SHUFFLE TO LEARN: SELF-SUPERVISED LEARNING FROM PERMUTATIONS VIA DIFFERENTIABLE RANKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-supervised pre-training using so-called "pretext" tasks has recently shown impressive performance across a wide range of tasks. In this work, we advance self-supervised learning from permutations, that consists in shuffling parts of input and training a model to reorder them, improving downstream performance in classification. To do so, we overcome the main challenges of integrating permutation inversions (a discontinuous operation) into an end-to-end training scheme, heretofore sidestepped by casting the reordering task as classification, fundamentally reducing the space of permutations that can be exploited. We make two main contributions. First, we use recent advances in differentiable ranking to integrate the permutation inversion flawlessly into a neural network, enabling us to use the full set of permutations, at no additional computing cost. Our experiments validate that learning from all possible permutations improves the quality of the pre-trained representations over using a limited, fixed set. Second, we successfully demonstrate that inverting permutations is a meaningful pretext task in a diverse range of modalities, beyond images, which does not require modality-specific design. In particular, we improve music understanding by reordering spectrogram patches in the time-frequency space, as well as video classification by reordering frames along the time axis. We furthermore analyze the influence of the patches that we use (vertical, horizontal, 2-dimensional), as well as the benefit of our approach in different data regimes.

1 INTRODUCTION

Supervised learning has achieved important successes on large annotated datasets (Deng et al., 2009; Amodei et al., 2016). However, most available data, whether images, audio, or videos are unlabelled. For this reason, pre-training representations in an unsupervised way, with subsequent fine-tuning on labelled data, has become the standard to extend the performance of deep architectures to applications where annotations are scarce, such as understanding medical images (Rajpurkar et al., 2017), recognizing speech from under-resourced languages (Rivière et al., 2020; Conneau et al., 2020), or solving specific language inference tasks (Devlin et al., 2018). Among unsupervised training schemes, self-supervised learning focuses on designing a proxy training objective, that requires no annotation, such that the representations incidentally learned will generalize well to the task of interest, limiting the amount of labeled data needed for fine-tuning. Such "pretext" tasks, a term coined by Doersch et al. (2015), include learning to colorize an artificially gray-scaled image (Larsson et al., 2017), inpainting removed patches (Pathak et al., 2016) or recognizing with which angle an original image was rotated (Gidaris et al., 2018). Other approaches for self-supervision include classification to original images after data augmentation (Chen et al., 2020) and clustering (Caron et al., 2018).

In this work, we consider the pretext task of reordering patches of an input, first proposed for images by Noroozi & Favaro (2016), the analogue of solving a jigsaw puzzle. In this setting, we first split an input into patches and shuffle them by applying a random permutation. We train a neural network to predict which permutation was applied, taking the shuffled patches as inputs. We then use the inner representations learned by the neural network as input features to a low-capacity supervised classifier (see Figures 1 and 2 for illustration). We believe that permutations provide a promising avenue for self-supervised learning, as they are conceptually general enough to be applied across a large range of modalities, unlike colorization (Larsson et al., 2017) or rotations (Gidaris et al., 2018) that are specific to images. Moreover, the encouraging results of Noroozi & Favaro (2016) when

