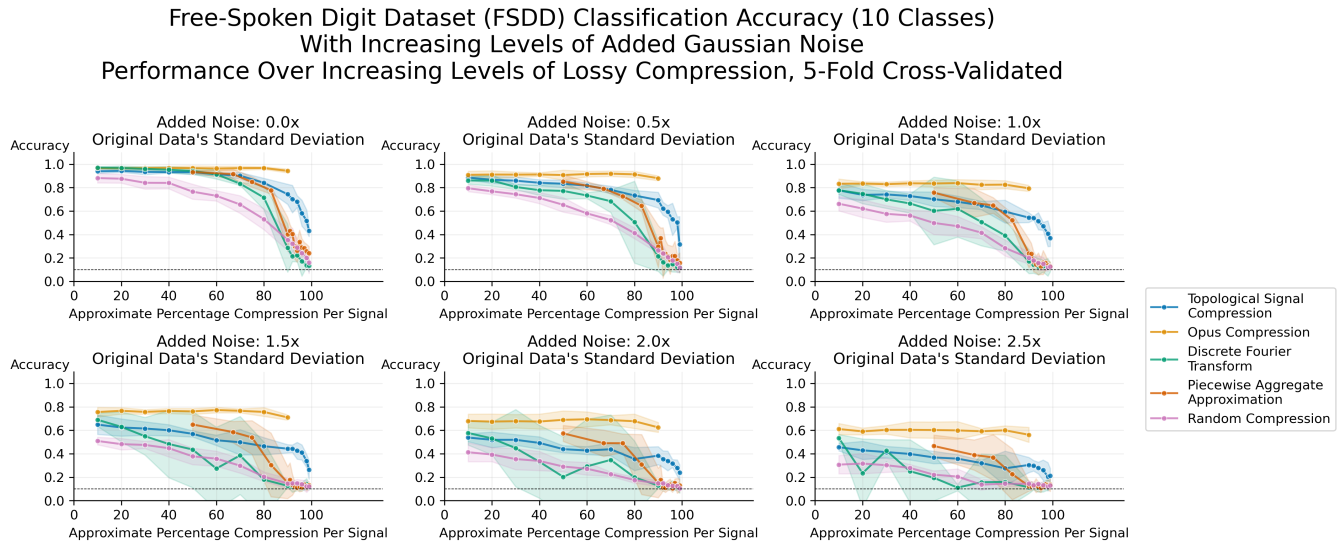

Figure 7. Accuracy of various compression methodologies over increasingly noisy FSDD dataset. Gaussian noise as high as 2.5 times the noise in the standardized signals was added to the dataset. Accuracy declines overall as noise increases, but *relative* performance between compression schemes is mostly consistent with Figure 5, with one major exception being DFT compression performing noticeably worse with more noise. Although TSC's performance suffers more than Opus with increasing levels of noise, TSC still outperforms the other compression methodologies at greater than 90% compression. Error bars represent 2 times the standard deviation of accuracy over the 5-fold cross-validated results at each compression level.

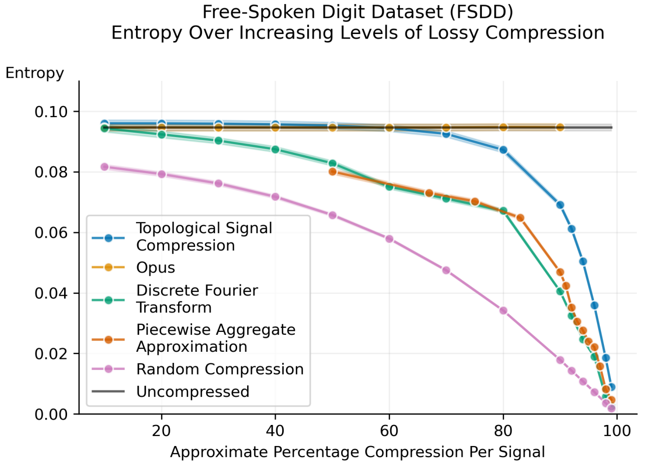

Figure 8. Entropy calculations on increasingly-compressed Free-Spoken Digit Dataset (FSDD) signals. Error bars represent 2 times the standard error of entropy over all digits in the dataset. TSC maintains entropy levels comparable to Opus at up to 70% compression and similarly to Figure 5 outperforms other compression methodologies above 90% compression.

