Unsupervised Rotation Factorization in Restricted Boltzmann Machines

Mario Valerio Giuffrida, and Sotirios A. Tsaftaris, *Senior Member, IEEE*

Finding suitable image representations for the task at hand is critical in computer vision. Different approaches extending the original Restricted Boltzmann Machine (RBM) model have recently been proposed to offer rotation-invariant feature learning. In this paper, we present an extended novel RBM that learns rotation invariant features by explicitly factorizing for rotation nuisance in 2D image inputs within an unsupervised framework. While the goal is to learn invariant features, our model infers an orientation per input image during training, using information related to the reconstruction error. The training process is regularised by a Kullback-Leibler divergence, offering stability and consistency. We used the γ-score, a measure that calculates the amount of invariance, to mathematically and experimentally demonstrate that our approach indeed learns rotation invariant features. We show that our method outperforms the current state-of-the-art RBM approaches for rotation invariant feature learning on three different benchmark datasets, by measuring the performance with the test accuracy of an SVM classifier. Our implementation is available at `https://bitbucket.org/tuttoweb/rotinvrbm`.

***Index Terms*—machine learning, neural networks, rotation-invariant features, Restricted Boltzmann Machines.**

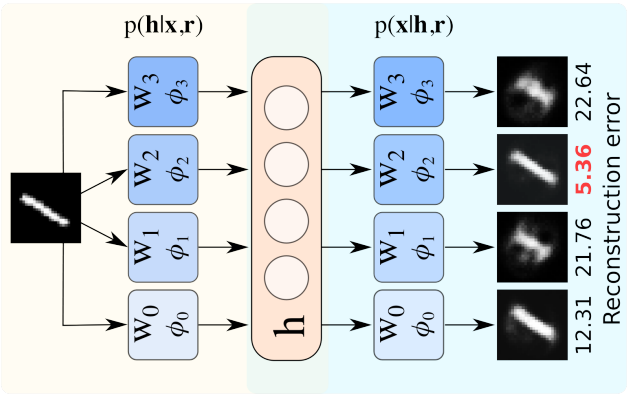

Fig. 1. Representation of the proposed rotation-invariant RBM. In this example, we have $S = 4$ rotations, corresponding to the equidistant angles $\Phi = \{\phi_0 = 0°, \phi_1 = 90°, \phi_2 = 180°, \phi_3 = 270°\}$. Each angle is associated with a matrix W_s. When an image is provided to the network, the weight matrix minimizing the reconstruction error is chosen, as highlighted in bold red. We depict the unfolded steps of the CD-1 [13]. (Best viewed in color).

I. Introduction

The unsupervised learning of image representations is an important computer vision task, allowing to learn suitable features from unlabeled data. Several algorithms have been proposed [1], such as kernel PCA [2], autoencoders [3], and Restricted Boltzmann Machines [4]. When features are learned in an unsupervised manner, it is typically unclear what can be considered a "good" representation [5]. However, it is widely acknowledged that invariant features possess good properties, as they disentangle irrelevant variation from the dataset [6].

There are two potential approaches to learning invariant representations. On one side, we hope that an algorithm will learn to factor out invariance *implicitly* [7], assuming the presence of sufficient training data. On the other side, invariance can be learned *explicitly*, where a feature extractor Φ has some prior knowledge of invariance in the form $\Phi(x) = \Phi(Tx)$ [8], [9]. In this case, the transformation T is modeled during training and the algorithm learns to factorize it out.

It comes as no surprise that explicit encoding of prior knowledge should help, instead of trying to learn it (and factor it out) implicitly. Findings in human understanding do actually point to this direction, noting the distinct benefits of explicit learning (such explicability of actions and interpretation) [10], [11]. Thus, in the context of learning from few data, one must openly consider that shallow neural network architectures that are specifically designed to be invariant to some form of nuisance can be beneficial.

We endow a shallow unsupervised generative model to be invariant to rotations, which is an ubiquitous transformation in several computer vision problems [12]. Specifically, we propose an extension of Restricted Boltzmann Machines (RBMs) [4]. In their original formulation, RBMs are a shallow neural network characterized by a bipartite graph, whose sides are called *visible* (denoted as $\mathbf{x}$) and *hidden* (denoted as $\mathbf{h}$) layers respectively. In its original formulation, RBMs do not accommodate for geometric transformations occurring in an image. The most straightforward way to learn variability in a dataset is to provide the network with a sufficient amount of data. However, training sets may lack variability, resulting in models with poor generalization capabilities. To cope with this, other approaches to regularize the learning process are considered, such as dataset augmentation. The drawback of this approach is that it does not explicitly enforce the network to learn transformation-invariant features. Therefore, our aim is to build a model that is capable of learning features invariant to rotations, which is one of the most ubiquitous geometric transformations in images, extending the original RBM model.

In this paper, we present RBMs that explicitly factorize rotations in 2D images in an unsupervised manner. Our architecture, represented in fig. 1, uses a weight matrix per

Paper submitted on 21st September 2018.

M.V. Giuffrida is at the Edinburgh Napier University, School of Computing, 10 Colinton Road, EH10 5DT (v.giuffrida@napier.ac.uk).

S. A. Tsaftaris is with the Institute for Digital Communications, School of Engineering, University of Edinburgh Thomas Bayes Road, EH9 3FG, Edinburgh, UK, and with The Alan Turing Institute, 96 Euston Road, NW1 2DB, London, UK (s.tsaftaris@ed.ac.uk).

each dominant orientation of the input images. During training, an input image is passed through all weight matrices. To determine the orientation of the input image, the reconstruction error (per orientation) is computed. The weight matrix that best reconstructs the input is chosen, and then the gradient for that matrix is computed to update the parameters. Furthermore, the contribution of each input is also shared with the other weight matrices, by applying proper rotations to the gradient [14].[1] This step of sharing the gradients is essential to achieve rotation-invariance. The training process is regularized with a KL-Divergence term that enforces a prior distribution of the rotations. We measure the performance of our method by training an SVM classifier [14]–[16].