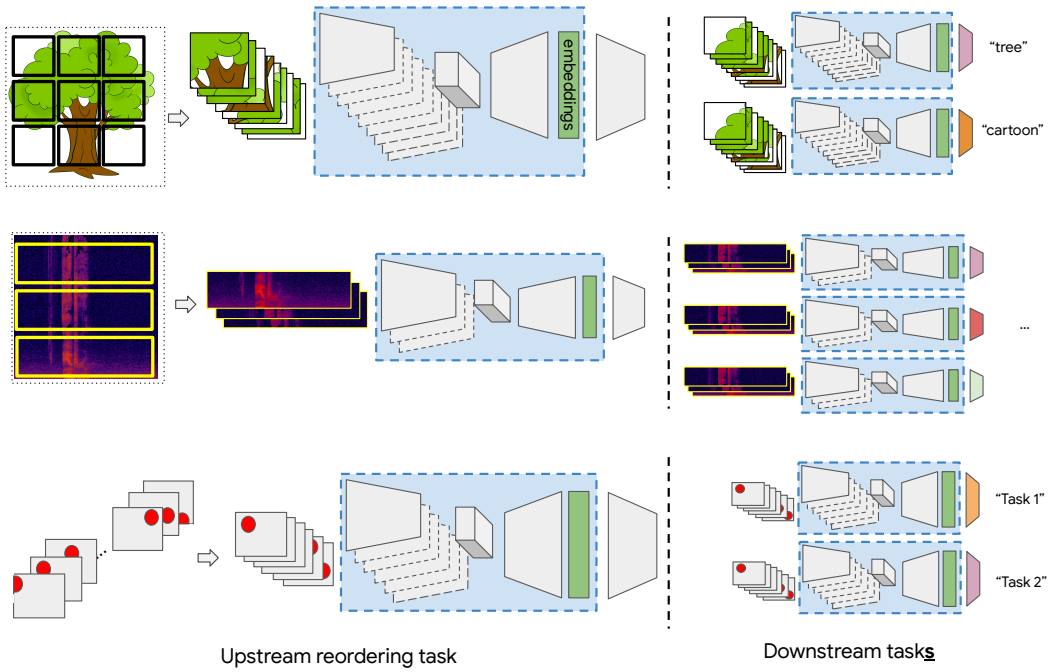

Figure 1: Permutations as a self-supervised technique can handle a variety of modalities with minimal changes to the network architecture. Dotted layers indicate weight sharing across input patches. The learned embeddings can be used for several downstream tasks. The inputs for the upstream task are permuted, while the inputs for the downstream task are not.

transferring learned image features for object detection and image retrieval inspire us to advance this method a step forward. However, including permutations into an end-to-end differentiable pipeline is challenging, as permutations are a discontinuous operation. Noroozi & Favaro (2016) circumvent this issue by using a fixed set of permutations and casting the permutation prediction problem as a classification one. Given that the number of possible permutations of n patches is $n!$, this approach cannot scale to exploiting the full set of permutations, even when n is moderately small.

In this work, we leverage recent advances in differentiable ranking (Berthet et al., 2020; Blondel et al., 2020b) to integrate permutations into end-to-end neural training. This allows us to solve the permutation inversion task for the entire set of permutations, removing a bottleneck that was heretofore sidestepped in manners that could deteriorate downstream performance. Moreover, we successfully demonstrate for the first time the effectiveness of permutations as a pretext task on multiple modalities with minimal modality-specific adjustments. In particular, we improve music understanding by learning to reorder spectrogram frames, over the time and frequency axes. We also improve video understanding by reordering video frames along time.

To summarize, we make the following two contributions.

- We integrate **differentiable ranking** into end-to-end neural network training for representation learning. This provides an efficient manner to learn in reordering tasks for all permutations, for larger numbers of patches. We show that this drastic increase in the number of permutations improves the quality of learned representations for downstream tasks.
- We successfully demonstrate for the first time the effectiveness of permutations as a **general purpose self-supervision method**, efficient on multiple modalities with extremely minimal modifications to the network. Additionally, the pre-trained representations perform well across diverse tasks of the same modality. We purposefully divert from domain-specific transformations, often inspired by data augmentation techniques, as predicting the permutation applied to input patches is not restricted either to a domain, nor a modality or a dimensionality. This also creates opportunities for applications beyond the scope of what we illustrate in our work.

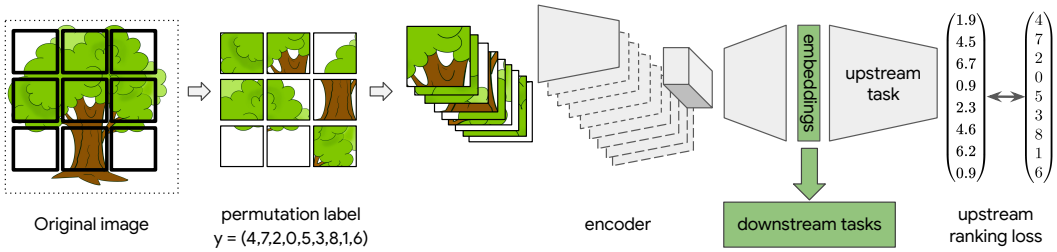

Figure 2: Data processing and upstream training procedures. All permutations are sampled on the fly, during training. To obtain embeddings reusable for downstream tasks, we truncate the network by removing the last few layers. This truncation depends on the task and is explained in Section 3.

The rest of the paper is organized as follows. In Section 2 we present the problem formulation and the methods used. In Section 3 we demonstrate the effectiveness of our experiments on audio, video, and image tasks. Further details can be found in the Appendix.

2 METHODS

2.1 GENERAL METHODOLOGY

To efficiently leverage the existence of large amounts of unlabeled data, we present a self-supervised pretext task that predicts the permutation applied to patches of an input. We do so in a manner that removes an existing bottleneck, and allows to use all possible permutations as targets during training. This pretext task is performed *upstream* and the internal representation learned by the pretext neural network can then be transferred and used on secondary *downstream* tasks – see Figures 1 and 2.