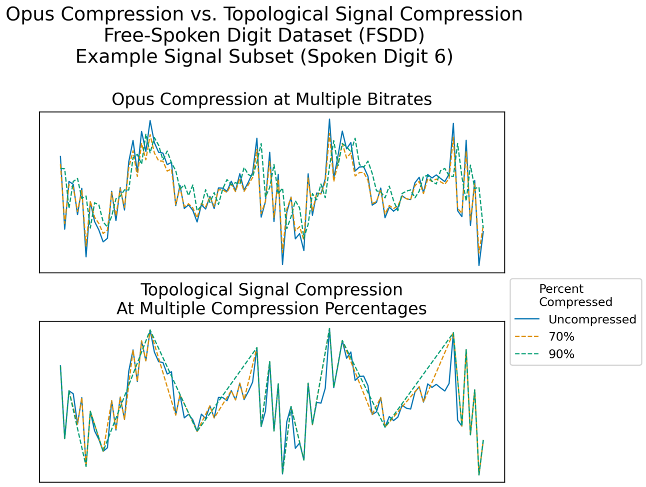

Figure 9. Comparing a reconstruction of the FSDD signal used in Figures 3 and 12, zoomed in on a 1000 point subset of the signal. Note that at the 90% compression level, TSC preserves the location of critical points while Opus does not.

if we choose to remove only a single additional persistence pair from the reconstruction, we will "un-kink" a small subset of the reconstruction, leaving the remaining reconstruction unaltered. This results in increased interpretability *between* reconstructions at variable compression levels, which supports the ability to use TSC in an environment with a variable communications budget, discussed in more detail later in this section.

Handling of Noise

The extent to which the compression algorithms perform well with respect to noise deserves special consideration since many real-world applications of compression occur in noisy environments. To explore this in the context of our machine learning exercises, we will reference Figure 10, which is simply the data from Figure 7 but separated instead by compression methodology to allow us to compare each method's performance against itself as noise increases.

DFT seems to be both theoretically and practically the least robust to noise. Theoretically, as noise increases, the risk of overfitting a Fourier coefficient to local noise within

