Shift-Invariant Dictionary Learning using TCN-WTA Autoencoders for Discovering Musical Relations

Anonymous ACL submission

Abstract

Music temporal structure is full of shift-invariant patterns (e.g. motifs, ostinatos, loops, samples, etc.). We propose using a using a Fully Convolutional Temporal Autoencoder (FCTA) with a k-Winner Takes All (k-WTA) activation function to find a shift-invariant dictionary to represent symbolic, multivariate, musical signals. We train our model to represent fixed drum tracks and variable length piano music. We show few applications of this sparse representation such as de-noising musical ideas, unsupervised stylistic segmentation, and music generation. To assist related work, we have made interactive code available along with the trained models

1 Introduction

The dictionary learning framework, aims at finding a sparse representation of the input data (sparse coding) in the form of a linear combination of basic elements we call atoms. There are multiple benefits in obtaining a sparse encoding of a signal. To name a few, they are lightweight—requiring less memory to store and allowing faster inference and easier interpretability and they can encode prior knowledge in the sparsity patterns. In addition, sparsity provides a way to discern patterns in an informed and principled manner .

Sparse dictionary learning has led to state-of-art results in various image and video processing tasks as well as to texture synthesis (Peyré, 2009) and unsupervised clustering (Ramírez et al., 2010) . In evaluations with the Bag-of-Words model (Koniusz et al., 2017), sparse coding was found empirically to outperform other coding approaches on the object category recognition tasks.

The ability to dis til complex data structures such as music down to sets of dictionaries—salient features of a specific performer or music, has a multitude of applictions in music. Music transcription and classification tasks have seen a strong usage of sparse dictionary learning in the past (Grosse et al., 2007) (Costantini et al., 2013), (Blumensath and Davies, 2006), (Srinivas M et al., 2014), (Srinivas et al., 2014), (Cogliati et al., 2016). Nonetheless, we have yet to see a study that harnesses the advantages of sparse representation for the purpose of music creation. Instead, the popular methods for discovering music relations and achieving music generation have been a transformer with some sort of attention mechanism or the recurrent architectures. (Jiang Junyan et al., 2020) for instance, uses an attention module that is tailored to the discovery of sequence level relations in music, while studies like (Roberts et al., 2018) uses the recurrent variational autoencoder and a hierarchical decoder in order to model long-term musical structures. In our study we explore applications of sparse representation such as de-noising musical ideas, unsupervised stylistic segmentation, and music generation.

2 Related Work

2.1 Dictionary learning

Given the data: $X = [x_1, \ldots, x_K], x_i \in \mathbb{R}^d$ We want a dictionary $\mathbf{D} \in \mathbb{R}^{d \times n} : D = [d_1, \ldots, d_n]$ And a representation $R = [r_1, \ldots, r_K], r_i \in \mathbb{R}^n$ such that the reconstruction $\|X - \mathbf{D}R\|_F^2$ is minimized and r_i are sparsed. The optimization problem can be formulated as:

$$\underset{\mathbf{D} \in \mathcal{C}, r_i \in \mathbb{R}^n, \lambda > 0}{\operatorname{argmin}} \sum_{i=1}^{K} \|x_i - \mathbf{D}r_i\|_2^2 + \lambda \|r_i\|_0$$

$$\mathcal{C} \equiv \left\{ \mathbf{D} \in \mathbb{R}^{d \times n} : \|d_i\|_2 \leq 1 \forall i = 1, \ldots, n \right\}$$

There are various methods to solve this problem, however this formulation does not look for shift invariant features. The dictionary components are the same size as original signal we are seeking to recustruct.