The contributions of this paper are multi-fold:

i. *rotation invariant features:* we mathematically prove, using the γ-score [17], that our model learns rotation invariant features. This is further shown with quantitative experimental results on several benchmark datasets. The feature space that our method learns is compact (limited number of hidden units are required);
ii. *robust dominant orientation inference:* we show that our model can infer robust dominant orientations, rather than relying on exogenous inputs. During training, the use of a regularization term maintains the balance among the predictions of the dominant orientations. Our inference method obtains similar test-time performance with respect to supervised methods;
iii. *no augmentation:* we show that our approach outperforms baseline and state-of-the-art methods when trained with limited data, without requiring augmentation.

The rest of this paper is structured as follows. Section II discusses related works. In Section III, we offer the theoretical background of the original RBM. Then, Section IV discusses our proposed approach, showing mathematical proofs of its validity. Section V describes the algorithm to infer the dominant rotation of an input image, offering a stability analysis that shows the robustness of our approach. In Section VI we present experimental results on three different datasets, including ablation experiments demonstrating how our approach benefits from the gradient sharing method [14] and the KL-Divergence based regularization term. We also experimentally show how consistent our unsupervised dominant orientation inference method is, including a comparison with supervised methods. Finally, Section VII concludes the manuscript.

II. Related Works

Several approaches have been proposed in the last years to improve the original RBM model to accommodate for geometric image transformations. In [18], the authors showed that Deep Belief Networks (DBNs), obtained by stacking several RBMs, produce high-level features that carry a certain amount of invariance, as the number of layers increases. In [19], they proposed a different way to train DBNs, such that filters were transformed at the end of each training epoch to account for geometric transformations. A more sophisticated DBN model that learns rotation equivariant features is STEER-DBN [20], where the authors aim to learn steerable filters [21]–[24] to achieve translation and rotation equivariance. Filters are defined as *steerable* when they can be expressed as a linear combination of (directional) basis filters [25]. In [16], the authors proposed an RBM model that can learn transformation-invariant features (TI-RBM), using a set of transformations $\mathcal{S}$. The transformations are incorporated during training and the actual image representation is then obtained using max-pooling. In [26], they proposed a block RBM, where invariance is achieved by pre-aligning the input patches according to their dominant orientation and scale, computed using SIFT descriptor [27]. In [28], the authors proposed a convolutional RBM that learns rotation equivariant features [9].

Extensive efforts have also been made to explicitly learn transformation invariant features using deep networks. The Convolutional RBM (C-RBM) [29] learns shift invariant features, extending the original RBM to accommodate the convolution. The *Spatial Transformer Network* aligns input images in a common reference space, allowing end-to-end training [30]. In [31], the authors introduce *TI-POOLING*, a siamese network that extracts features from transformed versions of the inputs using convolutional layers. Then, the output of each of the sub-networks is merged using a max-pool operation. In [9], the authors proposed the *RotEqNet*, a convolutional neural network that accounts for rotation equivariant features. A similar approach producing rotation invariant features was proposed in [32]. Recently, harmonic networks have been proposed to learn deep translation and rotation equivariant features [33]. They replace the convolutional filters with circular harmonics, then max-pooling is applied to obtain the orientation at each location of the receptive field.

Albeit the important contributions that deep learning brings to the computer vision community, often optimizing such networks involves learning of millions of parameters [34], which are not suitable for all datasets and tasks. As an example, in [35], [36] the authors adapt the network architecture to perform image segmentation for each of the datasets they tested. Furthermore, deep networks have also a vast number of hyper-parameters that need to be tuned, such as the number and size of the filters, or the depth (in terms of layers) of the network. The main drawbacks of deep neural networks are essentially two: i) prone to overfitting, especially for reduced training set; and ii) computationally expensive [37].

III. Mathematical Background

A. Adopted Notation and Conventions

In this paper, we will adopt the following notation. Matrices are written in bold and capital letters (e.g. $\mathbf{W}$), while vectors are written in bold and lower letters (e.g. $\mathbf{x}$). Vectors are always of size $n \times 1$ (column-wise vectors). Scalars are written with italic and lower letters, using both Latin and Greek alphabet (e.g., a or λ), and constant are written with capital italic Latin letters (e.g., H). Vector elements (e.g. x_k) are

[1]This manuscript is an extension of [14] but is different in: (i) how the estimate of the dominant orientation is performed (here done via reconstruction rather than relying on exogenous processes); (ii) that we further regularize the learning process with a KL-Divergence term; (iii) that we provide mathematical proofs of the invariance achieved by our model; and (iv) that we included extensive tests in more datasets and several ablations.

considered as scalars. The notation W_{kj} refers to the item located at the k-th row and j-th column in the matrix $\mathbf{W}$. Then, we will use $W_{k\cdot}$ and $W_{\cdot j}$ to refer to the k-th column and j-th row respectively. Capital Greek letters or calligraphic capital Latin letters are typically used for sets (e.g., $\mathcal{X}$ or Φ). The corresponding small letter generally denotes an item in such a set (e.g., $x \in \mathcal{X}$ or $\phi \in \Phi$). Lastly, the notation $\mathcal{X}_s$ denotes a (finite) partition of a set $\mathcal{X}$.