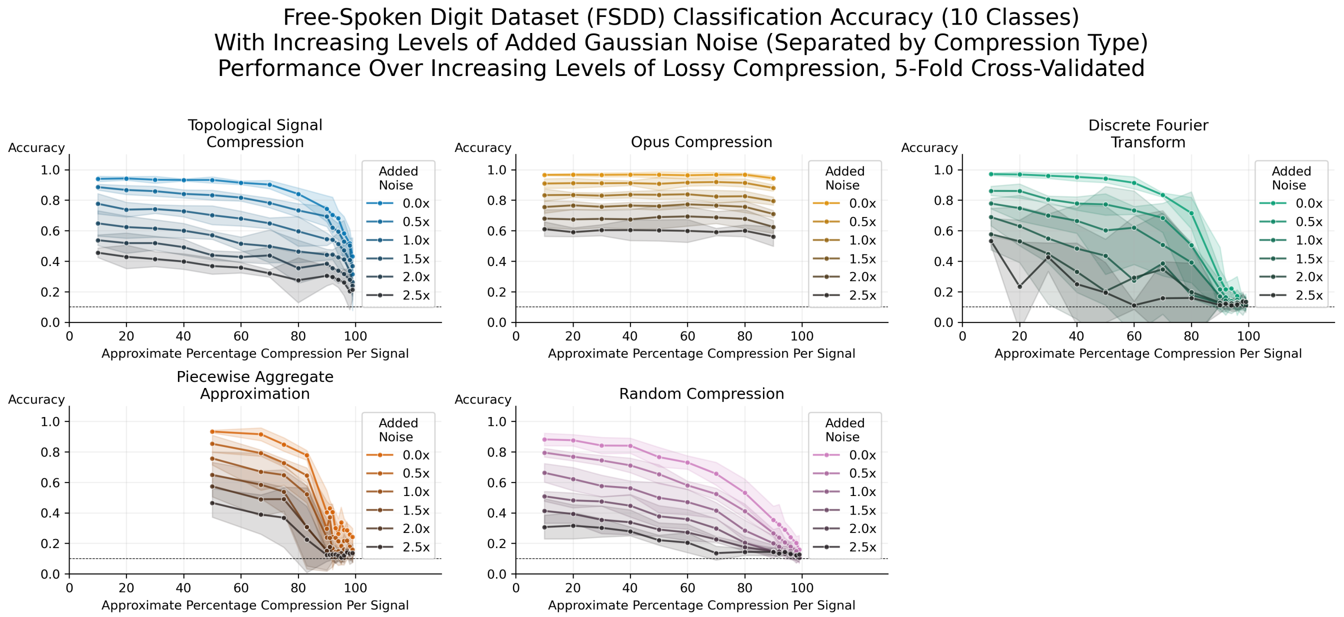

Figure 10. Accuracy of various compression methodologies over increasingly noisy FSDD dataset. Gaussian noise as high as 2.5 times the noise in the standardized signals was added to the dataset. Though machine learning accuracy declines as noise increases for all compression methodologies, TSC and Opus produce more consistent results, as exemplified by the smaller error bars and slower, smoother decline in machine learning accuracy at higher compression levels, even at high levels of noise. Note, we are visualizing the same data as in Figure 7, instead separated by compression method as opposed to the amount of noise. Error bars represent 2 times the standard deviation of accuracy over the 5-fold cross-validated results at each compression level.

a signal and thus disrupting the signal globally increases. Furthermore, with potentially more poorly-formed Fourier coefficients, DFT should pay an additional penalty as one increases compression by keeping a smaller number of Fourier coefficients. In practice, this appears to hold true in Figure 10, with the error bars in machine learning accuracy spanning more than 40 percentage points at some compression levels as noise increases. We should note, however, an obvious means of improving DFT would be to run a "windowed" DFT, where to keep n points, one would partition a signal into m windows and keep m/n points per window. Of course, windowing would also be a natural extension for TSC and even Random Compression as well, but we chose to only explore running these compression algorithms "globally" in this paper.

PAA should be theoretically robust to noise as long as the noise is unbiased. In particular, as window size increases, the chance of the noise in a window averaging out to 0 increases. In practice, looking at Figure 10, this seems to hold, with the larger error bars at higher noise levels appearing to narrow as compression increases. Varying stability costs with respect to window size as noise increases is a clear negative to using PAA; however, it should be noted that PAA can likely play a strong role in situations with unbiased noise and very high sampling rates (where large windows can average out the noise without excessively compressing the actual information content in the signal).

The interpretability of noise on Opus compressions is somewhat uncertain, mainly due to the same concern raised by Figure 9, but the machine learning results were highly stable to noise over increased compression levels in practice, as demonstrated by the relatively small error bars for Opus in Figure 10.

TSC has a convenient interpretability when it comes to noise. Although the returned critical points may be shifted by noise, as soon as our persistence cutoff exceeds our noise level, the resulting reconstructed signals will be otherwise unaffected by the noise. This offers a generalizable, interpretable means by which one could insulate a model from minor amounts of noise. For example, if one had noiseless training data and were worried about the external validity of the resulting trained model to signals with small amounts of noise, then one could instead train and use the model in practice with TSC-simplified, noise-reduced signals, that is, signals with low-persistence critical points removed. As for TSC's machine learning results in practice, although showing a greater decline in classification accuracy when compared to Opus, Figure 10 shows a stability in ML classification that is visually comparable to Opus, as exemplified by the relatively small error bars and smooth decay in accuracy for TSC both as noise increases and along the span of compression levels for a given amount of noise.

Variable Communications Budget

In order to make the most of a tight communications budget, a compression algorithm must have the flexibility to generate outputs of a precise size. For any given tight communications budget, there exists a *nearly exact* compression level at which TSC could send a topologically simplified signal due to TSC's ability to throw out individual points from a given compression on the margin (e.g. sending one less critical point from the persistence diagram) as opposed to making global changes to the signal being compressed, as is done when changing compression levels using Opus and PAA (a marginal change to the bitrate or window size, respectively, will change the size of the compression more dramatically).

If the tight communications budget were *variable*, then the marginal compression capability of TSC at the level of *bytes* would allow one to efficiently utilize the entire communications budget by reconstructing the signal using the exact number of points allowed under the variable budget at any given moment in time.

In our earlier machine learning exercises, in particular Figures 5 and 6, we trained a machine learning model for each specific level of compression. If one were sending compressed data over a variable communications budget, however, then one would also need to be ready to *learn from variably-compressed data*. This would require a model (or set of models) that can be trusted over a range of compression levels.