2.2 Shift-invariant dictionary learning (SIDL)

Shift-invariant dictionary learning (SIDL) refers to the problem of discovering a latent basis that captures local patterns at different locations of input signal, and a sparse coding for each sequence as a linear combination of these elements (Zheng et al., 2016)

This has a similar formulation as eq.1 except to reconstruct we have to stride along our signal. We can rewrite
$\mathbf{D}r_i \longrightarrow \sum_{k=1}^{K} \boldsymbol{r}_{ik} T\left(\mathbf{d}_k, t_{ik}\right)$
where

$$T_p(\mathbf{d}, t) = \begin{cases} \mathbf{d}_{i-t} & \text{if } 1 \leq i-t \leq q \\ 0 & \text{otherwise} \end{cases}$$

here t_{ik} corresponds first location where d_k matches our signal. Therefore, $t_{ik} = 0$ indicating that $\mathbf{d}_k$ is aligned to the beginning of $\mathbf{x}_i$ and that $t_{ik} = p - q$ indicating the largest shift $\mathbf{d}_k$ can be aligned to $\mathbf{x}_i$ without running beyond

In previous works, various shift-invariant dictionary learning (SIDL) methods have been employed to discover local patterns that are embedded across a longer time series in sequential data such as audio signals. While (Grosse et al., 2007) employs shift- invariant sparse coding with a convolutional optimization and gradient descent method for an audio classification task, (Zheng et al., 2016) demonstrates an efficient algorithm with the ability to combine shift-invariant patterns in a sparse coding of the original data for audio reconstruction and classification tasks.

2.3 Temporal Convolutional Networks (TCN)

Recent results suggest that TCNs convincingly outperform baseline recurrent architectures across a broad range of sequence modeling tasks.

The distinguishing characteristics of TCNs are: 1) the convolutions in the architecture are causal, meaning that there is no information "leakage" from future to past; 2) the architecture can take a sequence of any length and map it to an output sequence of the same length, just as with an RNN.

To put it simply: TCN = 1D FCN + causal convolutions.

This architecture is informed by recent convolutional architectures for sequential data (van den Oord et al., 2016), (Kalchbrenner et al., 2017) ; (Dauphin et al., 2016); (Zheng et al., 2016), TCN have several advantages, for example, the TCN is much simpler than WaveNet (van den Oord et al., 2016) (no skip connections across layers, conditioning, context stacking, or gated activations). Compared to the language modeling architecture of Dauphin et al. (2017), TCNs do not use gating mechanisms and have much longer memory.

2.4 SIDL by TCN K-WTA Autoencoders

To learn the sparse dictionaries, we use autoencoders and more specifically the Winner-Take-All Autoencoders [11]. WTA autoencoders are similar to other autoencoders except that the goal is to learn sparse representations rather than dense ones. To achieve this goal, after training the encoder the single largest hidden activity of each feature map is kept and the rest (as well as their derivatives) are set to zero. Next, the decoder reconstructs the output from the sparse feature maps. This results in a sparse representation where the sparsity level is the number of non-zero feature maps. The key principle: *If a shallow decoder (1 layer) is used, the kernel weights of the decoder are the atoms of the dictionary.*

In theory, all we need to learn a dictionary is a Ecoder-Decoder 1D-Conv layers to generate a sprase representation. However, if our signal is dense, learning can improve if we add a TCN Block Layer to extract temporal features. This TNC block can have variable depth depending on the task.

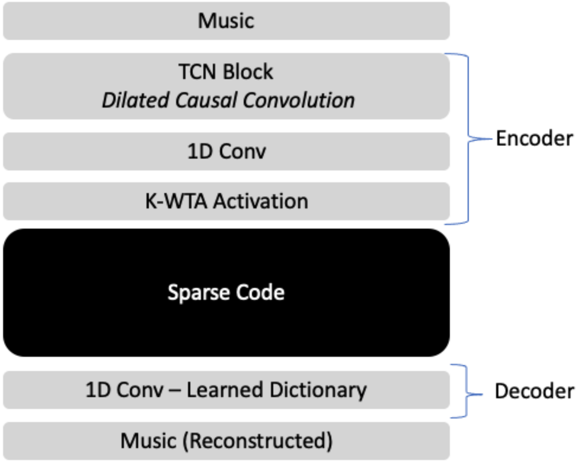

Figure 1: After training the model we can use it to encode datapoints of arbitrary length unsupervised stylistic segmentation. We use PCA on the average sparse code for each piece. We project into 2 dimensional sparse to visualize