In the upstream task, for each data point, n *patches*, sub-parts $x_1, \ldots, x_n$ of of identical dimensions d are extracted. Their exact structure depend naturally on the modality: e.g. horizontal bands of an image, frequency bands of a spectrogram, frames of a video. Accordingly, the dimensions in d can represent height, width, channels, etc. These patches are then permuted randomly, and organized in a tensor X_i of dimension $n \times d$ (see Figure 2), which is paired to the applied permutation as a label y_i (see Section 2.2 for details on permutation encoding).

We then train the weights w of a neural network to minimize the loss between its output $f_w(X_i)$, of size n, and the encoding $y_i \in \mathbb{R}^n$ of the permutation applied to the n patches (see Figure 2). Note that the last operation of the network is a differentiable ranking operator y^*_ε. This operator, the encoding of the permutations for the labels, and the loss used in this upstream task are detailed in Section 2.2 below. The network, and the details of the data-processing pipeline generating the patches, are detailed in Section 2.3.

After upstream training on the initial dataset, the upstream network weights can be used to generate representations. By truncating the network, removing some of the last layers, we can extract an embedding of any input vector. These representations can be used in a *downstream* task to train a new network, with its own weights, minimizing a loss (typically classification or regression) between its output and the downstream task labels (see Figures 1 and 2).

We mostly evaluate our methods on downstream performance: the accuracy after training of the downstream network, on a task that is unknown during upstream training. However, the pretext reordering task can be of interest in and of itself, as in learning-to-rank problems (Liu, 2011), and we also report generalization performance in this task.

As an aside, in the Jigsaw puzzle reassembly task, Noroozi & Favaro (2016) show that the choice of permutation set matters when it comes to performance. They make use of the Hamming distance to choose a permutation set that it maximally separated. This permutation set is diverse but is not close to covering the entire permutation space. By supporting the full set of permutations, our approach does not suffer from this issue.

The downstream tasks are dataset-dependent. However, the network in the upstream reordering task requires minimal modification across tasks and modalities (see Section 3 for details). In this work, we demonstrate the effectiveness of our method on audio, video, and images, across several classification and regression tasks.

2.2 DIFFERENTIABLE RANKING METHODOLOGY

Our methodology for representation learning relies importantly on the ability to incorporate ordering or ranking operations in an end-to-end differentiable pipeline. During training for the upstream task, the last two layers of the network consist of: a vector of score values $\theta_w(X) \in \mathbb{R}^n$, and network outputs $f_w(X) = y^*_\varepsilon(\theta_w(X)) \in \mathbb{R}^n$, using a differentiable ranking operator y^*_ε that we detail here. The goal of the upstream task is to find good parameters w such that the network outputs $f_w(X)$ correctly recover the label y representing the permutation applied to the patches in X.

In earlier works using permutations as pretext task (Noroozi & Favaro, 2016; Lee et al., 2017), training with permutation labels is achieved by reducing the permutation inversion task to classification. More specifically, it is encoding a fixed number of permutations $L \ll n!$ as classes. Each class is represented by a one-hot vector, and network outputs are logits $\theta_w(X)$ of size L, leading to a prediction by taking a softmax among these L classes. This approach is obviously limited: representing all the permutations requires in principle $n!$ classes, which is quickly not manageable, even for small values of n. Further, this does not take into account the similarity between permutations: with this encoding, permutations are orthogonal, no matter how similar they are.

We address these issues by having network outputs $\theta_w(X)$ of size only n, and interpreting their relative orders as the predicted permutation (e.g. $y = (0, 1, 2, 3)$ if they are in decreasing order, predicting the identity permutation). The pretext labels also encode the permutations in this manner. This gives a unique encoding to each permutation, operates in dimension n, and can be computed with sorting operations in time $O(n \log n)$. Further, small distances in these encodings naturally represent similar permutations.

However, applying directly a ranking operator y^* on the last layer of the network would not allow for backpropagation of gradients: the function of the weights $w \mapsto L(y^*(\theta_w(X)); y)$ is piece-wise constant. Small changes in the weights w induce either large jumps or no change in value at all, its gradients are 0 almost everywhere, and undefined otherwise. In order to overcome this matter, we consider instead two differentiable ranking operations, one using stochastic perturbations, introduced in Berthet et al. (2020) and another one using regularization (Blondel et al., 2020b). These operations, denoted here by y^*_ε, map any vector of k values to a point in the convex hull of permutation encodings in dimension k (e.g. $(0.1, 0.9, 2.2, 2.8)$ over 4 elements). They can be thought of analogous to the softmax operator for ranking. They share some of its properties: good approximation of the original function, differentiability in the input values θ with non-zero derivatives everywhere, ease of computation, and tuning by a temperature parameter ε (see Appendix A.1 for further details).