Setting the valid compression range of a model poses a challenge. If the ranges get too wide, machine learning accuracy would likely suffer, but if the ranges are too narrow, one risks spreading the data too thin to reliably train models to span all ranges. Problems in either direction would reduce the trustworthiness of any modeled insights. The implied optimization scheme here is thus to maintain the largest possible compression ranges as long as the resulting signals are sufficiently comparable to deserve being classified by the same model. Therefore, the more smoothly the signals change as they are compressed, the greater the range of compression levels one would expect to be able to reliably utilize for a given trained model.

In addressing this concern, we will focus only on TSC and DFT, as they are the only two informed algorithms considered in this paper that can easily achieve a byte-specific variable communications budget.

On a theoretical level, we should already expect TSC to excel at this task relative to DFT, based on the earlier discussion of TSC's "marginal" compression effects on a signal as opposed to DFT's "global" compression effects. On an anecdotal level, note in Figures 3 and 12, the "shape" of the compressed signal is relatively consistent for TSC but changes drastically for DFT at higher levels of compression.

To empirically explore this, we looked at the Dynamic Time Warping distance between original and compressed signals over increasing amounts of both compression and Gaussian noise, shown in Figure 11. Despite DFT achieving greater similarity and lower variance at low levels of compression, TSC outperforms DFT on both fronts at higher levels of compression while additionally maintaining a more stable change both over increased compression and noise. Thus, the relative "smoothness" of Dynamic Time Warping distances for TSC over DFT is indicative of a greater ability for a TSC-trained model to generalize to a larger range of compression levels.

6. Conclusion

We find Topological Signal Compression offers the most promising ability to create actionable information under a variable communications budget. More generally, TSC's interpretability combined with its ability to fine-tune its compression to arbitrarily high byte constraints, locally compress on the margin, smoothly change its increasingly compressed signals, handle noise, and generalize to any signal data make it worthy of consideration in any constrained communications scenario.

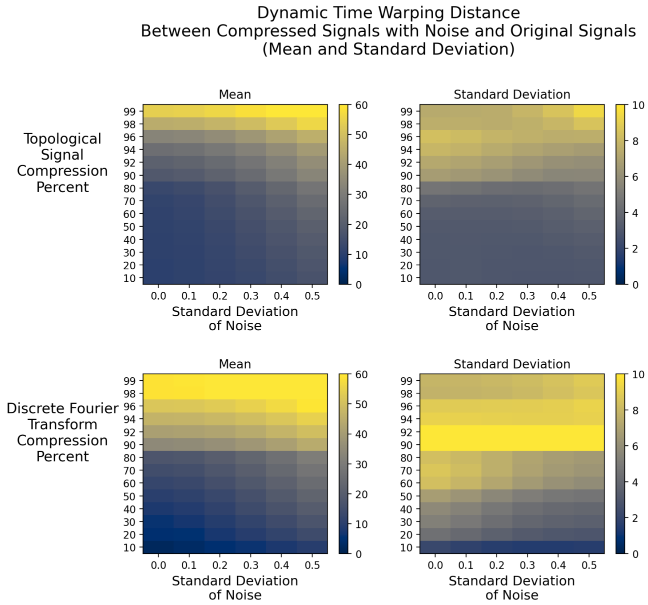

Figure 11. Dynamic Time Warping (DTW) distance between original signals and reconstructions of the FSDD signals compressed using Topological Signal Compression (TSC) and Discrete Fourier Transform (DFT). Although DFT achieves closer DTW distance and lower variance of distances at lower levels of compression, TSC outperforms DFT on both fronts at higher levels of compression. TSC also exhibits a smoother increasing of DTW distances than DFT.

Future Work

Finally, we comment on an upcoming research direction that will show additional promise of TSC within signal-processing and machine-learning pipelines. The experiments above operate under the abstraction where multiple edge devices each sense a one-dimensional time series, compress them using TSC, send the compressed versions to a common center, and then the center reconstructs them before applying a chosen classification technique. Somewhat implicit in this abstraction is the assumption that the time series come from a common modality, or at least that it makes sense to apply a common classification technique to them. However, we are also exploring applications of TSC as part of an upstream fusion (e.g., [23]) pipeline.