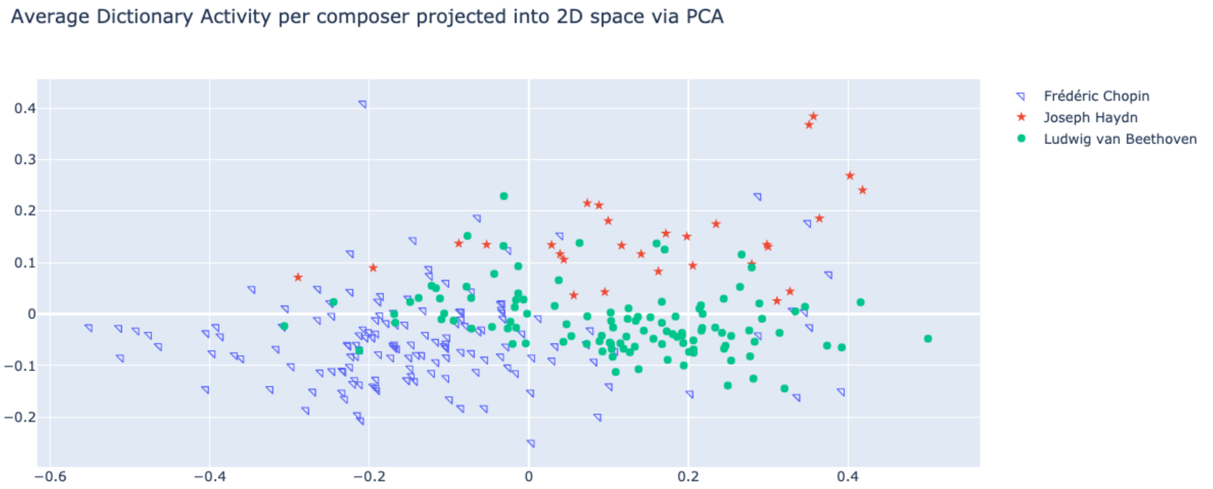

Figure 2: After training the model we can use it to encode datapoints of arbitrary length unsupervised stylistic segmentation. We use PCA on the average sparse code for each piece. We project into 2 dimensional sparse to visualize

3 Experiments

We show a few applications of our TCN k-WTA model, De-noising musical ideas; Unsupervised stylistic segmentation; Music generation. We do this for two distinct datasets with different MIDI encodings.

3.1 Datasets

We use two distinct datasets: MAESTRO (Hawthorne et al., 2019), and grove Groove (Gillick et al., 2019) . See table 2 for more details on the datasets used. We also use distinct MIDI representations for each dataset.

3.2 Model Implementation

Our model implementation differs slightly for the different datasets used. Below are the implementation details

Maestro Model : We use an architecture as in figure 1. The TCN layer uses a [1,8,16,24] feature map, and we use 1000 dictionary size. We use a decaying k-WTA since this empirically improved our model accuracy. We begin with $k = 95$ and decay to $k = 75$ over 60 Epochs. The Convolutions are non-overlapping strides–meaning every kernel-length time step is only made up one column in our sparse code–this helps reduce memory requirements and help us find repeating sections over a fixed kernel length. We implemented the model in Pytorch, with MSE loss on recustruction, AdamW optimizer, and use a batch size of 1. This batch size allows us to train on variable length music since our autoencoder is fully convolutional–however our sparse representation will also be variable length.

Groove Model : For the Groove, drum dataset our design is We use an architecture as in figure 1 but without TCN layer, and we use 100 dictionary size. We use a decaying k-WTA since this empirically improved our model accuracy. We begin with $k = 95$ and decay to $k = 75$ over 60 The Convolutions are similarly also non-overlapping strides. We implemented the model in Pytorch, with MSE loss on reconstruction, AdamW optimizer. We do not batch, we train on the full dataset. We preprocess the dataset to match 120 bpm, 4/4 time signature and only train on 1 bar of music. For this dataset we have good understanding of repeating time intervals and structure, unlike maestro dataset.