Adjusting the network parameters w requires a notion of loss function between $y^*_\varepsilon(\theta)$ and y. For the version of $y^*_\varepsilon(\theta)$ based on stochastic perturbations (Berthet et al., 2020), we use the associated *Fenchel–Young loss* (Blondel et al., 2020a), that act directly on $\theta = \theta_w(X)$ (outputs of the vector, and inputs of the sorting operations), written here as $L_{\mathsf{FY}}(\theta; y)$ (see Appendix A.1). This loss is convex in θ, smooth, equal to 0 if and only if $y^*_\varepsilon(\theta) = y$. Its gradients are given by

$$\nabla_\theta L_{\mathsf{FY}}(\theta; y) = y^*_\varepsilon(\theta) - y\,.$$

We call this loss "Perturbed F-Y" in our empirical results. For the regularized version of $y^*_\varepsilon(\theta)$ (Blondel et al., 2020b), we use

$$\frac{1}{2}\|y^*_\varepsilon(\theta) - y\|^2.$$

We call this loss "Fast Soft Ranking (FSR)" in our empirical results. We opt for these two losses for their good theoretical properties and $O(n \log n)$ complexity. Other choices (Mena et al., 2018; Cuturi et al., 2019; Vlastelica et al., 2019; Rolínek et al., 2020; Grover et al., 2019) are also possible, potentially with higher computational cost, or regions with zero gradient.

2.3 IMPLEMENTATION AND ARCHITECTURE

Data-processing When constructing the self-supervised task, inputs that can be treated as images (i.e. with width and height) are sliced into patches. This slicing is controlled by two variables n_x and n_y, determining respectively the number of columns and rows used. In Noroozi & Favaro (2016), 9 square patches are used. Across modalities, different slicing scheme can lead to better performance. In video analysis, there is only one axis along which to permute: time. In audio processing, the downstream task may benefit from the pretext task using frequency slices of the spectrogram instead of time slices (see Figure 1 for illustration). We explore these questions Section 3.

Upstream task. For the reordering pretext task, performed upstream, we use a Context Free Network (CFN) from Noroozi & Favaro (2016). This network uses an AlexNet (Krizhevsky et al., 2012) backbone which processes each patch individually while sharing weights across all patches as shown in Figure 1. By processing each patch independently, but with shared weights, the network cannot rely on global structure. After the shared backbone, the patches are passed together through two fully connected layers. The layer represents the predicted ranks of the input permutation. Although CFN was originally designed for images, one of our empirical contributions is to show its application to a number of separate modalities without any change to the training mechanism (Figure 1). In particular, we show important performance gains on audio tasks.

Downstream task. For the downstream task, we use classifiers with low capacity: a 3-layer multilayer perceptron (MLP) for audio and video task, and a linear classifier for images. It is trained on embeddings extracted at the first aggregate fully connected layer of the pretext network (whose weights are frozen during this part). The MLP's output layer is task dependent. For a regression downstream task, the output of the MLP is a single scalar and the downstream model is trained to minimize mean-squared error. On the other hand, for a downstream classification task, the output of the MLP is a softmax over the class logits, and we train the downstream model by minimizing the cross-entropy between the predictions and the true labels.

3 EXPERIMENTS

In this section, we demonstrate the effectiveness of permutation-based self-supervised learning as measured by the test accuracy in downstream tasks across several modalities. All experiments were carried out in Tensorflow (Abadi et al., 2016) and run on a single P100 GPU. For all modalities, in addition to the downstream task, we also report the performance on the upstream task using partial ranks, the proportion of patches ranked in the correct position. We will open-source our codebase for reproducibility and reusability.

3.1 AUDIO

Experimental setup. The NSynth dataset (Engel et al., 2017) offers about 300,000 audio samples of musical notes, each with a unique pitch, timbre, and envelope recorded from 1,006 different instruments. The recordings, sampled at 16kHz, are 4 seconds long and can be used for 3 downstream classification tasks: predicting the instrument itself (1,006 classes) the instrument family (11 classes) and predicting the pitch of the note (128 classes). Since the pitch estimation is a regression task, we report in our results the mean squared error (MSE) for this task.

Audio processing systems usually take as input a log-compressed spectrogram of the recordings rather than the raw waveform, since it is a convenient and compact time-frequency representation of the input signal. Moreover, the 2D structure of the spectrogram allows us to use the 2D convolutional network as for images and videos.