The idea is quite simple. We imagine that the edge devices sense one-dimensional time series arising from potentially many different modalities. After applying TSC and transmitting and reconstructing at a common center, we will then apply any number of fusion techniques that operate directly on time series; for example, similarity network fusion ([24], [25]) or different variations of joint manifold learning ([26], [27], [28]). Subsequently, any time series classification technique can be applied.

Appendix

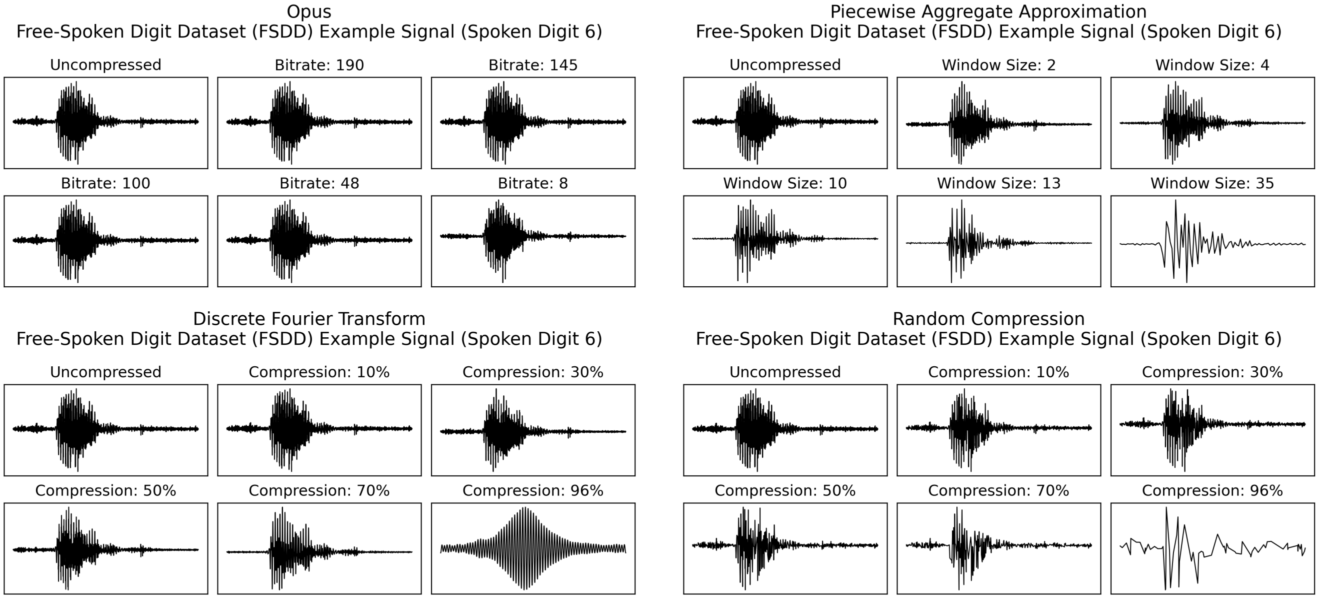

Figure 12. Counterfactual compression methodologies run on the same FSDD signal as in Figure 3. Note: the bitrate compressions for Opus and window size compressions for Piecewise Aggregate Approximation do not correspond to the same compression percentages shown for Topological Signal Compression, Random Compression, and the Discrete Fourier Transform.

Acknowledgments

Research on this project was partially supported by the DARPA Ocean of Things project, under contract N6600121C4006. This document has been Approved for Public Release, with Distribution Statement A, Distribution Unlimited. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

[1] W. Chaovalitwongse, O. Prokopyev, and P. Pardalos, “Electroencephalogram (eeg) time series classification: Applications in epilepsy,” *Annals of Operations Research*, vol. 148, pp. 227–250, 2006.

[2] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, “The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances,” *Data Min Knowl Disc*, vol. 31, pp. 606–660, 2017.

[3] H. I. Fawaz, G. Forester, J. Weber, L. Idoumghar, and P.-A. Muller, “Deep learning for time series for time series classification: a review,” *Data Min Knowl Disc*, vol. 31, pp. 917–963, 2019.

[4] J. Flaks, Z. Jackson, H. Nicolas, Y. Pan, C. Souza, and A. Thite, *Free Spoken Digit Dataset*, 2017. `https://zenodo.org/badge/latestdoi/61622039`.

