Visual Permutation Learning

Rodrigo Santa Cruz Basura Fernando Anoop Cherian Stephen Gould

Abstract—We present a principled approach to uncover the structure of visual data by solving a deep learning task coined *visual permutation learning*. The goal of this task is to find the permutation that recovers the structure of data from shuffled versions of it. In the case of natural images, this task boils down to recovering the original image from patches shuffled by an unknown permutation matrix. Permutation matrices are discrete, thereby posing difficulties for gradient-based optimization methods. To this end, we resort to a continuous approximation using doubly-stochastic matrices and formulate a novel bi-level optimization problem on such matrices that learns to recover the permutation. Unfortunately, such a scheme leads to expensive gradient computations. We circumvent this issue by further proposing a computationally cheap scheme for generating doubly stochastic matrices based on Sinkhorn iterations. To implement our approach we propose *DeepPermNet*, an end-to-end CNN model for this task. The utility of DeepPermNet is demonstrated on three challenging computer vision problems, namely, relative attributes learning, supervised learning-to-rank, and self-supervised representation learning. Our results show state-of-the-art performance on the Public Figures and OSR benchmarks for relative attributes learning, chronological and interestingness image ranking for supervised learning-to-rank, and competitive results in the classification and segmentation tasks of the PASCAL VOC dataset for self-supervised representation learning.

Index Terms—permutation learning, self-supervised learning, relative attributes, representation learning, learning-to-rank.

1 Introduction

Machine learning algorithms often use the structure of data in order to provide accurate and efficient solutions to difficult problems. For instance, in supervised learning-to-rank, list-wise methods exploit structural information beyond pairs of samples in order to learn better rankers [10]. Structured prediction models such as CRFs [40] and Structured SVMs [71] explicitly model what structural information should be exploited by the learning algorithm. Therefore, we can say that the structural information implicit in data is crucial to machine learning applications.

Similarly, visual data encompasses rich spatial (and temporal) structure, which is often useful for solving a variety of problems. For instance, surrounding background usually offers strong cues for object recognition, sky and ground usually appear at predictable locations in an image, and objects are made up of known parts at familiar relative locations. Such structural information within visual data has been used to solve several problems, such as object detection and semantic segmentation [51, 62, 49].

Following these ideas, we present a learning framework that uses the inherent structure in data to solve visual recognition tasks. As an example, consider the task of assigning a meaningful order (with respect to some visually salient attribute) to the images shown in the left panel of Figure 1. Indeed, it is difficult to solve this task by just processing a single image or even a pair of images at a time where extracting visual cues related to the attributes is limited. The task becomes feasible, however, if one exploits the structure and the broader context by considering the entire set of images jointly. Only then do we start to recognize shared patterns that could guide the algorithm towards a solution. A similar task involves recovering the spatial structure in images. For example, consider the task shown in the right panel of Figure 1. Here we ask the question "*given shuffled image patches, can we recover the original image?*". Although this is a difficult task (even for a humans), it becomes straightforward once we identify the object in the patches (e.g., a cat), and arrange the patches for the recognized object, thereby recovering the original image.

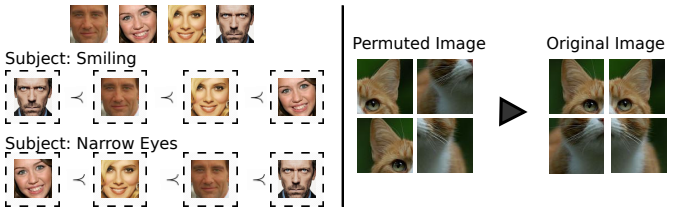

Fig. 1. Illustration of the proposed permutation learning task. The goal of our method is to jointly learn visual features and the predictors to solve the visual permutation problem. This can be applied to ordering image sequences (left) or recovering spatial layout (right).

Such shuffled images can be generated cheaply and in abundance from natural images. The problem of recovering the original image from shuffled ones can be cast in an unsupervised learning setting. Here the recovery task does not require any human annotations (and is thus unbiased given sufficient data [70]). Instead it uses the spatial structure as a supervisory signal. Such a learning task is commonly known as self-supervised learning [14, 23, 50, 53, 54, 24], and is very useful to learn rich features, especially in the context of training deep learning models, which often require large amounts of annotated datasets.

In this self-supervised context, Doersch et al. [14] show that the spatial layout of objects is a strong supervisory signal to learn transferrable image representations, while others [53, 50, 43] cast the problem of recovering the original image from shuffled ones as the prediction of a subset of permutations of image regions. Our current work reformulates the "unshuffling" problem allowing new applications and overcoming limitations of existing methods. More specifically, Misra et al. [50] model the problem via binary

- *R. Santa Cruz and S. Gould are with the Australian Centre for Robotic Vision at the Australian National University, Canberra. E-mail: firstname.lastname@anu.edu.au*
- *B. Fernando is with Artificial Intelligence Initiative, A*STAR, Singapore and the Australian Centre for Robotic Vision at the Australian National University, Canberra. E-mail: firstname.lastname@anu.edu.au*
- *A. Cherian is with the Mitsubishi Electric Research Labs (MERL) Cambridge, Massachusetts and the Australian Centre for Robotic Vision at the Australian National University, Canberra. E-mail: cherian@merl.com*

classification and learn to discriminate between correct and incorrect permutations of a sequence. Noroozi and Favaro [53] learn a multi-class classifier to distinguish between a few prototype permutations selected by a clustering procedure. Similarly, Lee et al. [43] formulate a multi-class problem on pairwise features. While making good progress towards the goal of recovering order, these approaches fail to consider structural information beyond pairs or subset of samples. Differently, we explore the entire structure of natural images encoded as sequences of image patches using a permutation prediction scheme which allows us to efficiently represent all possible permutations of the image regions.

As is clear, the aforementioned tasks essentially involve learning a function that can recover the order, i.e., infer the shuffling permutation matrix (see Figure 3). In learning such a function the model needs to infer the scene structure, visual attributes, or semantic concepts in visual data. Moreover, the knowledge acquired using such a learning framework could then be used to solve many other computer vision tasks, such as image ranking [22], image reconstruction [8], and object recognition [51].

Likewise, many machine learning applications can be described by the aforementioned permutation learning framework. For instance, the list-wise learning-to-rank problem [77] can be seen as a scheme to predict the correct permutation of a random set of samples given some criteria. Recommendation problems can be cast as selecting a subset of items permuted according to the users' profiles. In archeology, broken relics may be re-assembled by permuting fragments[8]. Therefore, a well defined learning framework for such a task would benefit different applications.

However, learning the correct permutation that shuffled the data is challenging. First, enumerating every possible permutation for a given set is usually infeasible since the number of permutations scales factorially with the cardinality of the set. As such, naively learning discriminative functions over enumeration of permutations is prohibitive. Second, a large amount of data is required to effectively learn the variations of a permutation problem, which requires more computational resources and efficient methods.

In this paper, we address the problem of learning to predict visual permutations. Towards this end, we propose a novel permutation prediction formulation and a model based on convolutional neural networks that can be trained end-to-end. This allows us to learn image representations suitable for predicting permutations and to exploit the structure existent in the data. Moreover, our formulation admits an efficient solution and allows our method to be applied to a range of important computer vision problems.

Our contributions are threefold. First, we propose the *Visual Permutation Learning* problem as a generic task to learn structural concepts intrinsic to natural images and ordered image sequences. Second, we formulate such a problem as the prediction of the permutation matrix that recovers the structure of the data from shuffled samples of it. Since permutation matrices are discrete, we extend our formulation to their nearest convex surrogate, doubly-stochastic matrices. From this proposed formulation, we develop an exact solution by deriving and solving a *bi-level optimization* problem and an approximated solution by using the iterative procedure *Sinkhorn normalization*. Last, we propose the *DeepPermNet* model, a end-to-end learning framework to solve the visual permutation problem using convolutional neural networks. Since our approaches are defined over continuous matrices and differentiable functions, the proposed model can be efficiently learned via backpropagation and stochastic gradient descent.

Initially, we evaluate how well our model can learn to predict permutation matrices from shuffled image sequences. We observe that our model can leverage the structure of large sequences to infer the shuffling permutation, while naive approaches are only able to explore small sequences. With our proposed approach validated, we apply our DeepPermNet model to three different applications: relative attributes, image ranking and self-supervised representation learning.

In Section 5.3, we demonstrate that our proposed approach can be used to compare images according to visual attributes by predicting permutations of unordered sets of images. We evaluate this strategy on the relative attributes task where we outperform state-of-the-art methods on the Public Figures and OSR datasets [57]. We also notice that our model is able to localize attributes without any explicit supervision.

In Section 5.4, we extend our inference for image sequences of arbitrary length by predicting permutations of fixed-length subsequences and aggregating the results with a sorting algorithm. Using this approach, we evaluate our model on learning-to-rank applications such as ranking scenes according to how interesting they look [30] and ranking cars according to their manufacturing date [44]. In both applications, we outperform the state-of-the-art methods in all utilized ranking metrics.

In Section 5.5, we show that our proposed formulation can be used to learn features in a self-supervised manner by exploring the structure and visual priors of natural images. Using our formulation as a self-supervised representation learning method, we achieve performance similar to the state-of-the-art methods on object classification, detection and segmentation on the Pascal VOC dataset [18, 17].

The current manuscript is an extension of the work Santa Cruz et al. [61] and it differs in the following ways. First, we present a deeper literature review and a preliminary section on topics important to the development of our framework in Sections 2 and 3, respectively. Second, in Section 4.3.1, we extend the visual permutation learning framework through bi-level optimization exploring an optimum solution to the doubly-stochastic matrix approximation. We also evaluate how such solutions impact upon applications performance. Last, in Section 5.4, we introduce a new application of our method that validates the effectiveness of our proposed model for long sequences.

2 Related Work

Broadly speaking, the permutation learning problem consists of learning a meaningful order for a collection of images based on some predetermined criterion. Variations of this task have been studied extensively by many scientific communities. In computer graphics, the jigsaw puzzle problem consists of reconstructing an image by a set of puzzle parts [11, 65]. In the same fashion, structured problems including DNA or RNA modeling in biology [48] and re-assembling relics in archeology [8], can be modeled as permutation learning problems. Although in this paper we limit our scope to computer vision, and review below topics that are relevant to the applications considered in the sequel.