B. Restricted Boltzmann Machine

An RBM is a probabilistic shallow neural network that maximizes the following joint probability density function:

$$p(\mathbf{x}, \mathbf{h}) = \frac{e^{-E(\mathbf{x},\mathbf{h})}}{Z}, \tag{1}$$

where E is an energy function, taking as input $\mathbf{x} \in \{0,1\}^V$ and the hidden layer $\mathbf{h} \in \{0,1\}^H$, and Z is the partition function that ensures that eq. (1) is a probability density function (the integral over all possible values of $\mathbf{x}$ and $\mathbf{h}$ is 1). The energy term E is defined as follows:

$$E(\mathbf{x}, \mathbf{h}) = -\mathbf{x}^T\mathbf{W}\mathbf{h} - \mathbf{x}^T\mathbf{c} - \mathbf{h}^T\mathbf{b} \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{V\times H}$ is a weight matrix, $\mathbf{c} \in \mathbb{R}^V$ and $\mathbf{b} \in \mathbb{R}^H$ are the bias vectors for the input and hidden layer respectively.

The network parameters to be optimized are $\Theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$. To achieve this, the negative log-likelihood $-log(p(\mathbf{x}|\Theta))$ is minimized, using the *Contrastive Divergence* algorithm [38]. The update rules for the parameters are [39]:

$$\nabla\mathbf{W} = \mathbf{x}h(\mathbf{x})^T - \tilde{\mathbf{x}}h(\tilde{\mathbf{x}})^T, \tag{3}$$

$$\nabla\mathbf{c} = \mathbf{x} - \tilde{\mathbf{x}}, \tag{4}$$

$$\nabla\mathbf{b} = h(\mathbf{x}) - h(\tilde{\mathbf{x}}). \tag{5}$$

The function $h(\cdot)$ in eqs. (3) and (5) computes the conditional probability of $\mathbf{h}$ given $\mathbf{x}$ (this probability is explained in the next paragraph). The first term in the update rules in eqs. (3) to (5) is usually referred to as the *positive phase*, whereas the second term is called the *negative phase* [39]. The positive phase is computationally easy to calculate, whereas the negative phase is generally intractable, due to the partition function Z that appears in the computation of the partial derivatives. To overcome this problem, the negative phase is approximated with Gibbs sampling, which produces the reconstructed input $\tilde{\mathbf{x}}$.

Gibbs sampling performs alternate inference and sampling of $\mathbf{h}$ and $\mathbf{x}$, using the following conditional probabilities

$$p(h_j|\mathbf{x}) = \sigma\left(\mathbf{x}^T W_{j\cdot} + b_j\right), \tag{6}$$

$$p(x_k|\mathbf{h}) = \sigma\left(W_{\cdot k}\mathbf{h} + c_k\right), \tag{7}$$

where the function $\sigma(\cdot)$ is the logistic activation function, $W_{j\cdot}$ is j-th column of $\mathbf{W}$, $W_{\cdot k}$ is the k-th row of $\mathbf{W}$, h_j is the j-th element in the hidden layer, and x_k is the k-th element in $\mathbf{x}$. The conditional probability in (6) is the one used to compute the function $h(\cdot)$ in eqs. (3) and (5), thus $h(\mathbf{x}) \equiv p(\mathbf{h}|\mathbf{x})$. This formulation of RBM is typically referred in literature as *Bernoulli-Bernoulli RBM* (BB-RBM), as eqs. (6) and (7) define the success probability of a Bernoulli distribution.

In general, the update rule for a generic parameter $\theta \in \Theta$ is applied as follows:

$$\theta^{(t)} = \theta^{(t-1)} + \eta\nabla\theta^{(t-1)}, \tag{8}$$

where η is the learning rate, and the superscript t refers to the current iteration number.

IV. Proposed Rotation Invariant Factorization

In our formulation of rotation-invariant RBM, we assume a set $\mathcal{S} = \{R_0, R_1, \ldots, R_{S-1}\}$ of $S = |\mathcal{S}|$ equidistant rotation matrices with $R_i \in \mathbb{R}^{V\times V}$ representing an in-plane rotation by $\phi_i = i\frac{2\pi}{S}$, for all $\phi_i \in \Phi$. Details of how the rotation matrices R_i are generated are in the Supplemental Material Sec I.

Given that angles have a periodicity of 2π radians, it can be proven that $\phi_t \in \Phi$, $|t| \geq S$. As an example, assuming $S = 4$, then ϕ_5 = ϕ_1 (the argument holds also for negative indices). In general, $\phi_i = \phi_{m(i,j)}$, where $m(i,j)$ is the *modulo function*, which allows for cyclical indexing. The modulo function is defined as $m(i,j) = (i+j) \bmod S$, where $i, j \in \{0, \pm1, \pm2, \ldots, \pm(S-1)\}$. The definition of the modulo function is useful to support the proof of the theorem showing our approach learns rotation invariant features.

A. Revised Energy Function

Differently from the original RBM formulation [4], we associate a specific weight matrix to each of the rotations in $\mathcal{S}$. Therefore, $\mathbf{W}$ can be seen as a third-order tensor of dimension $V \times H \times S$. A revised version of eq. (2), taking into account rotations, takes the form of:;

$$E(\mathbf{x}, \mathbf{h}, \mathbf{r}) = \sum_{s=0}^{S-1}\sum_{j=0}^{H-1}\sum_{k=0}^{V-1} r_s\left(-x_k h_j w_{jks} - b_j h_j - c_k x_k\right). \tag{9}$$

In this formulation, a new binary vector $\mathbf{r} \in \{0,1\}^S$ is introduced, with one non-zero entry (i.e., only one orientation is active at one time). This constraint can be formalized as:

$$\sum_{n=0}^{S-1} r_n = 1. \tag{10}$$

In addition, we will say that, if $r_s = 1$, then the dominant orientation is ϕ_s. (How $\mathbf{r}$ is inferred is discussed in section V). Consequently, the conditional probabilities in eqs. (6) and (7) are revised accordingly:

$$p(h_j = 1|\mathbf{x}, \mathbf{r}) = \sigma\left(\sum_{s=0}^{S-1} r_s\left(\mathbf{x}^T W_{j\cdot s} + b_j\right)\right), \tag{11}$$

$$p(x_k = 1|\mathbf{h}, \mathbf{r}) = \sigma\left(\sum_{s=0}^{S-1} r_s\left(W_{\cdot ks}\mathbf{h} + c_k\right)\right). \tag{12}$$