We train our CFN with an AlexNet backbone on the upstream task of predicting the applied permutation for 1000 epochs, over mini batches of size 32 and with an Adam optimizer (Kingma & Ba, 2014) with a learning rate of 10^{-6}. After convergence, we evaluate the downstream generalization performance over the 3 NSynth tasks, by replacing the last layers of the network by a 3-layer MLP and replacing the random permutation by the identity. To understand the quality of the produced embeddings, we vary the number of examples used to train train the downstream task and report the results for different data regimes.

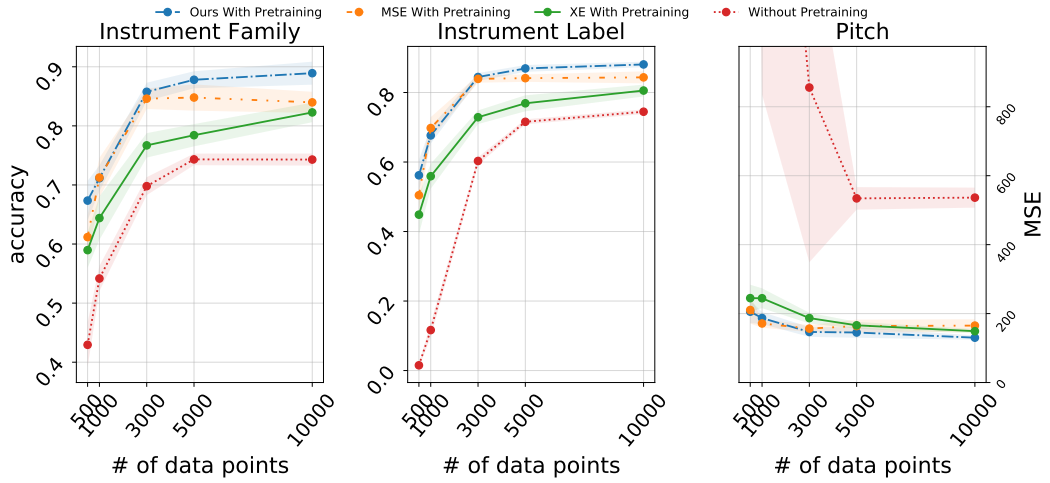

Figure 3: Performance of our permutation-based pretraining over 3 audio tasks when varying the number of data points in the downstream task.

Task	Random Embedding	Fixed Permutation	Fast Soft Ranking	Perturbed F-Y	MSE
Instr. Family (ACC)	0.46 $\pm$0.01	0.75 $\pm$0.02	0.84 $\pm$0.02	**0.85** $\pm$0.03	0.79 $\pm$0.02
Instr. Label (ACC)	0.35 $\pm$0.04	0.70 $\pm$0.03	0.72 $\pm$0.03	**0.76** $\pm$0.04	0.71 $\pm$0.03
Pitch (MSE)	258.76 $\pm$3	144.48 $\pm$18	133.85 $\pm$19	**124.6** $\pm$14	156.08 $\pm$21
Partial Ranks Accuracy	-	-	0.57 $\pm$0.0	**0.58** $\pm$0.0	0.51 $\pm$0.0

Table 1: Performance on three downstream tasks with 1000 downstream data points taken from NSynth. ACC stands for accuracy, and MSE for mean squared error.

We compare the different variants of our method (number and nature of the patches) with 2 baseline alternatives: i) training the downstream architecture on an untrained encoder (referred to as Random Embedding) and ii) solving the same upstream task but using instead a finite set of 100 permutations as proposed by Noroozi & Favaro (2016) (Fixed Permutation).

Lastly, we compare different losses to train the permutation pretext task: a) cross entropy (XE) when learning over 100 permutations, b) MSE loss (MSE), c) soft ranking via perturbations (Perturbed F-Y) and d) soft ranking (Fast Soft Ranking).

Empirical results. First, we compare the different methods for different data regimes in the downstream task and report graphically the results in Figure 3, choosing the accuracy for the classification tasks and MSE for the regression one. We observe that in the low data regime our method strongly outperforms fully supervised model trained end-to-end rather than on a fixed embedding. Moreover, even though the margin closes when adding more data, we observe that the features obtained greatly assist in the downstream learning process. We also observed that pretraining is particularly impactful for pitch estimation which aligns with results found by Gfeller et al. (2020).

We report in Table 1 the results for 1000 downstream examples (see 5 and 6, in the Appendix A.2 for 500 and 5000). Those experiments were run using 10 frequency bands, which corresponds to 10! permutations. This number of potential classes discards the use of classification loss: from a very practical point of view, such a huge number of classes would create a weight matrix on the last dense layer that would not fit in memory.