Visual Attributes. Visual attributes are human understandable visual properties shared among images. They may range from simple visual features (such as "narrow eyes" and "bushy eyebrows" in faces) to semantic concepts (like "natural" and "urban" scenes), or subjective concepts (such as "memorability" and "interestingness" of images). Due to the expressiveness of visual attributes,

researchers have successfully used them for many applications, including image search [38], fine-grained recognition [7], and zero-shot learning [57, 41].

Visual attributes are traditionally treated as binary predicates indicating the presence or absence of certain properties in an image. From this perspective, most of the existing methods use supervised machine learning, whose goal is to provide mid-level cues for object and scene recognition [19], or to perform zero-shot transfer learning [41].

However, there are also methods that can discover binary visual attributes in an unsupervised way [64, 31]. Huang et al. [31] use unsupervised discriminative clustering and cluster membership as a soft form of supervision to discover shared attributes. In contrast, our formulation directly learns the properties of visual attributes in a data driven manner in a single end-to-end trainable network by using the entire structure of the data.

A more natural view on visual attributes is to measure their strength in visual entities. For instance, Parikh and Grauman [57] introduced the problem of relative attributes, in which pairs of visual entities are compared with respect to their relative strength for any specific attribute. This problem is usually cast as a learning-to-rank problem using pair-wise constraints. Following this idea, Parikh and Grauman [57] propose a linear relative comparison function based on the well-known Rank-SVM [35], while Yu and Grauman [80] uses a local learning strategy.

With the recent success of deep learning methods in computer vision, CNN-based methods to tackle the relative attributes problem have been developed. Souri et al. [69] jointly learn image representation and ranking network to perform pair-wise comparisons according to a certain attribute. Similarly, Singh and Lee [67] propose to combine spatial transformer networks [33] and rank networks to localize, in addition to compare visual attributes. Differently from our proposed approach, the aforementioned methods use only pair-wise relationships and do not leverage structure within longer image sequences.

Supervised Learning-to-Rank. The objective of supervised learning-to-rank is to learn how to order an unseen sequence from a training set of correctly ordered sequences according to some predefined criterion. This topic has been explored by the scientific community with applications in active learning [46], zero-shot learning [57] and person re-identification [73]. In this paper, however, we will focus on image ranking where the goal is to order sequences of images according to some visual criterion.

Learning-to-rank algorithms can be categorized by the way they process the training sequences. Point-wise methods [12] learn from individual elements in the sequence and are easy to train, but prone to over-fitting. Pair-wise methods [35, 63] formalize the learning task as classification of pairs into correctly and incorrectly ordered pairs. They are efficient, but limited to ranking information of training pairs. List-wise methods [10, 78] optimize objective functions defined over entire sequences. However, they are more computationally expensive, yet very effective because they are able to explore the structural information present in the training sequences. Recently, Fernando et al. [22] introduced an approach between pair-wise and list-wise methods that explores subsequences in order to learn a global ranking function. Our proposed method belongs to this family of rankers, however, our method is CNN based and is able to learn image representations and ranking function jointly from the pixel data.

Self-Supervised Representation Learning. The main idea of self-supervision is to exploit supervisory signals, intrinsically in the data, to guide the learning process. In this learning paradigm, a model is trained on an auxiliary task that provides an intermediate representation that can be used as generic features in other tasks. In deep learning, these approaches are well-suited as a pre-training procedure in situations when there is not sufficient data to support fully supervised learning [26, 47].

Self-supervised learning has attracted a lot of attention recently and many methods have been proposed by the computer vision community. The only common objective of these methods is to learn visual representations without human supervision, and they differ greatly in the proposed pretext task and supervisory signal. For example, Doersch et al. [14] use spatial co-location of patches in images, Wang and Gupta [74] use object tracking in videos to provide similar representations for corresponding objects, Fernando et al. [23] use odd-one-out question answering, Pathak et al. [58] explore image context to recover missing parts in an image, Pathak et al. [59] exploit low-level motion-based grouping cues, Noroozi et al. [54] propose to count visual primitives in images, and Gidaris et al. [24] explore geometric transformations of images like 2d rotations. In contrast, our proposed method is generic and can be used to solve a broader set of problems.

On the other hand, there are pretext tasks that can be useful themselves. Isola et al. [32] learn to group visual entities based on their frequency of co-occurrence in space and time. Zhang et al. [81] propose a model to provide plausible color versions for grayscale images. Donahue et al. [16] build a generative model for natural images. Note, however, that these methods are highly engineered for their training task and they can not be easily extended to deal with other applications. On the other hand, our method is a general framework able to solve different problems.

A recent work closely related to ours is Noroozi and Favaro [53] that also proposes to train CNNs for solving image-based jigsaw puzzles. However, different from us, they train a CNN to predict only a tiny subset of possible permutations generated from an image shuffling grid of size 3×3 (specifically, they use only 100 permutations from the set of 362k possible permutations). Lee et al. [43] propose similar schema to order sequences of frames in videos. Our method, instead, can handle the full set of permutations and is scalable to finer shuffling grids (e.g., 4×4, 5×5, and so on). In addition, our scheme is generic and can also be used to solve different kinds of learning-to-rank problems.

3 PRELIMINARIES

In this section, we review the following background topics that we use in the subsequent sections for deriving our model for permutation learning: permutation matrices, doubly stochastic matrices and bi-level optimization.

3.1 Permutation Matrices

In matrix theory, a permutation matrix is a binary square matrix that has exactly a single unit value in every row and column, and zeros elsewhere. These matrices are used to compactly represent permutations of elements in an ordered sequence. For instance, given an ordered sequence $S = \langle a_1, \dots, a_n \rangle$ of n elements any permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ can be uniquely represented by a permutation matrix P_π. Furthermore, if we describe the original ordered sequence as a column vector, then any desired permutation π can be obtained by a simple matrix-vector multiplication,

$$S_\pi = P_\pi \, S \tag{1}$$

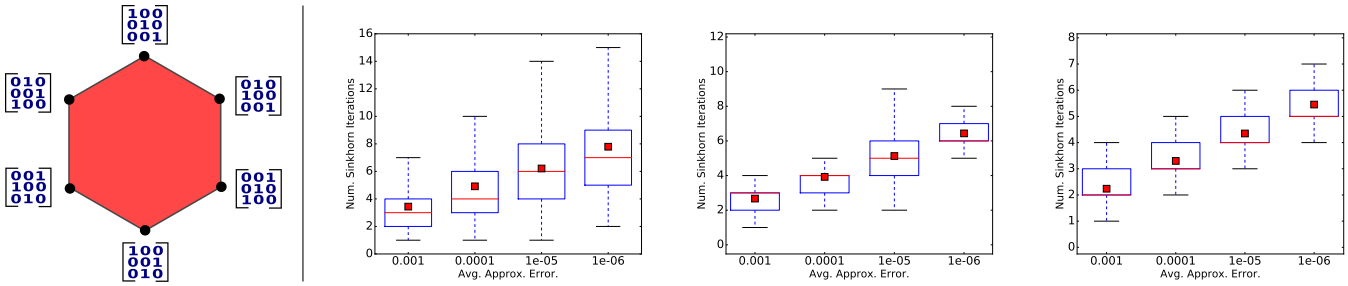

Fig. 2. Far Left: Illustration of Birkhoff polytope for $n \times n$ permutation matrices. From left to right: Boxplots of approximation error for the Sinkhron-Knopp algorithm applied on nonnegative random matrices of size 3x3, 6x6 and 9x9, respectively.

where P_π is formed by swapping the rows of the identity matrix according to the desired permutation π.

The set of $n \times n$ permutation matrices is a subgroup in the group of nonsingular matrices in $\mathbb{R}^{n \times n}$ with cardinality $n!$. These matrices have very interesting and useful properties. For instance, permutation matrices are closed under multiplication, that is, the product of two permutation matrices is again a permutation matrix representing the combined permutation. Likewise, the inverse of a permutation matrix is the inverse permutation, i.e., the permutation that recovers the original sequence from the permuted sequence, that can be efficiently computed by $P^{-1} = P^T$ (orthogonality).

3.2 Doubly Stochastic Matrices

A nonnegative matrix with the property that all its rows sum to one, is said to be a row stochastic matrix. Likewise, its transpose is said to be column stochastic matrix, since all its columns sum to one. A matrix that is simultaneously row and column stochastic is said to be a doubly stochastic (DSM). Mathematically, the requirements for a matrix $A \in \mathbb{R}^{n \times n}$ be doubly stochastic are,

$$A_{ij} \geq 0, \qquad A\,\mathbf{1} = \mathbf{1}, \qquad A^T\,\mathbf{1} = \mathbf{1}, \tag{2}$$

where $\mathbf{1}$ is an n-dimensional column vector of ones.

Permutation matrices are doubly stochastic matrices. In fact, according to the Birkhoff-von Neumann theorem [4, 72], any doubly stochastic matrix is a convex combination of finitely many permutation matrices. Thus, the set of $n \times n$ DSMs forms a convex hull for the set of $n \times n$ permutation matrices, known as the Birkhoff polytope $\mathcal{B}^n$. Consequently, it is natural to think of DSMs as convex relaxations of permutation matrices. Figure 2 illustrates the geometry of the Birkhoff polytope.

Doubly stochastic matrices, as well as permutation matrices, have a prominent history in engineering ranging from cryptography to topics in communication theory [9]. And approximating doubly stochastic matrices is a key problem in many applications. Here, we briefly demonstrate two efficient and principled approaches to fulfill such tasks. In later sections, we explain how these approaches can be applied in gradient based learners to solve our proposed permutation learning problem.

3.2.1 The Sinkhorn-Knopp algorithm

The Sinkhorn-Knopp [68] theorem states that if A is a real nonnegative squared matrix and has total support, then there exists a doubly stochastic matrix Q of the form,

$$Q = D_l A D_r \tag{3}$$

where D_l and D_r are diagonal matrices with positive main diagonals. Furthermore, there is a simple iterative procedure known as Sinkhorn Normalization, which can find D_l and D_r by repeatedly rescaling the rows and columns of a given matrix.

Knight [37] analyzes the convergence guarantees of Sinkhorn-Knopp algorithm. The author states that for a matrix A with entries in $[1, V]$, $\mathcal{O}(V\,|\log \epsilon|)$ iterations suffice to reach ϵ-near double stochasticity. However, we noticed empirically that only a few iterations are sufficient to reach acceptable approximations for most of the problems that we consider. Figure 2 shows empirical results for approximating DSMs from nonnegative random matrices of sizes 3×3, 6×6 and 9×9 using the Sinkhorn-Knopp algorithm. We find it to converge to an acceptable accuracy in approximately 4–6 iterations in most cases.