B. Sharing The Gradients

Optimizing the third-order tensor $\mathbf{W}$ via eqs. (11) and (12) has the drawback that inputs with a specific dominant orientation will contribute to update *only* the corresponding slice in $\mathbf{W}$. This is equivalent to splitting the training set $\mathcal{X}$ into several non-overlapping partitions $\mathcal{X}_s$ and train a separated RBM for each of them. This will negatively affect the learned features: each "individual" RBM is trained on a rather limited dataset [14]. To overcome this problem, the contribution of gradient $\nabla\mathbf{W_s}$ computed on $\mathcal{X}_s$ can be shared across the other slices in $\mathbf{W}$. Therefore, we will apply the gradient sharing step during training as proposed in [14], which is an essential step of our training procedure to guarantee rotation-invariance.

For sake of clarity, let us consider an example with only two rotations, $\mathbf{R_0}$ and $\mathbf{R_1}$, which account for the $0°$ and $180°$ rotations respectively. Since $\nabla\mathbf{W_0}$ and $\nabla\mathbf{W_1}$ were computed on different portions of the data, namely $\mathcal{X}_0$ and $\mathcal{X}_1$, we want to transfer the contribution of $\nabla\mathbf{W_1}$ to $\nabla\mathbf{W_0}$ (and vice versa). To do so, we add a rotated version of $\nabla\mathbf{W_1}$ by $-180°$ (we denote such a rotation as R_{-1}) to $\nabla\mathbf{W_0}$. In this example, we can define the new gradients used to update the parameters of the network as $\nabla\overset{\circ}{\mathbf{W}}_{\mathbf{0}} = R_0\left(\nabla\mathbf{W_0}\right) + R_{-1}\left(\nabla\mathbf{W_1}\right)$ (an expression $\nabla\overset{\circ}{\mathbf{W}}_{\mathbf{1}}$ can be defined similarly). Using this example, the update rules for $\mathbf{W}$ can be generalized as follows:

$$\nabla\overset{\circ}{\mathbf{W}}_{\mathbf{s}} = \sum_{q=0}^{S-1} R_{-q}\left(\nabla\mathbf{W}_{m(s,q)}\right). \tag{13}$$

We use the gradients computed in eq. (13) to update the parameters of the network $\theta \in \Theta$ using eq. (8).

C. Rotational Equivalence

Following eq. (13), using the periodicity of rotations as discussed in Section IV, the following holds:

$$\begin{aligned} R_{-1}(\nabla\overset{\circ}{\mathbf{W}}_{\mathbf{0}}) &= R_{-1}(R_0(\nabla\mathbf{W_0})) + R_{-1}(R_{-1}(\nabla\mathbf{W_1})) \\ &= R_{m(-1,0)}(\nabla\mathbf{W_0}) + R_{m(-1,-1)}(\nabla\mathbf{W_1}). \end{aligned}$$

Given that $m(-1,0) = 1$ and $m(-1,-1) = 0$, the above relation becomes:

$$\begin{aligned} R_{-1}(\nabla\overset{\circ}{\mathbf{W}}_0) &= R_1(\nabla\mathbf{W_0}) + R_0(\nabla\mathbf{W_1}) \\ &= R_{-1}(\nabla\mathbf{W_0}) + R_0(\nabla\mathbf{W_1}) = \nabla\overset{\circ}{\mathbf{W}}_1, \end{aligned}$$

where $R_{-1}(\cdot) = R_1(\cdot)$ due to the rotational periodicity (e.g., rotating by $\pm180°$ produces the same result).

This example with $S = 2$ shows that all gradients of the form $\nabla\overset{\circ}{W}_s$ are rotated versions of each other. We can generalize this property for eq. (13) as follows:

$$R_r(\nabla\overset{\circ}{\mathbf{W}}_s) = \sum_{q=0}^{S-1} R_{m(r,-q)}\left(\nabla\mathbf{W}_{m(s,q)}\right) = \nabla\overset{\circ}{\mathbf{W}}_{m(s,r)}. \tag{14}$$

In order to facilitate the proof of the theorem stating that our approach learns rotation-invariant features, we need the following lemma that makes use of this rotational equivalence.

Lemma 1. *Optimizing the tensor $\mathbf{W} \in \mathbb{R}^{V\times H\times S}$ as described above for $t > 0$ iterations, then $\mathbf{W}_{s'}^{(t)} = R_\kappa(\mathbf{W}_s^{(t)})$, with:*

$$\kappa = s' - s. \tag{15}$$

Proof is provided in Appendix A. This lemma states that all the slices in the tensor $\mathbf{W}$ are rotated versions of each other.