3.3 Music Reconstruction

De-noising musical ideas: Dictionary learning has successfully been shown to de-noise image, videos. By simply reconstructing our musical signal.

Keep N Active Atoms: Another application of having a sprase code, is the ability to recognize the most used words in the dictionary. Given a sparse code we can limit a piece to only be made up of the top N atoms. Some example applications of keeping top N atoms are for example, low dimensionality feature extraction for machine learning task; music reduction–reduction wherein the complexity of the arrangement is reduced to a simpler transcription and parts.

Dataset	Size	Instrument	MIDI Representation
MAESTRO	1020 (Hrs)	Piano	One-hot encoding over 388 different MIDI events. Every datapoint here has an arbitrary length
Groove	3.6 (Hrs)	Drum	T timesteps (one per 16th note) and 27 MIDI events. We use fixed length 64 time step sections

Table 1: Datasets used to experiment with fully convolutional temporal autoencoder model. All datasets used are MIDI format

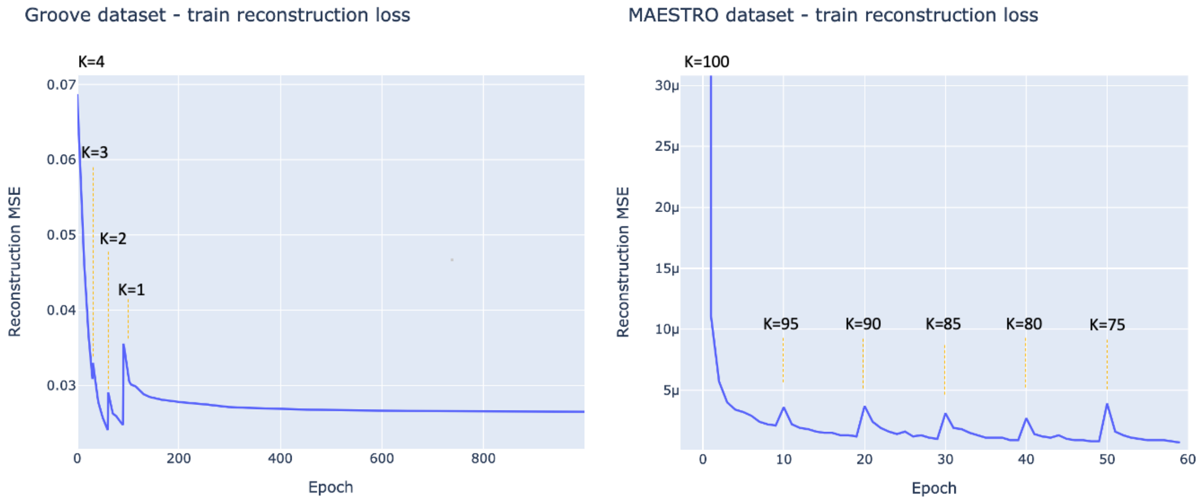

Figure 3: After training the model we can use it to encode datapoints of arbitrary length unsupervised stylistic segmentation. We use PCA on the average sparse code for each piece. We project into 2 dimensional sparse to visualize

3.4 Unsupervised Stylistic Segmentation

If we train with a dataset that includes multiple composers we should expect to find that different composers utilize different shift invariant patters. To visualize kernel usage we average out the rows of our sparse code. This should provide us with the average dictionary word value. If we do PCA on the average dictionary vector and reduce dimensionality to 2D. We obtain the plot in Figure 2

It is also possible to use this average kernel usage for each composer and make comparisons for composers, such as most similar or dissimilar between styles or comparisons.

3.5 Generating New Music

Interpolating Sparse Code We use encode existing music and interpolate between sparse code to generate new music. In order to obtain good results we measure cos similarity between sparse code to find candidates to interpolate. For drum generation we can all of our sections are same length; for piano we have variable length sprase code. To interpolate variable length code we can fix the input size, or interpolate specific sections of the sprase code. The results are significantly better for drum generation than piano generation. Although the piano generation does seem to follow high level musical features, such as key time, and scale.