[5] J. Waterston, J. Rhea, S. Peterson, L. Bolick, J. Ayers, and J. Ellen, “Ocean of things: Affordable maritime sensors with scalable analysis,” in *OCEANS 2019-Marseille*, pp. 1–6, IEEE, 2019.

[6] I. S. LLC, “Iridium sbd: Short burst data service.” Brochure, 2013.

[7] H. Edelsbrunner, D. Letscher, and A. Zomorodian, “Topological persistence and simplification,” in *Proceedings 41st annual symposium on foundations of computer science*, pp. 454–463, IEEE, 2000.

[8] F. Chazal, D. Cohen-Steiner, M. Glisse, L. J. Guibas, and S. Y. Oudot, “Proximity of persistence modules and their diagrams,” in *Proceedings of the 25th annual symposium on Computational geometry*, SCG ’09, (New York, NY, USA), pp. 237–246, ACM, 2009.

[9] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, “Stability of persistence diagrams,” *Discrete Comput. Geom.*, vol. 37, pp. 103–120, Jan. 2007.

[10] P. Bendich, J. S. Marron, E. Miller, A. Pieloch, and S. Skwerer, “Persistent homology analysis of brain artery trees,” *Annals of Applied Statistics*, vol. 10, no. 1, pp. 198–218, 2016.

[11] J. W. Milnor, *Morse Theory*. Princeton University Press, 1963.

[12] F. Laudenbach, “A proof of morse’s theorem about the cancellation of critical points,” *Comptes Rendus de l’Academie des Sciences*, pp. 483–488, 2013.

[13] U. Bauer, C.-B. Schönlieb, and M. Wardetzky, “Total variation meets topological persistence: A first encounter,” in *AIP Conference Proceedings*, vol. 1281, pp. 1022–1026, American Institute of Physics, 2010.

[14] H. Edelsbrunner, D. Morozov, and V. Pascucci, “Persistence-sensitive simplification functions on 2-manifolds,” in *Proceedings of the twenty-second annual symposium on Computational geometry*, pp. 127–134, 2006.

[15] A. Poulenard, P. Skraba, and M. Ovsjanikov, “Topological function optimization for continuous shape matching,” *Computer Graphics Forum*, vol. 37, no. 5, pp. 13–25, 2018.

[16] B. P. Bogert, “The quefrency alanysis of time series

for echoes; cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking," *Time series analysis*, pp. 209–243, 1963.

[17] B. Logan *et al.*, "Mel frequency cepstral coefficients for music modeling.," in *Ismir*, vol. 270, pp. 1–11, Citeseer, 2000.

[18] T. B. T. Jean-Marc Valin, Koen Vos, "Opus interactive audio codec." `https://opus-codec.org`.

[19] P. Schäfer and M. Högqvist, "Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets," in *Proceedings of the 15th international conference on extending database technology*, pp. 516–527, 2012.

[20] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[21] S. M. Pincus, "Approximate entropy as a measure of system complexity.," *Proceedings of the National Academy of Sciences*, vol. 88, no. 6, pp. 2297–2301, 1991.

[22] A. Delgado-Bonal and A. Marshak, "Approximate entropy and sample entropy: A comprehensive tutorial," *Entropy*, vol. 21, May 2019.

[23] A. Newman and G. Mitzel, "Upstream data fusion: History, technical overview, and applications to critical challenges," *APL Technical Digest*, 2013.

[24] B. Wang, A. M. Mezlini, F. Demir, M. Fiume, Z. Tu, M. Brudno, B. Haibe-Kains, and A. Goldenberg, "Similarity network fusion for aggregating data types on a genomic scale," *Nature methods*, vol. 11, no. 3, p. 333, 2014.

[25] C. J. Tralie, P. Bendich, and J. Harer, "Multi-scale geometric summaries for heterogeneous sensor fusion," in *Proc. 2019 IEEE Aerospace Conference*.

[26] M. A. Davenport, C. Hegde, M. F. Duarte, and R. G. Baraniuk, "High dimensional data fusion via joint manifold learning," in *AAAI Fall Symposium: Manifold Learning and Its Applications*, 2010.

[27] D. Shen, E. Blasch, P. Zulch, M. Distasio, R. Niu, J. Lu, Z. Wang, and G. Chen, "A joint manifold leaning-based framework for heterogeneous upstream data fusion," *Journal of Algorithms & Computational Technology*, vol. 12, no. 4, pp. 311–332, 2018.