We first observe that random embeddings performs poorly but does represent a good baseline to be compared to. Second, when training the encoder with the fixed set of permutations, we observe a significant increase in performance that confirms experimentally the results reported by (Noroozi & Favaro, 2016). In the first three rows of Table 1, we report the results of our methods in the downstream metrics. We first observe that even with a mean squared error loss, the performance

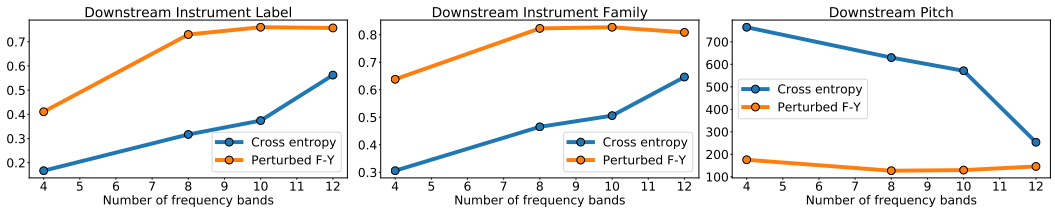

Figure 4: Performance on the NSynth dataset's downstream tasks, as a function of the number of frequency bands used for the pretext task on the audio experiment. With lower number of patches, the upstream task can become too easy to lead to good feature learning, risking overfitting.

on the downstream task is comparable or better than the fixed permutation method and we show that using a ranking loss further increases the performance. Those results tend to confirm that i) permutation is an interesting pretext task, ii) considering all possible permutation helps building better representations and iii) the use of a ranking loss is the right choice of loss for such a task.

Furthermore, Fig.4 shows the effect of the number of frequency bands on the downstream performance. It shows that, as the number of permutations grows, the performance over the downstream task increases, providing better representations. However, it eventually stops increasing or can even decrease the performance.

We report in the last row of Table 1 performance in the pretext task. Good performance on the downstream task is often connected to good performance on the pretext task. Here, we measure performance by the ability of the CFN network to reorder the shuffled inputs, reporting the proportion of items ranked in the correct position.

Task	Frequency	Time	Time-Frequency
Instr. Family (ACC)	**0.82** ± 0.003	0.79 ± 0.004	0.73 ± 0.003
Instr. Label (ACC)	**0.75** ± 0.01	0.68 ± 0.002	0.45 ± 0.01
Pitch (MSE)	**136.68** ± 12.63	206.11 ± 7.32	**137.81** ± 13.02

Table 2: Slicing biases on Nsynth downstream tasks.

Time-frequency structure and permutations. Unlike images, the horizontal and vertical dimensions of a spectrogram are semantically different, respectively representing time and frequency. While Noroozi & Favaro (2016) only exploited square patches, experimenting with audio allows exploring permutations over frequency bands (horizontal patches), time frames (vertical patches) or square time-frequency patches, and comparing the resulting downstream performance Table 2 reports a comparison between these three settings. Overall, shuffling along the frequency axis only is the best pre-training strategy. These results illustrate a particularity of the dataset: our inputs are single notes, many of them having an harmonic structure. In this context, learning the relation between frequency bands is meaningful both to recognize which instrument is playing, as well as which note (pitch). This also explains the poor performance of slicing along the time axis. Pitch is a time-independent characteristic, so the time structure is not relevant for this task. Moreover, musical notes have an easily identifiable time structure (fast attack and slow decay), which may make the task of reordering time frames trivial. We hypothesize that signals with a richer, non-stationary time structure, such as speech, would benefit more from shuffling time frames.

3.2 VIDEO

Experimental setup. Another modality where shuffling can assist learning is in video classification, as done by Lee et al. (2017). In this case, instead of slicing a single image with multiple patches, a time frame is treated as a patch and multiple frames are permuted. We sample these frames uniformly along the video, and use 20 frames. We experiment on the something-something dataset (Goyal et al., 2017), a dataset of videos labelled with 174 actions. We choose this dataset as many labels incorporate information about the dynamics (e.g. "Pulling something from right to left"), so that classification can be improved by learning the natural order of frames.

Experimental results. We report in Table 3 results on self-supervised learning with permutations on frames of video. Competitive standards for fully supervised performance (using the whole video) on this dataset are around 0.65. Our method, with drastically reduced amount of training data (3000 videos, 1.4% of the dataset), operating on 20 frames, still reaches 0.21. Interestingly, when using fewer frames than 20, the downstream performance does not improve above random chance. Hence, scaling to more patches is required to in this setting, which makes using a fixed number of permutations challenging. This may explain the poorer performance of the Fixed Permutation baseline on this task, and points to permutations as a promising avenue of exploration for self-supervised learning on videos.