D. Measuring the Invariance

We adopt the γ-score proposed in [17] to measure invariance. Considering a set of transformations $\mathcal{S}$ and a dataset $\mathcal{X}$, the mean activation of the j-th hidden unit h_j over all the transformations $T \in S$ is computed as:

$$\mu_j(\mathbf{x}) = \frac{1}{S}\sum_{T\in\mathcal{S}} h_j(T(\mathbf{x})), \tag{16}$$

where $h_j(x) \equiv p(h_j = 1|\mathbf{x},\mathbf{r})$. It is important to note that $\mathbf{r}$ is a function of $\mathbf{x}$, hence when the transformation $T(\mathbf{x})$ is applied, the vector $\mathbf{r}$ has to be recomputed accordingly. Then, the γ-score is defined as:

$$\gamma_j = \frac{\text{var}\left\{\mu_j(\mathbf{x})\right\}_{\mathbf{x}\in\mathcal{X}}}{\text{var}\left\{h_j(\mathbf{x})\right\}_{\mathbf{x}\in\mathcal{X}}}. \tag{17}$$

We employed the γ-score because it is bounded to the interval $[0,1]$, where values close to 1 indicate features invariant to the set of transformations $\mathcal{S}$. The γ-score is closely related to the auto-correlation [17] and does not require extra parameters to be computed, as e.g. the firing threshold in [18]. Although false positives (e.g., the ratio in eq. (17) is '1' but the full invariance is not achieved) might occur (as reported in [17]), we will show in the next section that such cases do not arise in our rotation-invariant RBM formulation.

E. Proving Rotation Invariance of the Proposed Method

In this section, we will prove that our model can learn rotation invariant features on the basis of the γ-score (c.f. section IV-D). Our theorem is based on the hypothesis that the model is trained using the revised energy model and further adaptions showed in Section IV. The proof shows that the γ-score reaches the highest value (numerator and denominator in eq. (17) are equal).

Theorem 1. *Under the hypotheses of Lemma 1 and given a support set $\mathcal{S}$ of S rotations, $\gamma = 1$ for our revised rotation-invariant RBM model.*

Proof is provided in Appendix B. The proof of this theorem shows that our method achieves full invariance w.r.t. the γ-score. We can make the following remarks about the theorem:

Remark 1. *Theorem 1 does not make any assumptions how the $\mathbf{r}$ vector is computed, as long as eq.* (10) *holds.*

Remark 2. *Theorem 1 is a theoretical result and does not account for artifacts due to the discrete nature of inputs and*

rotations. Empirical computations of the γ-score might result in slightly lower values.

Remark 3. *Lemma 1 is compatible with any typical additional terms that can be added in eq.* (8)*, such as momentum and L_2 regularizer [40].*

Remark 4. *Optimizing eq.* (9) *as in Section IV, the negative log-likelihood $-\ln p(\mathbf{x}|\Theta)$ is minimized as well.*

V. Inference Of The Dominant Orientation

In this section, we describe how to infer the optimal $\mathbf{r}$ vector for an input $\mathbf{x}$ in the dataset. We propose an approach that exploits the intrinsic information learned by the network during training, using the reconstruction error. In our formulation, we can define the reconstruction function $\varphi(\mathbf{x}, \mathbf{r})$ as $\varphi(\mathbf{x}, \mathbf{r}) = v(h(\mathbf{x}, \mathbf{r}), \mathbf{r})$, where $h(\mathbf{x}, \mathbf{r}) = p(\mathbf{h}|\mathbf{x}, \mathbf{r})$ and $v(\mathbf{h}, \mathbf{r}) = p(\mathbf{x}|\mathbf{h}, \mathbf{r})$. We define the dominant orientation for an input $\mathbf{x}$ as the one that minimizes the following function:

$$\begin{aligned} &\hat{s} = \operatorname{argmin}_s \|\varphi(\mathbf{x}, \mathbf{r}) - \mathbf{x}\|_2^2, \\ &\text{such that} \quad r_t = 0 \text{ and } r_s = 1, s \neq t. \end{aligned} \tag{18}$$

Thus, the corresponding $\mathbf{r}$ for the input $\mathbf{x}$ is $r_{\hat{s}} = 1$ and $r_t = 0$, $t \neq \hat{s}$. This satisfies the one-hot encoding constraint in eq. (10). The optimization of eq. (18) can be easily computed for all the possible values of $\mathbf{r}$, as it comes automatically during the forward pass of the training process.

A. Implementation Details

Training. Our training algorithm is an extended version of the *Contrastive Divergence* [38]. In our implementation, we represent the third-order tensor $\mathbf{W}$ explictly, although all the slices are rotated versions of each other. As discussed in the previous section, the core part of our architecture is the inference of the dominant orientation. For each minibatch $\mathcal{B}$, we compute the reconstructed input using all weight matrices in $\mathbf{W}$. For each image in $\mathcal{B}$, the dominant orientation is inferred, by selecting the weight matrix that better reconstruct the input (c.f. fig. 1). Then, the gradients to update the parameters Θ are computed and the contribution of all the $\nabla\overset{\circ}{\mathbf{W}}_{\mathbf{s}}$ is shared with the other weight matrices. Other gradients coming from e.g. sparsity regularizer [40], the KL-Divergence in eq. (19), or momentum are also used to update the parameters of the network. Details are shown in Algorithm 1. Our Theano [41] implementation takes ~ 0.8s for training on CPU (Intel Xeon E5-1660), and ~ 0.4s on GPU (TITAN Xp) per batch.

Testing. During inference, an input image is provided to the network, and is passed through to obtain activations using all the weight matrices (one per orientation). The hidden layer activations produced by the weight matrix minimizing the reconstruction error are selected. The chosen activations represent the features of the input image (cf. fig. 1).

Algorithm 1: Training procedure of our proposed rotation-invariant Restricted Boltzmann Machine.