[28] E. Solomon and P. Bendich, "Geometric fusion via joint delay embeddings," in *Proceedings of the 23rd Int. Conf. on Inf. Fusion*, 2020.

BIOGRAPHY

Gary Koplik *is a Senior Data Scientist and Data Visualization Engineer at GDA. He received a B.A. in Economics and Mathematical Science from Colby College and a M.S. in Economics and Computer Science from Duke University. His research has included topics such as historical market responses to unemployment reports, summarizing variable interactions in large databases, and the incidence of rare diseases in health systems. At GDA, he focuses on the geometry and coverage of non-stationary sensor networks, as well as on building company-wide static and dynamic visualization skills.*

Nathan Borggren *was a Senior Physicist while at GDA, where he led the blockchain and IoT efforts. His scientific journey has led him to the moons of Saturn aboard the Cassini spacecraft and to the nuclear furnace of particle collisions at the Relativistic Heavy Ion Collider. He received his Ph.D. from Stony Brook University in New York, completing a thesis on stochasticity in a genetic switch. Nathan is intrigued by noise wherever he can find it — from genetic networks to financial markets to superconducting circuits. Currently, he teaches Discrete Mathematics and Abstract Algebra for the Bard Prison Initiative.*

Sam Voisin *was a Data Scientist while at GDA. He received his B.S. in Financial Management from Clemson University and his M.S. in Statistical Science from Duke University. While at Duke, he researched methods for pre-processing sEMG signals as a means to classify physical gestures. Currently, he is a Data Scientist at Infinia ML.*

Gabrielle Angeloro *is a Data Scientist at GDA. She received a B.S. in Mathematics from SUNY Geneseo and a M.S. in Mathematics from Iowa State University. While at Iowa State, she developed a Python package implementing persistence landscapes: a vectorization scheme for persistent homology. Gabrielle's current research interests are in the intersection of topological tools and deep learning.*

Jay Hineman *is Chief Solutions Architect at GDA. He received his Ph.D. in Mathematics from the University of Kentucky in 2012. He has worked as a researcher and instructor at the University of Kentucky and Fordham University. Dr. Hineman has extensive knowledge of numerical simulation and analysis of liquid crystals, ion electrochemistry, and biomembranes; and holds a graduate certificate in computational fluid dynamics from the University of Kentucky. Many of these topics have rich geometric interpretations (e.g., harmonic maps and curvature flow) applicable to broader questions about data. In addition, he is experienced in configuring OS and hardware to build and run large scale scientific code. At GDA, Dr. Hineman has applied his mathematical and computational background to integrating topological data analysis tools with machine learning techniques. He has focused on the domains of data fusion for targeting and control of system of systems for agile logistics and military medicine. He also serves as an adjunct instructor in the ECE Department at Duke University, where he leads classes about the implementation of machine learning and reinforcement learning at scale.*

Tessa Johnson *was a Data Scientist while at GDA. She received a B.S. in Applied Mathematics and Statistics from Texas A&M University and a M.S. in Statistical Sciences from Duke University. She has broad research interests including applications of statistical modeling for forensics data and development of improved feature selection methodology for complex feature sets. During her graduate studies, she worked on applying Bayesian methodology to Bioinformatics data and exploring the evolution of Dynamic Social Networks in a Bayesian framework. At GDA, her research focuses on the implementation and application of novel topological algorithms and fusion techniques for high-dimensional data arising from multiple sensing modalities. Currently, she is a Data Scientist at Fidelity Investments.*

Paul Bendich *is Chief Scientist at GDA. He is an Associate Research Professor of Mathematics at Duke University and the Associate Director for Undergraduate Research in the Information Initiative at Duke. He received his Ph.D. in Mathematics from Duke in 2008 and held postdoctoral positions at the Institute for Science and Technology Austria and Penn State. Dr. Bendich's doctoral work laid some of the early theoretical foundations for topological data analysis (TDA). Since then, he has been at the forefront of the integration of TDA with more standard machine learning and statistical techniques. This work has found wide application in vehicle tracking, brain imaging, and image simplification, among many other areas. Dr. Bendich oversees all scientific efforts at GDA. Through his affiliation with the Rhodes Information Initiative at Duke, Dr. Bendich has developed broad and deep expertise across the field of modern data analysis, and he has frequently been the leader of interdisciplinary and vertically integrated teams.*