3.2.2 Norm Approximation

Norm approximation is a well known problem in the field of convex optimization [6]. The goal of the norm approximation problem is to approximate a vector, matrix, or space, as closely as possible, with deviation measured in the norm $\|\cdot\|$. Then, we can cast the doubly stochastic approximation problem as a norm approximation problem.

Formally, given an arbitrary matrix $A \in \mathbb{R}^{n \times n}$, its closest doubly stochastic matrix $Q \in \mathcal{B}^n$ can be obtained by solving the following problem,

$$\begin{aligned} \underset{Q \,\in\, \mathbb{R}_+^{n \times n}}{\text{minimize}} \quad & \|Q - A\| \\ \text{subject to} \quad & Q\,\mathbf{1} = \mathbf{1} \\ & Q^T\,\mathbf{1} = \mathbf{1} \end{aligned} \tag{4}$$

which is a convex problem. Thus, the solution is globally optimal. Moreover, it can often be stated as a quadratic program (QP) which can be solved efficiently by most publicly available solvers [29].

3.3 Bi-level Optimization

Given our interest in learning end-to-end models and solving DSMs approximation problems optimally in the derivation of our visual permutation learning framework, we need to deal with a bi-level optimization problem. We describe our specific problem in details in Section 4.3.1. Here we present a generic formulation for bi-level optimization problems and discuss how to solve them.

A bi-level optimization problem consists of an upper problem and a lower problem, whose objectives (and constraints) share a set of variables. More specifically, the former defines an objective over two sets of variables, say $\boldsymbol{x}$ and $\boldsymbol{y}$, and the latter binds $\boldsymbol{y}$ as an optimization problem parametrized by $\boldsymbol{x}$. We can state the problem mathematically as,

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f\left(\boldsymbol{x}, \boldsymbol{y}\right) \\ \text{subject to} \quad & \boldsymbol{y} \in \underset{\boldsymbol{y}'}{\text{argmin}}\; h\left(\boldsymbol{x}, \boldsymbol{y}'\right) \end{aligned} \tag{5}$$

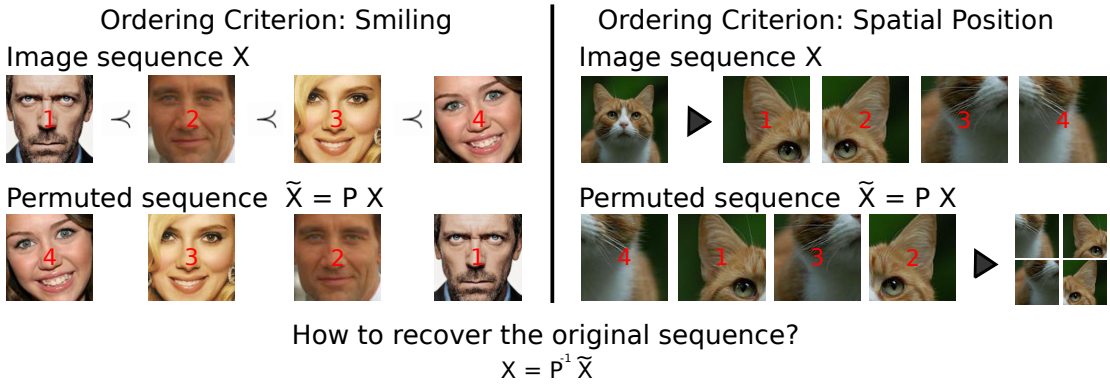

Fig. 3. Illustration of Permutation learning task as the prediction of the permutation matrix P from a given permuted image sequence $\tilde{X}$ such that $P^{-1} = P^T$ recovers the original ordered image sequence X.

where f and h are the upper and lower level objectives, respectively. Recently, bi-level optimization problems have found applications in machine learning and computer vision where they have been applied to hyper-parameter learning [75], image denoising [55], and video activity recognition [21].

We can solve such a problem by rewriting it as an equivalent single-level problem. This can be done by replacing the lower-level problem with an analytical solution (e.g., normal equations for a least-square problem) or a set of sufficient conditions for optimality (e.g., the KKT for convex problems). Then, the bi-level problem can be solved using the resulting single-level problem. However, for many lower-level problems either an analytical solution does not exist or the optimality conditions are hard to express. Furthermore, the resulting problem may be hard to solve.

However, if the lower-level problem can be solved efficiently, and there exists a method for finding the gradient at the current solution, we can solve the bi-level optimization problem via gradient descent. The main idea is to compute the gradient of the solution to the lower-level problem with respect to variables in the upper-level problem and perform updates of the form,

$$x \leftarrow x - \alpha \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} \frac{\partial y}{\partial x} \right) \Bigg|_{(x, y^\star)} \tag{6}$$

Note that the partial derivative $\frac{\partial y}{\partial x}$ may be difficult to compute, since it typically involves a parametrized argmin or argmax optimization problem. For a detailed explanation about procedures to differentiating such problems, we refer the readers to Faugeras [20] and Gould et al. [28].

4 Learning Visual Permutations

In this section, we describe our method for learning visual permutations. We start by formalizing the visual permutation learning task. Then, we describe our end-to-end learning algorithm, deep learning model, and inference procedure. We finalize this section by discussing alternative approaches to our method.

4.1 Task

Let us start by considering the tasks illustrated in Figure 3. Given a sequence of images ordered by a pre-decided visual criterion, for instance "smiling" (left) or spatial structure (right), we generate shuffled sequences by applying randomly sampled permutation matrices to the original sequences. Similarly, we can recover the original sequences from the shuffled ones by "un-permuting" them using the inverse of the respective permutation matrices. In this context, we define the *visual permutation learning* task as one that takes as input a permuted sequence and produces as output the permutation matrix that shuffled the original sequence.

Formally, let us define X to be an ordered sequence of l images in which the order explicitly encodes the strength of some predetermined criterion c. For example, c may be the degree of "smilingness" in each image. In addition, consider an artificially permuted version $\tilde{X}$ where the images in the sequence X are permuted by a randomly generated $l \times l$ permutation matrix P. Formally, the permutation prediction task is to predict the permutation matrix P from a given shuffled image sequence $\tilde{X}$ such that $P^{-1} = P^T$ recovers the original ordered sequence X.

We hypothesize that deep models trained to solve this task are able to capture high-level semantic concepts, structure, and shared patterns in visual data (In Section 5, we provide empirical evidence supporting this hypothesis). The ability to learn these concepts is important to perform well on the permutation prediction task, as well as to solve many other computer vision problems. Therefore, we posit that the features learned by our models are transferable to other related computer vision tasks as well.

Note that we describe our problem using only ordered sequences. This may seem a limitation, since structured information may be better represented by higher dimensional data. However, most of the time these higher order representations can be efficiently encoded as ordered sequences. For instance, the placement of 2-D image regions can be represented as an ordered sequence, where every position in the sequence is mapped to a distinct position in the 2D layout. Therefore, the proposed task can be used to solve different problems encoded in terms of ordered sequences.

4.2 Learning

With the visual permutation learning task described, we now focus on how to solve such a problem. We define a training set $\mathcal{D} = \{(X, P) \mid X \in \mathcal{S}^c \text{ and } \forall P \in \mathcal{P}^l\}$ composed by tuples of ordered image sequences X and permutation matrices P. Here, $\mathcal{S}^c$ represents a dataset of ordered image sequences, orderings implied by a predetermined criterion c. Each $X \in \mathcal{S}^c$ is composed of $X = \langle I_1, I_2, \ldots, I_l \rangle$, an ordered sequence of images I_i. The notation $\mathcal{P}^l$ represents the set of all $l \times l$ permutation matrices. Accordingly, the training set $\mathcal{D}$ is composed of all shufflings of each X by all P. Note that given an ordered X, such a dataset can be generated on-the-fly by randomly permuting the order, and the size of such permuted sets scales factorially on the sequence length l, providing a huge amount of data with low processing and storage cost to train high capacity models.

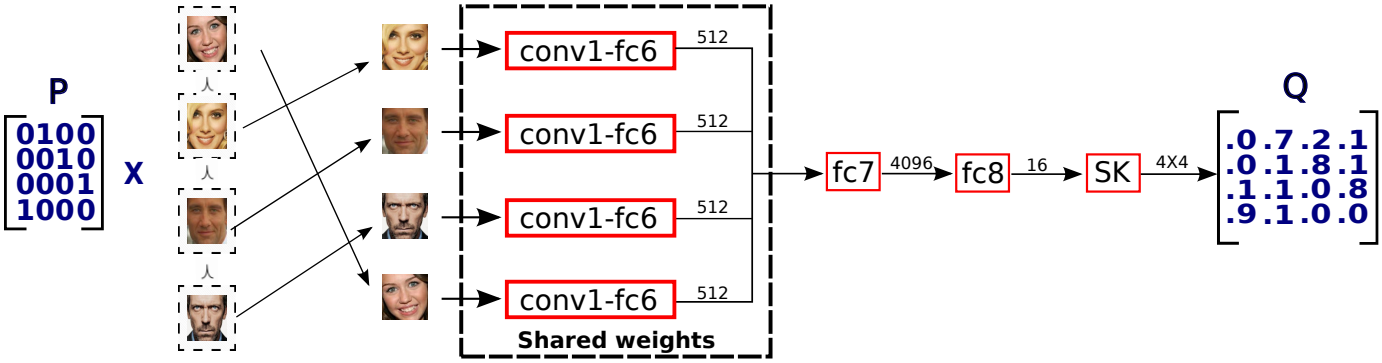

Fig. 4. DeepPermNet Architecture. It receives a permuted sequence of images as input. Each image in the sequence goes trough a different branch that follows the AlexNet [39] architecture from *conv1* up to *fc6*. Then, the outputs of *fc6* are concatenated and passed as input to *fc7*. Finally, the model predictions are obtained by applying the Sinkhorn Layer on the outputs of *fc8* layer.

Directly working with permutation matrices for deriving gradient-based optimization solvers is difficult as such solvers often start with an initial point and iteratively refines it using small steps (stochastic updates along gradient directions) towards an optimum. In this respect, working directly with discrete permutation matrices is not feasible. Thus, in this work, we propose to approximate inference over permutation matrices to inference over their nearest convex surrogate, the set of doubly-stochastic matrices. As discussed in Section 3, it is natural to think of DSMs as relaxations of permutation matrices.