Data: Training set $\mathcal{X}$, parameters $\Theta = \{b_j, c_k, w_{jks}\}$
Result: Updated parameters Θ

```
begin
    for e:= 1 to MaxEpochs do
        foreach mini-batch B ⊂ X do
            Perform a forward step with B of the network
              and find ŝ by minimizing eq. (18).
            Compute the gradient for the slice ∇W_ŝ, as
              well as ∇c, and ∇b as in eqs. (3) to (5).
            foreach t ∈ {0, ..., S − 1} \ {ŝ} do
                Share the gradient of ∇W_ŝ and obtain
                ∇W°_t using eq. (13).
            end
            Apply the update rule in eq. (8) to all the
            parameters in Θ using ∇W_ŝ and the shared
            gradients from all ∇W°_t.
            Update all the parameters in Θ with any other
              gradients coming from momentum or
              regularizer(s) (e.g., eq. (19)).
        end
    end
end
```

B. KL-Divergence Regularization Term to Improve Dominant Orientation Inference

The rotation estimation approach may potentially assign most inputs to one dominant orientation. To avoid this, we opted to regularize the training process, by forcing a prior on the distribution of orientations across the dataset. We achieve this by minimizing the following Kullback-Leibler divergence:

$$D_{KL}(\mathbf{p}||\bar{\mathbf{r}}) = \lambda_r \sum_{s=0}^{S-1} p_s \ln \frac{p_s}{\bar{r}_s}, \tag{19}$$

where $\mathbf{p}$ is a prior distribution, $\bar{\mathbf{r}}$ is the average assignment of the dominant orientation of the images in the training set, as discussed in Section V, and λ_r is a positive constant weighing the strength of the regularizer. Following [42], we compute the average prediction vector $\bar{\mathbf{r}}$ over a mini-batch, rather than the whole training set.

C. Consistency Analysis

In this section, we want to assess the consistency of the predictions performed by our approach to infer the dominant orientations, computing what we define the *change probability*. During training, we tracked the predictions made by our algorithm to infer the dominant orientation of each image in the training set. Then, we analyzed how many times each image has been assigned to a dominant orientation over time. We computed the probability at each epoch that an assignment change occurs and we plotted the result in fig. 2.

It can be observed that our inference method stabilizes in less than 10 epochs, becoming very consistent in ≈ 40 epochs

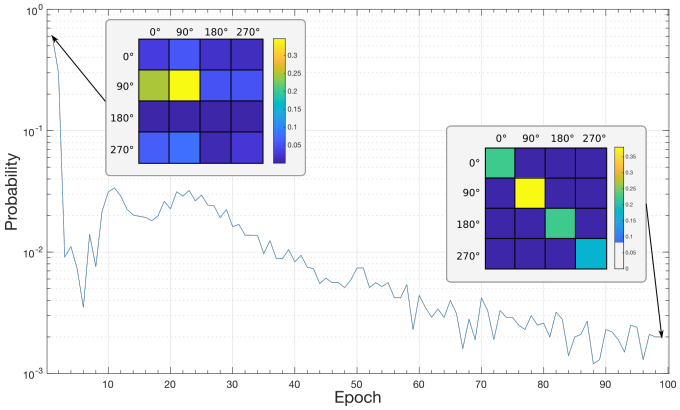

Fig. 2. Probability (in log-scale) of a prediction change of the dominant orientation occurs during training on the *mnist-rot* dataset (*change probability*). The stability of our inference method increases exponentially over time. We also show two inset transition matrices: the left-hand side inset shows the amount of misclassification between the first two epochs, whereas the right-hand side for the last two epochs. (Best viewed in color).

(probability of a reassignment is very close to 0). Furthermore, fig. 2 contains two inset transition matrices, where we show how assignments are redistributed between two consecutive epochs. Specifically, the left-hand side transition matrix show reassignments occurring between the first two epochs, whereas the right-hand side inset shows the changes occurring in the last two epochs. These two plots show that, although the initial predictions of the dominant orientation are unstable, the network is able to automatically assign each image to the same class of orientation.

In Supplemental Material Sec. II, we show a consistency analysis of the estimation orientation. In particular, we show the robustness of our approach to error in estimation during training, and consistent predictions are at test time. Experimental results show that our algorithm can 'self-correct' estimation errors occurred during training and we also show high consistency during testing.

VI. Experimental Results

We demonstrate our model on the following datasets: *mnist-rot* [8], the *MPEG-7 Shape Silhouette database* [43], and a rotated version of the *zalando fashion-mnist* dataset [44]. We compared our performance with the following baseline and state-of-the-art approaches:

- *Support Vector Machine [15]:* SVM trained directly on the data, without preprocessing.
- *RBM [4]:* the original model will provide a baseline result for our experiments.
- *RBM [4] with data augmentation:* we also compare with the original model trained with augmented data (we will refer to this method as RBM+).
- *TI-RBM [16]:* state-of-the-art method for learning transformation-invariant features. Specifically, we only used rotations as transformations.
- *ERI-RBM [14]:* state-of-the-art approach that computes rotation-invariant features, using histograms of gradients approach to split the dataset according to the dominant orientation of the inputs.

R4.1

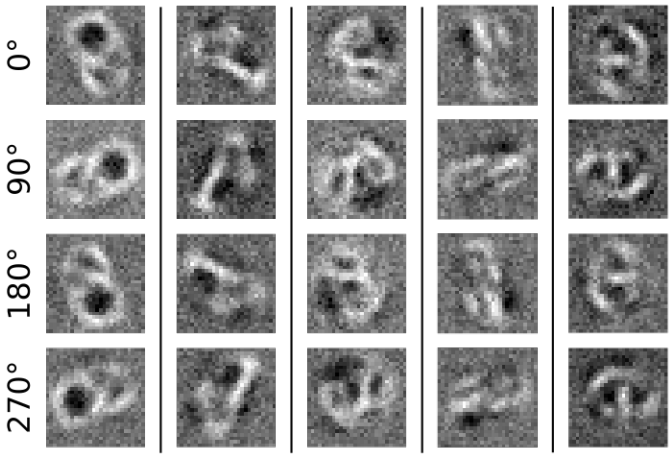

Fig. 3. Filters learned by our model IRI-RBM for the *mnist-rot* dataset [8]. We display for brevity a set of 5 filters for each of the $S = 4$ weight matrix.