Swapping atoms Another method to generate new music, is to measure the most active atoms in the dictionary of a musical section, and replace it with the most active atom in the dictionary of a distinct section. This also can be used as a tool for artist to mix and match features of ordered importance.

4 Conclusion

We have shown that TCN-kWTA autoencoder can learn a sparse representation of abitrary length musical signal. This shift invariant, sparse representation can be used to analyze, preprocess, or generate musical content in a structured, or unsctuctured way

The performance on the recustruction and generation on for the drum Grove dataset was significantly better than the piano (MAESTRO dataset). This is in part because the dataset was preproccessed to match with kernel size, and the drum sections were the same length and have lower dimensionality.

Acknowledgments

The acknowledgments should go immediately before the references. Do not number the acknowledgments section. Do not include this section when submitting your paper for review.

Preparing References:
Include your own bib file like this:
```
\bibliographystyle{nlp4MusA_natbib}
\bibliography{nlp4MusA}
```
where `nlp4MusA` corresponds to a nlp4MusA.bib file.

References

Thomas Blumensath and Mike Davies. 2006. Sparse and shift-invariant representations of music. In *IEEE Transactions on Audio, Speech and Language Processing*, volume 14.

Andrea Cogliati, Zhiyao Duan, and Brendt Wohlberg. 2016. Context-Dependent Piano Music Transcription with Convolutional Sparse Coding. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 24(12).

Giovanni Costantini, Massimiliano Todisco, and Renzo Perfetti. 2013. NMF based dictionary learning for automatic transcription of polyphonic piano music. *WSEAS Transactions on Signal Processing*, 9(3).

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083.

Jon Gillick, Adam Roberts, Jesse Engel, Douglas Eck, and David Bamman. 2019. Learning to groove with inverse sequence transformations. In *International Conference on Machine Learning (ICML)*.

Roger Grosse, Rajat Raina, Helen Kwong, and Andrew Y. Ng. 2007. Shift-invariant sparse coding for audio classification. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence, UAI 2007*.

Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. 2019. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*.

Jiang Junyan, Gus Xia, and Taylor Berg-Kirkpatrick. 2020. Discovering Music Relations with Sequential Attention. In *Proceedings of the 1st workshop on nlp for music and audio (nlp4musa)*, pages 1–5.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2017. Neural machine translation in linear time.

Piotr Koniusz, Fei Yan, Philippe-Henri Gosselin, and Krystian Mikolajczyk. 2017. Higher-order occurrence pooling for bags-of-words: Visual concept detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(2):313–326.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio.

Gabriel Peyré. 2009. Sparse modeling of textures. 34(1):17–31.

I. Ramírez, P. Sprechmann, and G. Sapiro. 2010. Classification and clustering via dictionary learning with structured incoherence and shared features. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3501–3508.

Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. 2018. A hierarchical latent vector model for learning long-term structure in music. In *35th International Conference on Machine Learning, ICML 2018*, volume 10.

M. Srinivas, Debaditya Roy, and C. Krishna Mohan. 2014. Learning sparse dictionaries for music and speech classification. In *International Conference on Digital Signal Processing, DSP*, volume 2014-January.

Srinivas M, Roy D, and Mohan CK. 2014. Music genre classification using on-line dictionary learning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1937–1941.

Guoqing Zheng, Yiming Yang, and Jaime Carbonell. 2016. Efficient shift-invariant dictionary learning. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August-2016.

A Appendices

Appendices are material that can be read, and include lemmas, formulas, proofs, and tables that are not critical to the reading and understanding of the paper. Appendices should be **uploaded as supplementary material** when submitting the paper for review. Upon acceptance, the appendices come after the references, as shown here. Use \appendix before any appendix section to switch the section numbering over to letters.

B Supplemental Material

Submissions may include non-readable supplementary material used in the work and described in the paper. Any accompanying software and/or data should include licenses and documentation of research review as appropriate. Supplementary material may report preprocessing decisions, model parameters, and other details necessary for the replication of the experiments reported in the paper. Seemingly small preprocessing decisions can sometimes make a large difference in performance, so it is crucial to record such decisions to precisely characterize state-of-the-art methods.