Following these ideas, we propose to learn a parametrized function $f_\theta : \mathcal{S}^c \rightarrow \mathcal{B}^l$ that maps a fixed length image sequence (of length l) denoted by $\tilde{X}$ to an $l \times l$ doubly stochastic matrix Q. In the ideal case, the matrix Q should be equal to P. Then, our permutation learning problem can be described as,

$$\underset{\theta}{\text{minimize}} \quad \sum_{(X,P)\in\mathcal{D}} \Delta\left(P, f_\theta(\tilde{X})\right) + R\left(\theta\right), \tag{7}$$

where $\tilde{X}$ is the image sequence X permuted by the permutation matrix P, $\Delta(\cdot,\cdot)$ is a loss function, θ captures the parameters of the permutation learning function, and $R(\theta)$ regularizers these parameters to avoid overfitting.

4.3 Model

Having the task and learning objective defined, here we focus on the parametrization of the function $f_\theta\left(\cdot\right)$. Note that we wish to learn the image representation that captures the structure behind our sequences and also solves the permutation problem jointly. As such, the function $f_\theta(\cdot)$ should learn intermediate feature representations which encode semantic concepts about the input data. We propose to implement the function $f_\theta(\cdot)$ as a convolutional neural network (CNN), which is able to exploit large datasets and learn valuable visual features, that can be used as intermediate representations, while jointly learning the required mapping.

More specifically, we use a Siamese type of convolutional neural network in which each branch receives an image from a permuted sequence $\tilde{X}$ (see Figure 4). Each branch up to the first fully connected layer *fc6* uses the AlexNet architecture [39]. The outputs of *fc6* layers are concatenated and given as input to *fc7*. All layers up to *fc6* share the same set of weights. We refer to our proposed model as **DeepPermNet**.

Note that, if we ignore the structure of permutation matrices, this problem can have many different naive and infective solutions which we discuss later in Section 4.5. However, incorporating the inherent structure of permutation matrices can avoid the optimizer from searching over impossible solutions, thereby leading to faster convergence and better solutions. Thus, in the sequel, we discuss approaches for the permutation learning problem that explore the geometry of permutation matrices (using doubly-stochastic approximations).

4.3.1 Bi-level Optimization

Note that we wish to provide the closest doubly stochastic matrix $\hat{Q} \in \mathcal{B}^l$ from an arbitrary matrix $Q \in \mathbb{R}_+^{l\times l}$, e.g., CNN outputs. A principled way to achieve such a objective is to define and solve a convex quadratic program (QP). In this way, we can restate our learning problem in Equation 7 as,

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \sum_{(X,P)\in\mathcal{D}} \Delta\left(P, \hat{Q}\right) + R\left(\theta\right) \\ \text{subject to} \quad & \hat{Q} \in \underset{Q\in\mathcal{B}^l}{\text{argmin}} \left\|Q - f_\theta(\tilde{X})\right\|_F \end{aligned} \tag{8}$$

where $\mathcal{B}^l$ is the Birkhoff polytope.

This formulation is an instance of a bi-level optimization problem discussed in Section 3. Here, the loss minimization is the upper problem and the doubly stochastic approximation is the a lower problem. Furthermore, this formulation is well suited for gradient-based optimization methods, provided that we can compute the gradient of the argmin function in our lower level problem as other authors observed [15, 55, 21].

In order to simplify the gradient computation, we can approximate the lower level problem in Equation 8 by the following function $\boldsymbol{h}\left(\cdot\right)$,

$$\begin{aligned} \boldsymbol{h}\left(\boldsymbol{q}\right) \quad = \quad & \underset{\hat{\boldsymbol{q}}\in\mathbb{R}^n}{\text{argmin}} \quad \tfrac{1}{2}\left\|\hat{\boldsymbol{q}} - \boldsymbol{q}\right\|_2^2 - \mu\sum_{i=1}^n log\left(\hat{q}_i\right) \\ & \text{subject to} \quad A\hat{\boldsymbol{q}} = \boldsymbol{1} \end{aligned} \tag{9}$$

where $\boldsymbol{q}, \hat{\boldsymbol{q}} \in \mathbb{R}^n$ are the vectorized versions of Q and $\hat{Q}$ respectively $\left(n = l^2\right)$. The equality constraints defined by $A \in \mathbb{R}^{(2l)\times n}$ and the log-barrier function approximates the Birkhoff polytope. The hyper-parameter $\mu \geq 0$ controls the quality of the approximation. As $\mu \rightarrow 0$ the solution to the problem in Equation 9 converge to the solution to the lower problem in Equation 8.

Recently, Gould et al. [28] reviewed an earlier work of Faugeras [20] and collected some results on differentiating argmin and argmax optimization problems. Here, we will make use of their Lemma 4.2 that is restated in Lemma 4.1 below.

***Lemma 4.1.*:** Let $f : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}$ be a continuous function with first and second derivatives. Let $A \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$

with $\text{rank}(A) = m$. Let $\boldsymbol{g}(x) = \text{argmin}_{\boldsymbol{y}:A\boldsymbol{y}=\boldsymbol{b}} f(x, \boldsymbol{y})$. Let $H = f_{YY}(x, \boldsymbol{g}(x))$. Then,

$$\boldsymbol{g}'(x) = \left(H^{-1}A^T\left(AH^{-1}A^T\right)^{-1}AH^{-1} - H^{-1}\right) f_{XY}(x, \boldsymbol{g}(x)) \quad (10)$$

where $f_{YY} \doteq \nabla^2_{\boldsymbol{yy}} f(x, \boldsymbol{y}) \in \mathbb{R}^{n\times n}$ and $f_{XY} \doteq \frac{\partial}{\partial x}\nabla_{\boldsymbol{y}} f(x, \boldsymbol{y}) \in \mathbb{R}^n$.

However, note that $\boldsymbol{h}(\cdot)$ is a vector-valued function on vector domain. As such, we compute the derivative with respect to each input entry separately $\nabla_{q_i}\boldsymbol{h}(\boldsymbol{q}) \in \mathbb{R}^n$ and aggregate the results to compose the gradient $\nabla\boldsymbol{h}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$. Finally, $H = \nabla^2_{\hat{\boldsymbol{q}}\hat{\boldsymbol{q}}} f(\boldsymbol{q}, \hat{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ and $\frac{\partial}{\partial q_k}\nabla_{\hat{\boldsymbol{q}}} f(\boldsymbol{q}, \hat{\boldsymbol{q}}) \in \mathbb{R}^n$ where $f(\boldsymbol{q}, \hat{\boldsymbol{q}}) \in \mathbb{R}$ is the objective in Equation 9, can be obtained using the following partial derivatives,

$$H_{i,j} = \frac{\partial f^2(\boldsymbol{q}, \hat{\boldsymbol{q}})}{\partial\hat{q}_i\partial\hat{q}_j} = [\![i = j]\!]\left(1 + \mu\hat{q}_i^{-2}\right) \quad (11)$$

$$\frac{\partial}{\partial q_k}\nabla_{\hat{\boldsymbol{q}}} f(\boldsymbol{q}, \hat{\boldsymbol{q}}) = -[\![i = k]\!] \quad (12)$$

where $[\![\cdot]\!]$ is the indicator function, evaluating to one if its argument is true and zero otherwise. Note the gradient $\nabla_{q_i}\mathbf{h}(\boldsymbol{q})$ can be efficiently computed because H is a diagonal matrix and the derivative $\frac{\partial}{\partial q_k}\nabla_{\hat{\boldsymbol{q}}} f(\boldsymbol{q}, \hat{\boldsymbol{q}})$ does not depend on $\boldsymbol{q}$. Finally, the gradient of the loss with respect to the inputs can be easily obtained by applying the chain rule.

4.3.2 Sinkhorn Normalization

Despite providing the optimal solution for the DSM approximation, the bi-level optimization approach may be computationally expensive, since we need to solve an optimization problem for every sample in the training batches. Alternatively, we can resort to an approximate solution based on the Sinkhorn-Knopp algorithm discussed in Section 3.

Inspired by Adams and Zemel [2], here we propose a CNN layer that performs Sinkhorn normalization. Consider a matrix $Q \in \mathbb{R}_+^{l\times l}$, which can be converted to a doubly stochastic matrix by repeatedly performing row and column normalizations. Define row $R(\cdot)$ and column $C(\cdot)$ normalizations as follows,

$$R_{i,j}(Q) = \frac{Q_{i,j}}{\sum_{k=1}^{l} Q_{i,k}}; \quad C_{i,j}(Q) = \frac{Q_{i,j}}{\sum_{k=1}^{l} Q_{k,j}} \quad (13)$$

Then, the Sinkhorn normalization for the n-th iteration can be defined recursively as:

$$S^n(Q) = \begin{cases} Q, & \text{if } n = 0 \\ C\left(R\left(S^{n-1}(Q)\right)\right), & \text{otherwise.} \end{cases} \quad (14)$$

The Sinkhorn normalization function $S^n(\cdot)$ is differentiable and we can compute its gradient w.r.t. the inputs by unrolling the normalization operations and propagating the gradient through the sequence of row and column normalizations. For instance, the partial derivatives of the row normalizations can be defined as,

$$\frac{\partial\Delta}{\partial Q_{p,q}} = \sum_{j=1}^{l}\frac{\partial\Delta}{\partial R_{p,j}}\left[\frac{[\![j = q]\!]}{\sum_{k=1}^{l} Q_{p,k}} - \frac{Q_{p,j}}{\left(\sum_{k=1}^{l} Q_{p,k}\right)^2}\right] \quad (15)$$

where Q and R are the inputs and outputs of the row normalization function. The derivative of the column normalization can be obtained by transposing indexes appropriately. In practice, before applying the Sinkhorn normalization, we add a small value ($\approx 10^{-3}$) to each entry of Q as a regularization term to avoid numerical instability.

Despite being a principled and efficient approach, the Sinkhorn normalization layer may have a notorious drawback from the CNN optimization point of view – the problem of vanishing gradients in deep networks [27]. This may happen because each normalization can be seen as an extra layer to the network which makes the network deeper. However, as observed for random matrices in Figure 2, a small number of normalizations are sufficient to approximate the doubly stochastic matrix from CNNs raw outputs, and consequently the vanishing gradients problem is avoided.

4.4 Inference

Finally, we describe the last component of our approach, the inference procedure. Our main goal is to recover the original image sequence from a permuted sequence. Thus, our inference consists of approximating the closest permutation matrix $\hat{P}$ from the predicted doubly stochastic matrix Q. This problem can be described as,

$$\begin{aligned} \hat{P} \in \underset{P}{\text{argmin}} \quad & \|P - Q\|_F \\ \text{subject to} \quad & P\cdot\mathbf{1} = \mathbf{1} \\ & \mathbf{1}^T\cdot P = \mathbf{1} \\ & P \in \{0, 1\}^{l\times l} \end{aligned} \quad (16)$$

where $\hat{P}$ is our approximated permutation matrix from Q.