In addition to the potential for self-supervised learning, performance on permutation task itself is interesting in the context of videos. We also report experimental results on the upstream reordering task itself in Table 3. In this case, the permutations acts on 20 frames, with an order of 2.10^{18} possible permutations. Our results show that a significant fraction of elements are correctly reordered, even though this particular permutation has in all likelihood not been observed during training.

Task	Fixed Permutation	Perturbed F-Y	Fast Soft Ranking	MSE
Label (ACC)	0.13 ± 0.01	0.19 ± 0.001	$\mathbf{0.21}$ ± 0.002	0.12 ± 0.03
Partial Ranks	-	0.27 ± 0.003	$\mathbf{0.29}$ ± 0.001	0.17 ± 0.03

Table 3: Up and downstream performance for video.

It is interesting to note that the methods that perform best on the pretext task also generally perform best on the downstream tasks. For example, we observe that our methods get higher performance than the MSE on the Jigsaw puzzle reassembly task. Correspondingly, our methods also achieve higher performance on the downstream tasks.

3.3 Images

Experimental setup. We also apply our methodology to images, using the MNIST dataset. In this experiment, we train a simpler network, whose context-free part consists of a small convolutional neural network (CNN), and add additional fully connected layers, so that the embeddings are extracted deeper in the network. In the downstream task, we use a linear classifier. We explore also several slicing methods, with horizontal and vertical bands, as well as with square patches. We also show that the representations generalize well to unseen data by training in the upstream task only five classes and evaluating in the downstream task on the remaining five classes (see Appendix A.2).

Experimental results. We focus on both the upstream and downstream performance for the final image task. Consistent with the other results, we find performance of the two differentiable ranks methods to be consistently higher than learning reordering with classification over fixed permutations or MSE. Additionally, we highlight the efficiency with which the upstream network learns the permutation inversion task. The results in Table 4 are with 4 horizontal bands and the entire label set for the upstream and downstream tasks.

Task	Fixed	Perturbed F-Y	FSR	MSE
Label - 3000 (ACC)	0.64 ± 0.001	$\mathbf{0.82}$ ± 0.002	$\mathbf{0.82}$ ± 0.002	0.77 ± 0.001
Label - 9000 (ACC)	0.67 ± 0.001	$\mathbf{0.86}$ ± 0.00	$\mathbf{0.85}$ ± 0.001	0.81 ± 0.002
Partial Ranks	-	0.97 ± 0.00	0.97 ± 0.00	0.96 ± 0.00

Table 4: Up and downstream performance on MNIST.

4 Conclusion

We present in this paper a general self-supervised learning method that uses permutations to learn high-quality representations. We demonstrate that our method outperforms previous permutation learning schemes by incorporating fully differentiable ranking as a pretext loss, enabling us to take advantage of all $n!$ permutations, instead of a small fixed set. In particular, we show significant improvements in low data regimes. We also demonstrate that our method can be applied seamlessly to improved downstream performance of several classification and regression tasks across audio, videos, and images.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pp. 173–182, 2016.

Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable perturbed optimizers. *arXiv preprint arXiv:2002.08676*, 2020.

Mathieu Blondel, André FT Martins, and Vlad Niculae. Learning with fenchel-young losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020a.

Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. *arXiv preprint arXiv:2002.08871*, 2020b.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132–149, 2018.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. Unsupervised cross-lingual representation learning for speech recognition. *arXiv preprint arXiv:2006.13979*, 2020.

Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. In *Advances in Neural Information Processing Systems*, pp. 6861–6871, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pp. 1422–1430, 2015.

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders, 2017.

Beat Gfeller, Christian Frank, Dominik Roblek, Matt Sharifi, Marco Tagliasacchi, and Mihajlo Velimirović. Spice: Self-supervised pitch estimation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1118–1128, 2020.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, volume 1, pp. 5, 2017.

Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. *arXiv preprint arXiv:1903.08850*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6874–6883, 2017.

Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 667–676, 2017.

Tie-Yan Liu. *Learning to rank for information retrieval*. Springer Science & Business Media, 2011.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665*, 2018.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pp. 69–84. Springer, 2016.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544, 2016.

Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*, 2017.

Morgane Rivière, Armand Joulin, Pierre-Emmanuel Mazaré, and Emmanuel Dupoux. Unsupervised pretraining transfers well across languages. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7414–7418. IEEE, 2020.

Michal Rolínek, Vít Musil, Anselm Paulus, Marin Vlastelica, Claudio Michaelis, and Georg Martius. Optimizing rank-based metrics with blackbox differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7620–7630, 2020.

Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. *arXiv preprint arXiv:1912.02175*, 2019.