A. Parameters

If not otherwise stated, we run all the experiments using the following parameters, which were set the same for all methods. We trained RBMs with $H = 500$ hidden units for 100 epochs, using a learning rate $\eta = 0.003$. We also adopted a sparsity regularizer target $p = 0.1$ with regularization constant $\lambda = 0.003$ [40].[2] Furthermore, for the regularizer in eq. (19), we set the constant $\lambda_r = 100$ and the prior probability distribution $\mathbf{p} = \mathcal{U}(0, S-1)$. We used a momentum $\alpha = 0.5$ for the first 5 epochs, then we increased it to $\alpha = 0.9$. If it is not explicitly specified, number of rotations is set to $S = 4$, as angles multiple of 90° avoid pixel interpolation (this allows fair comparison with the other baseline and state-of-the-art approaches, but we show later the effect of changing the number of rotations S). We initialized the weight matrices using the *Glorot* method [45], using random numbers sampled from a Gaussian distribution with zero mean and standard deviation of $\sqrt{2/(V+H)}$, where bias terms are initialized with 0s. For classification, we followed the same protocol as in [14], adopting SVM [15] with an RBF kernel. For the classifier,[3] we set the spread parameter $\sigma = 0.002$. The loss parameter C is set accordingly for each dataset. Experiments were repeated 5 times, with different initialization, and mean and standard deviation were computed.

B. Experiments

Tests on mnist-rot [8]. This well-known benchmark dataset contains 10,000 images for training and 50,000 for testing of hand-written digits. For training, we adopted the parameters discussed in Section VI-A, setting $C = 10$ for the classifier.

Table I shows the results of our experiments. Overall, TI-RBM and ERI-RBM perform similarly on this setup, outperforming the baseline by $\approx 10\%$. Our proposed method obtains the best performance, achieving more than 92% test accuracy.

[2] As reported in [40], we only update the bias of the hidden units $\mathbf{b}$.

[3] We set $\sigma = 0.02$ when trained directly on the data (first row on Table I), as this value gave the best performance.

TABLE I
TEST ACCURACY OF OUR METHOD COMPARED WITH SVM [15], THE ORIGINAL FORMULATION OF RBM [4], RBM TRAINED WITH DATA AUGMENTATION (RBM+), THE TRANSFORMATION-INVARIANT RBM [16], AND THE EXPLICIT ROTATION-INVARIANT RBM [14] ON THREE DIFFERENT DATASETS. WE REPORT "*best result (mean ± std)*" OF 5 RANDOM INITIALIZATIONS.

Method	mnist-rot [8]	MPEG-7 [43]	zalando mnist-rot
SVM [15]	89.36%	82.00%	74.71%
RBM [4]	80.18% (79.91% ± 1.96)	78.57% (78.28% ± 0.27)	62.99% (62.92% ± 0.10)
RBM+	86.90% (76.71% ± 0.15)	71.57% (70.66% ± 0.87)	67.54% (67.34% ± 0.13)
TI-RBM [16]	89.90% (89.75% ± 0.16)	76.14% (75.14% ± 0.57)	76.54% (76.42% ± 0.12)
ERI-RBM [14]	90.61% (90.48% ± 0.13)	73.00% (72.52% ± 0.25)	74.84% (74.49% ± 0.18)
Ours	**92.12%** (91.81% ± 0.30)	**85.71**% (83.43% ± 1.38)	**77.14**% (76.94% ± 0.24)

This result shows that our method of inferring rotations is more reliable than e.g., max-pooling across all rotations [16], or relying on exogenous methods [14]. Then, we empirically computed the γ-score as described in eq. (17) and our method scored $\gamma = 0.98$, as expected from Theorem 1. In fig. 3, we show a subset of the filters learned by our method and, as it can be observed, filters are rotated versions of each other, providing experimental evidence for Lemma 1.

As mentioned in Section I, our method learns compact representations with high discriminative power. We compared also with the *Contractive Autoencoder* on the same dataset [46]. This method minimizes a regularization term based on the Jacobian matrix of the encoder step of the network to learn invariant features. Their results with 1,000 hidden units showed a smaller test accuracy of 90.34%. In comparison, our method learns high discriminative features with half of the hidden units, thus learning a more compact representation.

Tests on a small training set. Here, we want to demonstrate that our method learns robust features even when trained on a small dataset. We used the *MPEG-7 Shape Silhouette database* [43], containing only 1,400 images belonging to 70 categories. Since the images have a variable size, we cropped and resized them to 28×28 pixels. We randomly split the dataset into 700 images for training and 700 for testing, maintaining class balance. In this case, we set the loss parameter for SVM $C = 100$.

Results on this dataset are also reported in Table I. Our method outperforms all other approaches, reducing the testing error by $\approx 10\%$. Specifically, we can observe that TI-RBM [16] and ERI-RBM [14] suffer from lack of data in the training set, obtaining a testing accuracy lower than the SVM and RBM baselines. On the other hand, RBM is not able to accommodate the rotational variance in this dataset, causing it to perform poorly compared with our approach. Therefore, our method can learn better rotation-invariant features also in the case of a reduced training set.

Tests on the rotated zalando fashion mnist dataset. We also tested our approach on a customized version of the *zalando fashion mnist* dataset [44]. Specifically, the original dataset contains images of 10 categories of clothes. Images are grayscale and 28×28 pixels size.[4] For these experiments, we generated a rotated version of the dataset, using uniformly distributed random rotations. To create this customized dataset, we adopted the original code from [8] used to generate *mnist-rot*.[5] We generated 10,000 images for training and 50,000 for testing [8]. To the best of our knowledge, an equivalent *mnist-rot* for the *zalando fashion mnist* has not been created yet. We refer to such a generated dataset as *zalando fashion mnist-rot*. Results on this version of the *zalando* dataset are also shown in Table I.[6] Overall, we can observe that our method outperforms all the other approaches on this dataset as well.