This problem is an instance of a mixed-boolean program and can be efficiently solved by branch-and-bound methods available in public solvers [13]. These methods begin by finding the optimal solution to the convex relaxation of the problem without the boolean constraints. If the optimal solution has any non-boolean variables, it creates new subproblems where the variables are more tightly constrained and this process is repeated until a solution that satisfies all boolean constraints is found.

After solving this problem to obtain $\hat{P}$, we transpose it to compute the inverse permutation matrix since $\hat{P}^T = \hat{P}^{-1}$. Then we can recover the original sequence X from the permuted sequence $\tilde{X}$ as,

$$X = \hat{P}^T\tilde{X}. \quad (17)$$

4.5 Alternative Approaches

In this section, we describe alternative approaches to solve the visual permutation learning problem. Overlooking all nice properties of permutation matrices, we can reformulate the permutation learning problem as a regression on the correct ordered sequence. More specifically, we can explicitly lean to predict the correct position of each item in a given shuffled input sequence by minimizing the euclidean loss between the correct sequence and the predictions. However, this solution may focus on correcting outliers, like swapping the first element by the last, which generates most part of the overall loss leading to a suboptimal solution.

Likewise, we can follow a discriminative setup (as done in [53, 43]) and cast our problem as a multi-class classification problem which we enumerate every possible permutation as a independent class. However, this solution is not feasible in practice since the number of parameter and predictors in the model scales factorial with the the input length. For instance, for a sequence of length 8 we need 40320 classes, which is intractable even for deep models.

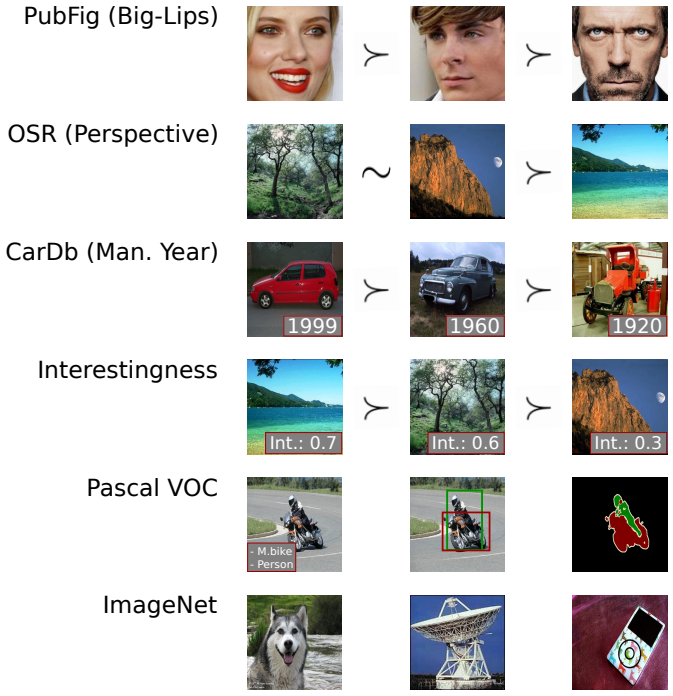

Fig. 5. Datasets used in our experiments: PubFig and OSR [57], CarDb [44], Interestingness [44], Pacal VOC [18, 17] and ImageNet [39].

On the other hand, we can use the permutation matrices formulation only to avoid the aforementioned enumeration problem and cast our problem as an l^2 binary classification problem by optimizing the combination of sigmoid outputs and cross-entropy loss,

$$\Delta(P,Q) = -\frac{1}{l^2}\sum_{ij}^{l\times l}\Big[P_{i,j}\log\left(Q_{i,j}\right) + (1-P_{i,j})\log\left(1-Q_{i,j}\right)\Big] \tag{18}$$

where each entry $P_{i,j}$ is a binary entry in the target permutation matrix P and $Q_{i,j}$ is an arbitrary prediction outputted by the function $f_\theta(\tilde{X})$. We refer to this solution as naive approach since it is more related to our proposed model. Note this solution do not explore the geometry of permutation matrices and has a series of inefficiencies which will be demonstrated in our experiments.

5 Experiments

We now describe how our model can tackle different computer vision problems and measure our models performance on well established benchmarks. First, we give some details of the datasets used in our experiments. Second, in Section 5.2, we analyze how effectively our proposed model solves the permutation prediction problem under different settings. Third, in Section 5.3, we evaluate our model on the relative attributes task. Fourth, in Section 5.4, we evaluate our model for long sequences using image ranking applications. Last, in Section 5.5, we evaluate our method for self-supervised representation learning.

5.1 Datasets

We evaluate our proposed model in different computer vision applications using the following datasets (see Figure 5):

Public Figures (PubFig) [57]. This dataset consists of 800 facial images of eight public celebrities annotated with eleven physical attributes, such as big lips, white, and young. This is a *relative attribute dataset* with category level annotation, i.e., all images in a specific category may be ranked higher, equal, or lower than all images in another category, with respect to an attribute. Our goal is to rank subsets of images according to these visual attributes.

Outdoor Scene Recognition (OSR) [57]. This is another *relative attribute dataset* with category level annotation. It consists of 2688 images of eight different types of outdoor scenes such as Mountain, Forest, and Coast, annotated with six different visual attributes such as natural and open. This dataset has more ties between pair of images than PubFig, which may impose some difficulties to our model as discussed in later sections.

Historical Car (CarDb) [44]. This dataset consists of 12k images of cars annotated with manufacturing information such as model and manufacturing year. In this work, we are interested in ranking the cars according to their manufacturing date. Different from the PubFig and OSR datasets, CarDb has instance-level annotations, i.e., each image may be ranked higher, equal or lower than other image. This is a harder problem, since fine-grained comparisons have to be made in order to correctly rank the images.

Interestingness Annotations. This dataset comes from an investigation of human interest in photos by Gygli et al. [30]. Using psychophysical experiments on Mechanical Turk, they annotate the images from OSR dataset with an interestingness score which measures the degree of interestingness of an image. Our goal is to rank images according to how interesting they are. Similar to CarDb, this dataset is instance-level annotated and we use the OSR train/test splits in our experiments.

ImageNet [39]. This is a large scale dataset for object recognition. It consists of approximately 1.3M images of 1k different object categories. In our experiments, we use the training set images of this dataset discarding the labels to learn image representations in a self-supervised fashion.

Pascal VOC [18, 17]. This is a fine-grained object recognition dataset. It has 9,963 images containing 24,640 annotated objects of 20 different classes. This dataset provides image, bounding boxes and pixel level annotations and it is widely used in the literature. In this work, we evaluate our self-supervised image representations in this dataset for object classification, detection and segmentation.

5.2 Permutation Prediction

In this experiment, we evaluate our proposed method on the permutation prediction task and compare with a naïve approach which combines sigmoid outputs and cross-entropy loss by casting the permutation prediction problem as a multi-label classification problem. In this experiment, we use the Public Figures dataset [57] and its default train and test splits.

In our implementation, we use stochastic gradient descent with mini-batches of 32 image sequences, images of 256×256 pixels and different sequence lengths. During preprocessing, we subtract the mean and randomly crop each image to size 227×227. We initialize our network from *conv1* to *fc6* layers using an AlexNet model pre-trained on the ILSVRC 2012 [39] dataset for the task of image classification, while other layers are randomly initialized from a Gaussian distribution. We set the learning rate to 10^{-5} and fine-tune our model for permutation prediction over 25k iterations using the multi-class cross entropy loss. With the exception of the self-supervised representation learning experiments in Section 5.5, these hyper-parameters and implementation details are used throughout all of our experiments.

As performance metrics for the permutation prediction task, we use Kendall-tau and Hamming similarity. Kendall tau is defined

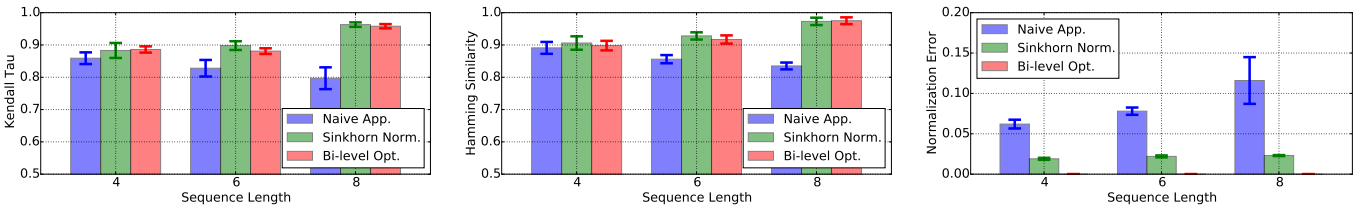

Fig. 6. Evaluating and comparing naive approach, Sinkhorn normalization and bi-level optimization variants of the proposed model on the permutation prediction task using the Public Figures Dataset [57]. The models are trained and tested for each attribute separately. We report the mean and standard deviation of the the performance metrics (Kendall Tau, Hamming similarity, and normalization error) across the attributes.

as $KT = \frac{c^+ - c^-}{0.5l(l-1)}$, where c^+ and c^- denote the number of all pairs in the sequence that are correctly and incorrectly ordered, respectively. It captures how close we are to the perfect rank. The Hamming similarity measures the number of equal entries in two vectors or matrices normalized by the total number of elements. It indicates how similar our prediction is to the ground truth permutation matrix. In addition, we measure the averaged ℓ_1 normalization error of rows and columns of the predicted doubly stochastic matrices.

We train a CNN model for each attribute in the Public Figures dataset by sampling 30K ordered image sequences from the training images. We then evaluate the trained models on 20K image sequences generated from the test set by sampling correctly ordered sequences and randomly permuting them. We averaged the results over the 11 attributes and repeat the experiment for image sequences of length 4 , 6 and 8. Figure 6 presents the results for our proposed methods and the naïve approach.

We observe the naïve approach works well for small sequences and is able to learn the normalization by itself. As the sequence length increases, however, the performance of the naïve approach degenerates and the ℓ_1 normalization error increases. On the other hand, the Sinkhorn Normalization and Bi-level optimization approaches reach better results in both *Kendall-Tau* and Hamming similarity while keeping the normalization error almost unchangeable even for longer sequences. This fact suggests that exploring the geometrical structure of the space of doubly-stochastic matrices (and thereby the permutation matrices) is useful.