A APPENDIX

A.1 DIFFERENTIABLE RANKING AND FENCHEL-YOUNG LOSSES

The differentiable operators for ranking that we consider map a vector of values to the convex hull of ranking vectors - i.e. vector whose entries are 1 to n. This set $\mathcal{C}$ is referred to as the *permutahedron* (see, e.g., Blondel et al. (2020b)). Both our approaches rely on the fact that for a vector $\theta \in \mathbb{R}^n$, the vector of ranks of θ in ascending order is given by

$$y^*(\theta) = \arg\max_{y \in \mathcal{C}} \langle y, \theta \rangle \,.$$

The two variations on differentiable ranking that we introduce are based on modifications of this formulation. The approach in Berthet et al. (2020) is to introduce random perturbations Z tuned by $\varepsilon > 0$ to the vector of values

$$y^*_\varepsilon(\theta) = \mathbb{E}[\arg\max_{y \in \mathcal{C}} \langle y, \theta + \varepsilon Z \rangle] \,.$$

Its value as well as its derivatives can be approximated by Monte-Carlo (MC) schemes (see Berthet et al., 2020, for details). In this work, we use a normal distribution for Z, and in learning experiments use $\varepsilon = 0.1$ and average over 10 MC samples. If we denote by $F_\varepsilon(\theta)$ the expectation of the maximum (rather than the argmax), and $\varepsilon\Omega$ its Fenchel dual over $\mathbb{R}^n$, we also have that

$$y^*_\varepsilon(\theta) = \arg\max_{y \in \mathcal{C}} \{\langle y, \theta \rangle - \varepsilon \Omega(y)\} \,.$$

This is the approach favored in Blondel et al. (2020b), where the function Ω is not defined as the dual of an expectation, but instead as an explicit convex regularizer over $\mathcal{C}$. In this paper, we use $\Omega(y) = \frac{1}{2}\|y\|^2$. The corresponding Fenchel-Young loss (Blondel et al., 2020a), whose definition extends to other problems than ranking, is defined for such convex regularized problems by

$$L_{\mathsf{FY}}(\theta; y) = F_\varepsilon(\theta) + \varepsilon\, \Omega(y) - \langle \theta, y \rangle \,,$$

where we recall that $F_\varepsilon = (\varepsilon\Omega)^*$ (Fenchel dual). As described in the main text, its gradient over the parameter θ is given by

$$\nabla_\theta L_{\mathsf{FY}}(\theta; y) = y^*_\varepsilon(\theta) - y \,.$$

Stochastic gradients can therefore be obtained by Monte-Carlo approximation, which we use to minimize this loss in this work.

A.2 ADDITIONAL EXPERIMENTAL RESULTS

Appendix Tables for additional information on the audio task (see next page) and on the split MNIST dataset.

Task	Random Embedding	Fixed Permutation	Fast Soft Ranking	Perturbed F-Y	MSE
Instr. Family (ACC)	0.44 ± 0.04	0.64 ± 0.04	$\mathbf{0.73}$ ± 0.05	$\mathbf{0.71}$ ± 0.05	0.62 ± 0.05
Instr. Label (ACC)	0.23 ± 0.04	0.52 ± 0.06	$\mathbf{0.56}$ ± 0.05	$\mathbf{0.55}$ ± 0.05	0.47 ± 0.05
Pitch (MSE)	256.20 ± 38.75	$\mathbf{188.37}$ ± 28.77	$\mathbf{186.03}$ ± 25.49	$\mathbf{183.96}$ ± 26.45	276.59 ± 53.46

Table 5: Performance on three downstream tasks on the NSynth dataset, with **500** downstream data points. ACC stands for accuracy, and MSE for mean squared error.

Task	Random Embedding	Fixed Permutation	Fast Soft Ranking	Perturbed F-Y	MSE
Instr. Family (ACC)	0.59 ± 0.02	0.91 ± 0.01	$\mathbf{0.93}$ ± 0.01	$\mathbf{0.94}$ ± 0.01	0.90 ± 0.05
Instr. Label (ACC)	0.49 ± 0.04	0.89 ± 0.01	$\mathbf{0.91}$ ± 0.01	$\mathbf{0.91}$ ± 0.01	0.89 ± 0.01
Pitch (MSE)	251.45 ± 5.36	$\mathbf{83.71}$ ± 8.55	97.98 ± 8.36	94.98 ± 8.05	108.46 ± 12.28

Table 6: Performance on three downstream tasks on the NSynth dataset, with **5000** downstream data points. ACC stands for accuracy, and MSE for mean squared error.

Task	Fixed Perturbations	Perturbed F-Y
Label - 9000 (ACC)	0.66 ± 0.001	$\mathbf{0.87}$ ± 0.001
Partial Ranks	-	0.97 ± 0.001

Table 7: Comparison of MNIST on up and downstream tasks on split label set.