It is worth noting that we could train our model for all attributes jointly by sharing the convolution layers and adding as many fully connected layers as the number of attributes. Such an approach is well known in multi-task CNNs [1] and usually provides more generalizable models. However, this approach requires more memory resources which would slow down our experiments.

5.3 Relative Attributes

In this experiment, we use DeepPermNet to compare images in a given sequence according to a certain attribute by predicting permutations and applying their inverse. This procedure can be used to solve the relative attributes task, the goal of which is to compare pairs or sets of images according to the "strength" of a given attribute. In this context, we compare our proposed approach to state-of-the-art methods for relative attributes.

For this application, we use the OSR scene dataset [57], the Public Figures Dataset [57], and the implementation details and hyper-parameters described in the previous section. We train our model for each attribute with 30k ordered image sequences of length 8 generated from the training set. Then, we report our models performance in terms of pairwise accuracy measured on the predicted ordering for 20k image sequences of length 8 generated from the test set using stratified sampling.

Different from the existing methods [69, 67] which also use deep features and pre-trained models, we directly predict the order of sequences of images instead of pairs. Our scheme allows us to make use of the structure in the sequences as a whole, which is more informative than pairs providing better performance. For a fair comparison to prior methods, we measure our performance by computing the pairwise accuracy for all pairs in each sequence. Tables 1 and 2 present our results.

On the Public Figures dataset, DeepPermNet outperforms the state-of-the-art models by a margin of 3% in pairwise accuracy. It is a substantial margin, consistently observed across all attributes. Note that, we outperform the recent method Souri et al. [69], which uses a pre-trained VGG16 model that has significantly more modeling capacity than the AlexNet [39] architecture that we use. On the other hand, our method works slightly worse than [69] on the OSR dataset. We also provide results by building our scheme on a VGG16 model. As is clear, using this variant, we demonstrate even better results outperforming the state-of-the-art methods. In addition, the bi-level variant of our model, despite providing optimal solution to the doubly stochastic approximation, works on par with the Sinkhorn layer, which shows that the Sinkhorn operator is a sufficient approximation for our problem.

It is worth noting that DeepPermNet works better when we use longer sequences for training, because they provide rich information that can be directly used in our method. For instance, the performance of DeepPermNet drops 7% in terms of average pairwise accuracy on the Public Figures dataset when we train our model using just pairs. In addition, the proposed model is not able to explicitly handle equality cases, since the permutation learning formulation assumes each permutation is unique, which is not true in the relative attributes task. Perhaps, this is the reason for the difference in performance between the Public Figures and OSR datasets. Nonetheless, DeepPermNet is able to learn very good attribute rankers from data as shown in our experiments.

We also compute the saliency maps of different attributes using the method proposed by Simonyan et al. [66]. More specifically, we take the derivative of the estimated permutation matrix w.r.t. the input, given a set of images. We perform max pooling across channels to generate the saliency maps. Figure 7 presents qualitative results and saliency maps generated by DeepPermNet for different attributes.

This map is a simplified way to visualize which pixels, regions, and features of a given image are more relevant to the respective permutation predicted by our method. For instance, the attribute "bushy eyebrows" is sensitive to the region of eyes, while the attribute "smiling" is more sensitive to the mouth region. An interesting observation is the possibility of localizing such features without any explicit supervision (e.g., bounding boxes), which could be used for unsupervised attribute localization.

TABLE 1
Evaluating the proposed model on the Public Figures Dataset. We report the pairwise accuracy as well as its mean across the attributes.

Method	Lips	Eyebrows	Chubby	Male	Eyes	Nose	Face	Smiling	Forehead	White	Young	Mean
Parikh and Grauman [57]	79.17	79.87	76.27	81.80	81.67	77.40	82.33	79.90	87.60	76.97	83.20	80.56
Li et al. [45]	81.87	81.84	79.97	85.33	83.15	80.43	86.31	83.36	88.83	82.59	84.41	83.37
Yu and Grauman [80]	90.43	89.83	87.37	91.77	91.40	89.07	86.70	87.00	94.00	87.43	91.87	89.72
Souri et al. [69]	93.62	94.53	92.32	95.50	93.19	94.24	94.76	95.36	97.28	94.60	94.33	94.52
DeepPermNet (Sinkhorn Norm.)	**99.55**	**97.21**	97.66	**99.44**	96.54	**96.21**	99.11	97.88	**99.00**	**97.99**	**99.00**	**98.14**
DeepPermNet (Bi-level Opt.)	99.53	96.65	**98.54**	98.99	**97.21**	94.72	**99.44**	**98.55**	98.77	95.66	98.77	97.89

TABLE 2
Evaluating the proposed model on the OSR dataset. We report the pairwise accuracy as well as its mean across the attributes.

Method	Depth-Close	Diagonal-Plane	Natural	Open	Perspective	Size-Large	Mean
Parikh and Grauman [57]	87.53	86.5	95.03	90.77	86.73	86.23	88.80
Li et al. [45]	89.54	89.34	95.24	92.39	87.58	88.34	90.41
Yu and Grauman [80]	90.47	92.43	95.7	94.1	90.43	91.1	92.37
Singh and Lee [67]	96.1	97.64	98.89	97.2	96.31	95.98	97.02
Souri et al. [69]	97.65	98.43	**99.4**	97.44	96.88	96.79	97.77
DeepPermNet (Sinkhorn Norm.)	96.09	94.53	97.21	96.65	96.46	98.77	96.62
DeepPermNet (Bi-level Opt.)	97.99	98.21	97.76	97.10	97.21	96.65	97.49
DeepPermNet (Sinkhorn Norm. + VGG16)	96.87	97.99	96.87	**99.79**	**99.82**	**99.55**	**98.48**
DeepPermNet (Bi-level Opt. + VGG16)	**98.12**	**99.92**	98.13	97.78	98.72	97.87	98.42

5.4 Supervised Learning to Rank

A sequence of length l is correctly ordered if and only if all of its subsequences of length l are correctly ordered. Therefore, we can use our method to order subsequences and a classic sorting algorithm like bubble sort or merge sort to rank image sequences of arbitrary length according to a predefined criterion. Then, this pipeline can be used for supervised learning to rank applications.

We select two supervised image ranking applications to compare our method with other supervised learning-to-rank algorithms, namely, ranking images based on interestingness and ordering car images by manufacturing date. For the former, we use the annotations provided by [30] which assign an interestingness score for images of the OSR dataset. For the latter, we use the car dataset [44] which is composed by images of cars manufactured from 1920 to 1999. As implementation details, we use the same model hyper-parameters described in Section 5.2.

In this experiment, we train our model by sampling 30k sequences of length 4 from the training images, and use our learned model and the basic bubble sorting algorithm to rank 20k sequences of length 20 sampled from the test images. The final rank obtained is evaluated with rank metrics like NDCG (Normalized Discounted Cumulative Gain), Kendall-tau and Pairwise accuracy. Table 3 presents the results.

TABLE 3
Evaluating the proposed model on ranking scenes according how interesting they look and ranking cars according to their manufacturing date. We report normalized discounted cumulative gain (NDCG), Kendall Tau (KT), and pairwise accuracy.

	Scene Interestigness			Car Chronology		
Method	NDCG	KT	Pair. Acc.	NDCG	KT	Pair. Acc.
Joachims [35]	0.870	0.317	65.8	0.928	0.482	74.1
Xu and Li [78]	0.745	-0.077	46.1	0.827	0.118	55.9
Wu et al. [76]	0.860	0.315	64.3	0.935	0.409	70.6
Cao et al. [10]	0.821	0.118	55.9	0.872	0.291	64.5
Xia et al. [77]	0.862	0.282	64.1	0.854	0.278	63.9
Fernando et al. [22]	0.887	0.347	67.4	0.949	0.553	76.9
Ours (Sinkhorn Norm.)	0.922	0.360	68.0	**0.968**	**0.724**	**86.2**
Ours (Bi-level Opt.)	**0.923**	**0.363**	**68.2**	0.964	0.700	84.9

We observe that our method improves the accuracy of the ranking consistently for all evaluation criteria. It is worth pointing that the proposed model work drastically better than other neural network models such as ListNet [10]. We argue that this improvement is caused by the image representation implicitly learned in a end-to-end fashion by our method. We again observe that the Sinkhorn normalization presents results as good as the exact solution provided by the bi-level optimization variant.

5.5 Self-Supervised Representation Learning

Yosinski et al. [79] observed that pre-training a network helps to regularize the model reaching better performance on the test set. Doersch et al. [14] and Noroozi and Favaro [53] observed that the spatial structure of objects is a strong supervisory signal to learn transferable weights. Following their work, we exploit such a self-supervisory signal and generate ordered sequences of patches to train our model and transfer the learned weights for target tasks such as object classification, detection, and segmentation.

More specifically, we use the train split of the ImageNet dataset [39] as training set discarding its labels. For each image, we split it into a grid with 3×3 cells, extract a patch of size 64×64 pixels within each grid cell and generate a sequence where the ordering is defined by the spatial position of each patch in the grid (see Figure 1 right). We then train our models to predict random permutations of these generated patch sequences as before.

It has been observed in the literature that self-supervised learning methods can exploit "shortcuts" involving information useful for solving the pre-text task but not for a target task [14, 53]. For instance, chromatic aberration and edge continuity are good cues for solving the visual permutation task, but are not useful for generic object detection or image classification. In order to avoid these "shortcuts", we follow image preprocessing procedures described by Doersch et al. [14]. We first resize the images having the smallest side equal to 256 pixels. Then, we randomly crop a squared region of the image and resize to 225×225 pixels. Then we split the resized crop into a 3×3 grid cell, each with 75×75 pixels. Finally, we randomly select 64×64 pixels tiles from each cell and train our model as described above. This allows us to have an 11 pixel gap between tiles. Noroozi and Favaro [53] show improvements in the target task by using additional procedures such as augmenting the data with gray-scale images, jittering the

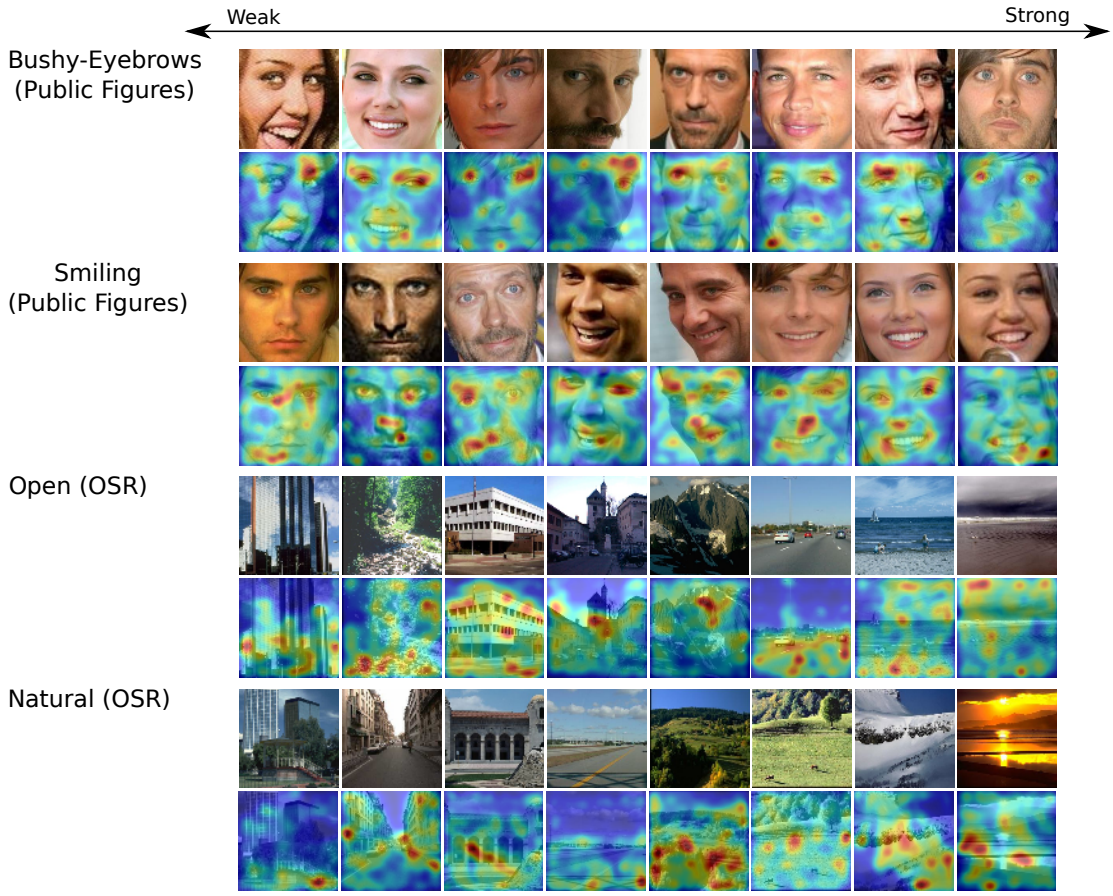

Fig. 7. Qualitative results: Samples from the Public Figures and OSR test images are ordered according to different attributes. Saliency maps: Smoothed visualization of the derivative of the estimated permutation matrix w.r.t the input images. Regions with warmer color are more relevant to the predicted permutation for the specified attribute. Better viewed in color.

color channels, and increasing the gap between sampled tiles. We did not investigate these additional procedures, but they can be easily added to our framework.

It is important to emphasize that we do not use any pre-trained models or human annotated labels in this self-supervised learning experiment. Instead, we train our CNN models from scratch using random initialization and self-supervised labels. The model is trained for 400k iterations using a initial learning rate of 0.001, which is dropped by one-tenth every 100k iterations. We use batches of 256 sequences each of 64×64 image patches.

Using a CNN to recover an image from its parts is a challenging task because it requires the network to learn semantic concepts, contextual information, and objects-parts relationships, in order to predict the right permutation. In order to evaluate how well the proposed models can solve such a task, we use 50k images on the ImageNet validation set and apply random permutations using the 3×3 grid layout. In this self-supervised setting, DeepPermNet reaches a score of 0.72 on the Kendall-tau metric.

Following the literature on self-supervised pre-training [14, 16, 58, 53, 42, 60], we test our models on the commonly used self-supervised benchmarks on the PASCAL Visual Object Challenge and compare against supervised and self-supervised procedures for pre-training. We transfer our learned weights to initialize from *Conv1* to *Conv5* layers of AlexNet [39], Fast-RCNN [25] and Fully Convolutional Network [47] models and fine-tune them for object classification, detection, and segmentation tasks respectively, using their default training parameters. For object classification and detection, we report the mean average precision (mAP) on PASCAL VOC 2007 [18], while for object segmentation, we report mean average intersection over union (mIU) on PASCAL VOC 2012 [17]. In order to make the competing methods directly comparable, we use stride 2 in the first layer of our network during the training of visual permutation learning task, while we use a standard AlexNet (stride 4 on the first layer) in the transfer learning experiments. Table 4 presents our results.

We observe that the self-supervised methods are still behind the supervised approach, but this gap reduces gradually. Our DeepPermNet works as well as most of the self-supervised competitors, while it is marginally superseded by very recent approaches. It also overcomes its direct competitor [53] in object classification and segmentation by exploiting our permutation prediction schema. In addition, DeepPermNet is a more generic method than the method proposed by Noroozi and Favaro [53], since our method can be used to solve many different computer vision tasks as shown in our previous experiments. We also notice that the bi-level approach performs slightly worse than the Sinkhorn normalization approach in this self-supervised experiment. Perhaps, the reason for that is computation of the gradient which requires inverting a matrix, and can cause numerical issues.

Interestingly, when finer grid schemes are used (e.g., 4×4), we do not observe any improvement in the target tasks. This agrees with the ablation study presented in [53], which shows that the performance in the target task increases with the total number of permutations, but decreases with the increasing of the similarity between these permutations in their jigsaw task. Therefore, we believe when we deal with all possible permutations

and increase the grid partition, we end up increasing the general similarity between the target permutations, which is prejudicial for transfer learning. Perhaps, a solution to this issue is to weight the permutations according to their average similarity to the other permutations, which is a compelling direction for future work.

TABLE 4
Classification and detection results on PASCAL VOC 2007 test set under the standard mean average precision (mAP), and segmentation results on the PASCAL VOC 2012 validation set under mean intersection over union (mIU) metric. *Noroozi and Favaro [53] and our methods use a more computationally intensive ConvNet architecture with a finer stride at conv1 during the self-supervised training, but we use standard Alex-net architecture when finetune in the target task allowing a fair comparison with all competing methods.

Pre-training Method	Cls.	Det.	Seg.
ImageNet	78.2	56.8	48.0
Random Gaussian	53.3	43.4	19.8
Agrawal et al. [3]	52.9	41.8	-
Doersch et al. [14]	55.3	46.6	-
Wang and Gupta [74]	58.4	44.0	-
Pathak et al. [58]	56.5	44.5	29.7
Donahue et al. [16]	58.9	45.7	34.9
Zhang et al. [81]	65.6	47.9	35.6
Noroozi and Favaro [53]*	67.6	53.2	37.6
Owens et al. [56]	61.3	44.0	-
Bojanowski and Joulin [5]	65.3	49.4	-
Noroozi et al. [54]	67.7	51.4	36.6
Lee et al. [43]	63.8	46.9	-
Pathak et al. [59]	61.0	52.2	-
Zhang et al. [82]	67.1	46.7	36.0
Larsson et al. [42]	65.9	-	38.0
Jenni and Favaro [34]	69.8	52.5	38.1
Gidaris et al. [24]	**72.97**	54.4	39.1
Kim et al. [36]	69.2	52.4	39.3
Nathan Mundhenk et al. [52]	69.6	**55.8**	**41.2**
Ren and Jae Lee [60]	68.0	52.6	-
DeepPermNet (Sinkhorn Norm.)*	69.4	49.5	37.9
DeepPermNet (Bi-level Opt.)*	65.5	45.7	36.4

6 Discussion and Conclusion

In this paper, we tackled the problem of learning the structure of visual data by introducing the task of visual permutation learning. We formulated an optimization problem for this task with the goal of recovering the permutation matrix responsible for generating a given randomly shuffled image sequence based on a pre-defined visual criteria. We proposed novel CNN layers that can convert standard CNN predictions to doubly-stochastic approximations of permutation matrices using Sinkhorn normalizations and bi-level optimization. Thus, the proposed CNN model can be trained in an end-to-end manner.

Through a variety of experiments, we assess the proposed method and demonstrate that permutation learning can be applied to different tasks. More specifically, we first validate the hypothesis of exploring the geometrical structure of doubly-stochastic matrices helps to learn visual permutations. As shown in Figure 6, both variants of the proposed DeepPermNet outperform the naïve approach. We then continued our evaluation for real-world applications and state-of-the-art methods such as relative attributes (Section 5.3), supervised learning to rank (Section 5.4), and self-supervised representation learning (Section 5.5). In all experiments, we present state-of-the-art results demonstrating the usefulness of the proposed permutation learning schema.

It is important to highlight the advantages and disadvantages of our two variants of the proposed approach. The bi-level optimization variant optimally solves the doubly-stochastic approximation problem, while the Sinkhorn normalization variant is an efficient and approximate solution for such a problem. However, in practice the Sinkhorn variant works slightly better than the bi-level variant in most of the cases which, perhaps, is a consequence of the quality of image representations learned as evidenced in Section 5.5. Even so, the bi-level variant is able to provide improvements in some cases, e.g, four attributes in Pubfig (Table 1), two attributes in OSR (Table 2), and Scene Interestingness (Table 3). However, it comes to the cost of solving a QP problem for every input during training and inference.

As future work, we intend to explore structured information beyond 2D images. We believe that our model is also effective for other modalities such as text, videos and 3D data. One compelling direction is to evaluate our model in other tasks such as video summarization, motion representation and view synthesis.

Acknowledgements

This research was supported by the Australian Research Council (ARC) through the Centre of Excellence for Robotic Vision (CE140100016) and was undertaken with the resources from the National Computational Infrastructure (NCI), at the Australian National University (ANU).

References

[1] A. H. Abdulnabi, G. Wang, J. Lu, and K. Jia, "Multi-task cnn model for attribute prediction," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1949–1959, 2015.

[2] R. P. Adams and R. S. Zemel, "Ranking via Sinkhorn propagation," *arXiv preprint arXiv:1106.1925*, 2011.

[3] A. Agrawal, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen, "Interactive digital photomontage," *ACM Transactions on Graphics (ToG)*, vol. 23, no. 3, pp. 294–302, 2004.

[4] G. Birkhoff, "Three observations on linear algebra," *Univ. Nac. Tucumán. Revista A*, vol. 5, pp. 147–151, 1946.

[5] P. Bojanowski and A. Joulin, "Unsupervised learning by predicting noise," in *ICML*, 2017.

[6] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[7] S. Branson, O. Beijbom, and S. Belongie, "Efficient large-scale structured learning," in *CVPR*, 2013.

[8] B. J. Brown, C. Toler-Franklin, D. Nehab, M. Burns, D. Dobkin, A. Vlachopoulos, C. Doumas, S. Rusinkiewicz, and T. Weyrich, "A system for high-volume acquisition and matching of fresco fragments: Reassembling theran wall paintings," in *ACM Transactions on Graphics (TOG)*, 2008.

[9] R. A. Brualdi, "Some applications of doubly stochastic matrices," *Linear Algebra and its Applications*, vol. 107, pp. 77 – 100, 1988.

[10] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *ICML*, 2007.

[11] T. S. Cho, S. Avidan, and W. T. Freeman, "The patch transform," *PAMI*, vol. 32, no. 8, 2010.

[12] W. S. Cooper, F. C. Gey, and D. P. Dabney, "Probabilistic retrieval based on staged logistic regression," in *Proceedings*

of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, 1992.

[13] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *JMLR*, vol. 17, no. 83, pp. 1–5, 2016.

[14] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *ICCV*, 2015.

[15] J. Domke, "Generic methods for optimization-based modeling." in *AISTATS*, 2012.

[16] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," in *ICLR*, 2017.

[17] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," 2007.

[19] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth, "Every picture tells a story: Generating sentences from images," in *ECCV*, 2010.

[20] O. Faugeras, *Three-dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.

[21] B. Fernando and S. Gould, "Learning end-to-end video classification with rank-pooling," in *ICML*, 2016.

[22] B. Fernando, E. Gavves, D. Muselet, and T. Tuytelaars, "Learning-to-rank based on subsequences," in *ICCV*, 2015.

[23] B. Fernando, H. Bilen, E. Gavves, and S. Gould, "Self-supervised video representation learning with odd-one-out networks," in *CVPR*, 2017.

[24] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *ICLR*, 2018.

[25] R. Girshick, "Fast R-CNN," in *ICCV*, 2015.

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[27] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks." in *AISTATS*, 2010.

[28] S. Gould, B. Fernando, A. Cherian, P. Anderson, R. Santa Cruz, and E. Guo, "On differentiating parameterized argmin and argmax problems with application to bi-level optimization," *arXiv preprint arXiv:1607.05447*, 2016.

[29] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016. [Online]. Available: http://www.gurobi.com

[30] M. Gygli, H. Grabner, H. Riemenschneider, F. Nater, and L. Van Gool, "The interestingness of images," in *ICCV*, 2013.

[31] C. Huang, C. Change Loy, and X. Tang, "Unsupervised learning of discriminative attributes and visual representations," in *CVPR*, 2016.

[32] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson, "Learning visual groups from co-occurrences in space and time," *ICLR Workshop*, 2016.

[33] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *NIPS*, 2015.

[34] S. Jenni and P. Favaro, "Self-supervised feature learning by learning to spot artifacts," in *CVPR*, 2018.

[35] T. Joachims, "Training linear svms in linear time," in *SIGKDD*, 2006.

[36] D. Kim, D. Cho, D. Yoo, and I. S. Kweon, "Learning image representations by completing damaged jigsaw puzzles," in *WACV*, 2018.

[37] P. A. Knight, "The sinkhorn-knopp algorithm: convergence and applications," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 1, pp. 261–275, 2008.

[38] A. Kovashka, D. Parikh, and K. Grauman, "WhittleSearch: Image Search with Relative Attribute Feedback," in *CVPR*, 2012.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[40] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001.

[41] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *PAMI*, vol. 36, no. 3, pp. 453–465, 2014.

[42] G. Larsson, M. Maire, and G. Shakhnarovich, "Colorization as a proxy task for visual understanding," in *CVPR*, 2017.

[43] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang, "Unsupervised representation learning by sorting sequences," in *ICCV*, 2017.

[44] Y. J. Lee, A. A. Efros, and M. Hebert, "Style-aware mid-level representation for discovering visual connections in space and time," in *ICCV*, 2013.

[45] S. Li, S. Shan, and X. Chen, "Relative forest for attribute prediction," in *ACCV*, 2012.

[46] L. Liang and K. Grauman, "Beyond comparing image pairs: Setwise active learning for relative attributes," in *CVPR*, 2014.

[47] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[48] W. Marande and G. Burger, "Mitochondrial dna as a genomic jigsaw puzzle," *Science*, vol. 318, no. 5849, 2007.

[49] M. Marszalek, I. Laptev, and C. Schmid, "Actions in context," in *CVPR*, 2009.

[50] I. Misra, C. L. Zitnick, and M. Hebert, "Shuffle and learn: Unsupervised learning using temporal order verification," in *ECCV*, 2016.

[51] R. Mottaghi, X. Chen, X. Liu, N. G. Cho, S. W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *CVPR*, 2014.

[52] T. Nathan Mundhenk, D. Ho, and B. Y. Chen, "Improvements to context based self-supervised learning," in *CVPR*, 2018.

[53] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *ECCV*, 2016.

[54] M. Noroozi, H. Pirsiavash, and P. Favaro, "Representation learning by learning to count," in *ICCV*, 2017.

[55] P. Ochs, R. Ranftl, T. Brox, and T. Pock, "Bilevel optimization with nonsmooth lower level problems," in *International Conference on Scale Space and Variational Methods in Computer Vision*, 2015.

[56] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba, "Ambient sound provides supervision for visual learning," in *ECCV*, 2016.

[57] D. Parikh and K. Grauman, "Relative attributes," in *CVPR*, 2011.

[58] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016.

[59] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan, "Learning features by watching objects move," in *CVPR*, 2017.

[60] Z. Ren and Y. Jae Lee, "Cross-domain self-supervised multi-task feature learning using synthetic imagery," in *CVPR*, 2018.

[61] R. Santa Cruz, B. Fernando, A. Cherian, and S. Gould, "Deeppermnet: Visual permutation learning," in *CVPR*, 2017.

[62] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *PAMI*, vol. 31, no. 5, pp. 824–840, 2009.

[63] D. Sculley, "Large scale learning to rank," in *NIPS Workshop on Advances in Ranking*, 2009.

[64] S. Shankar, V. K. Garg, and R. Cipolla, "Deep-carving: Discovering visual attributes by carving deep neural nets," in *CVPR*,

2015.

[65] D. Sholomon, O. David, and N. S. Netanyahu, "A genetic algorithm-based solver for very large jigsaw puzzles," in *CVPR*, 2013.

[66] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *ICLR Workshop*, 2014.

[67] K. K. Singh and Y. J. Lee, "End-to-end localization and ranking for relative attributes," in *ECCV*, 2016.

[68] R. Sinkhorn and P. Knopp, "Concerning nonnegative matrices and doubly stochastic matrices," *Pacific Journal of Mathematics*, vol. 21, no. 2, 1967.

[69] Y. Souri, E. Noury, and E. Adeli-Mosabbeb, "Deep relative attributes," in *ACCV*, 2016.

[70] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR*, 2011.

[71] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *ICML*, 2004.

[72] J. Von Neumann, "A certain zero-sum two-person game equivalent to the optimal assignment problem," *Contributions to the Theory of Games*, vol. 2, pp. 5–12, 1953.

[73] T. Wang, S. Gong, X. Zhu, and S. Wang, "Person re-identification by discriminative selection in video ranking," *PAMI*, vol. 38, no. 12, pp. 2501–2514, 2016.

[74] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *ICCV*, 2015.

[75] P. Wohlhart, V. Lepetit, T. Klatzer, and T. Pock, "Continuous hyper-parameter learning for support vector machines," in *Computer Vision Winter Workshop*, 2015.

[76] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao, "Adapting boosting for information retrieval measures," *Information Retrieval*, vol. 13, no. 3, pp. 254–270, 2010.

[77] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, "Listwise approach to learning to rank: theory and algorithm," in *ICML*, 2008.

[78] J. Xu and H. Li, "Adarank: a boosting algorithm for information retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.

[79] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *NIPS*, 2014.

[80] A. Yu and K. Grauman, "Fine-grained visual comparisons with local learning," in *CVPR*, 2014.

[81] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *ECCV*, 2016.

[82] R. Zhang, P. Isola, and A. A. Efros, "Split-brain autoencoders: Unsupervised learning by cross-channel prediction," in *CVPR*, 2017.

Rodrigo Santa Cruz is a Ph.D. candidate since 2015 at the Australian National University under supervision of Prof. Stephen Gould. He is also associated with the Australian Center of Robotic Vision (ACRV). He finished B.Sc. in Computer Engineering at University of Pernambuco, Brazil. His research interests are in the areas of computer vision, machine learning and optimization.

Basura Fernando is a research scientist at the Agency for Science, Technology and Research (ASTAR) Singapore. He is also a honorary lecturer at the Australian National University (ANU). Prior to that he was a research fellow at the Australian Centre for Robotic Vision (ACRV), the Australian National University (ANU). He was the project leader of "SR1: video representations" of the Australian Centre for Robotic Vision. He obtained PhD from VISICS group of KU Leuven, Belgium in March 2015. He is interested in Computer Vision and Machine Learning research. He has contributed to statistical visual domain adaptation and human action recognition.

Anoop Cherian is a Research Scientist at Mitsubishi Electric Research Labs (MERL) Cambridge, MA and an Adjunct Researcher affiliated to the Australian Centre for Robotic Vision (ACRV) at the Australian National University. Previously, he was a Postdoctoral Researcher in the LEAR team at INRIA at Grenoble. He received his B.Tech (honors) degree in computer science and Engineering from the National Institute of Technology, Calicut, India in 2002, his M.S. and Ph.D. degrees in computer science from the University of Minnesota, Minneapolis in 2010 and 2013 respectively. From 2002–2007, he worked as a software design engineer at Microsoft. His research interests lie in the areas of computer vision and machine learning.

Stephen Gould is an Associate Professor in the Research School of Computer Science in the College of Engineering and Computer Science at the Australian National University. He received his BSc degree in mathematics and computer science and BE degree in electrical engineering from the University of Sydney in 1994 and 1996, respectively. He received his MS degree in electrical engineering from Stanford University in 1998. He earned his PhD degree from Stanford University in 2010. His research interests are in computer and robotic vision, machine learning, probabilistic graphical models, and optimization.