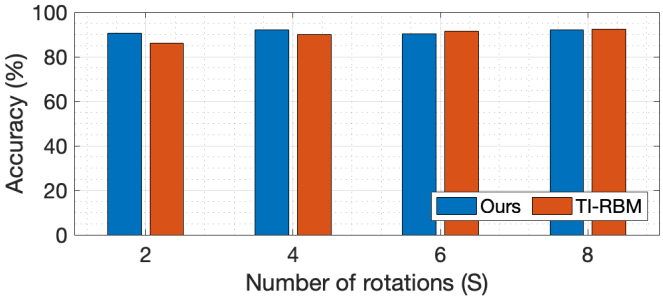

Fig. 4. Comparison of our approach and TI-RBM [16] with respect to the number of rotations S on mnist-rot test dataset [8]. Overall, TI-RBM requires the double of rotations (S=8) to match our performance.

The effect of increasing the number of rotations S. Here, we asssess the effect of increasing the number S of rotations during training. Comparing with TI-RBM [16], we determine the minimum number of rotations S required by TI-RBM to match our performance. In fig. 4, we report the classification accuracy when the methods are trained on *mnist-rot* and *zalando mnist-rot* datasets. Overall, TI-RBM requires $S = 8$ rotations to match our best performance. Our method improves when $S = 8$ is used, although it results in a minimal increase in performance. Thus, our method can sufficiently learn highly discriminative rotation-invariant features with $S = 4$.

Discussion. From our experiments, it appears that it is better to rely on the intrinsic information encoded in the

[4]Further details on the *zalando fashion mnist* at https://github.com/zalandoresearch/fashion-mnist.

[5]Available at http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/DeepVsShallowComparisonICML2007.

[6]We also trained our model on the original *zalando fashion mnist* dataset [44] and it performed similarly to an RBM (84.55% vs 85.99% for ours and RBM respectively). Furthermore, our method had a higher accuracy compared with TI-RBM and ERI-RBM. Thus, our method works well when there are no rotations present in the training set.

TABLE II
ABLATION RESULTS SHOWING THAT OUR METHOD BENEFITS FROM SHARED GRADIENTS AND THE REGULARIZER PRESENTED IN SECTION V-B.

Method	mnist-rot [8]	MPEG-7 [43]	zalando mnist-rot
RBM [4] (as a reference)	80.18%	78.57%	62.99%
Ours without sharing gradients and eq. (19)	53.80%	68.71%	56.76%
Ours without eq. (19)	91.96%	82.14%	76.21%
Ours (proposed method)	92.12%	85.71%	77.14%

network to infer the dominant orientation. In this way, our model explicitly cancels out the nuisance given by rotations, producing fully rotation-invariant image representations with a reduced number of rotations. We showed that our approach is better than marginalizing across all possible rotations, as it happens in TI-RBM [16], or using an exogenous process to estimate orientations, as in ERI-RBM [14]. Our method is better than the standard approach of training a neural network with data augmentation (RBM+ in Table I). This shows that data augmentation does not automatically ensure the learning of invariant features, as it still needs to learn variability in the dataset [31]. In addition, we showed that our method works particularly well in datasets with small size. To further demonstrate this, we compared our method with TI-POOLING [31], a recent deep learning method for transformation-invariant feature learning. Most importantly and differently from all the methods described in Table I, TI-POOLING is a deep network, trained end-to-end with supervision.

R4.1

We trained TI-POOLING with 4 rotations and we set the size of the last fully connected layers to 500 to maintain a similar setup as the unsupervised methods described in Section VI. Although TI-POOLING outperformed our method by $\sim 5\%$ on the mnist-rot and zalando-rot datasets, classification accuracy in the MPEG-7 dataset was $78.11\% \pm 1.28$ (best results was 79.57%), compared to 85.71% for ours. This indicates that deep architectures require big datasets to efficiently train their network parameters.

R2.2

C. Ablation Experiments

We want to assess how our approach benefits from the gradient sharing step [14] and the KL-Divergence based regularizer described in Section V-B. Experiments were performed using the same protocol as discussed in the previous sections. We show the result of our experiments in Table II.

To establish a reference baseline, we trained the original RBM model [4]. Next, we trained our model disabling sharing gradients and eq. (19). In this case, our model has lower performance compared to the baseline. Training our network without shared gradients is similar to training S different RBMs, such as the s-th model is trained with only the $\mathcal{X}_s$ partition of the data. This means that the training set $\mathcal{X}$ is split and each RBM is trained independently on a smaller portion of the data. This procedure is closely related to the *Oriented RBM* baseline method described in [14]. By enabling the shared gradients, the performance of our approach improves by 20% (even $\approx 40\%$ on *mnist-rot*), showing that this technique is effectively improving the training. The gradient sharing step also ensures the learning or rotation-invariant features (see Theorem 1), thanks to the rotation equivalence property in eq. (14). When the regularizer in Section V-B is also enabled during training, performance improves further, as the inference of the dominant orientation becomes more robust and reliable.

D. Unsupervised vs. Supervised Rotation Inference

In this experiment, we want to assess the performance of our unsupervised method to infer the dominant orientation (c.f. Section V). It is important to point out that making the *actual* correct orientation estimation is not the purpose of our method. What is indeed important is the consistency of the predictions during training. As a first test, we compared the performance of our algorithm with an SVM classifier in the dominant orientation prediction task, using the ground truth rotations (c.f. Section VI-B), setting $C = 1$ and $\gamma = 0.02$ as parameters. As expected, the supervised SVM classifier outperformed our unsupervised approach. Specifically, on the *zalando mnist-rot* dataset, our method predicts the correct dominant orientation with an accuracy of 60% vs 92% obtained by SVM.

Given this result, we trained our rotation-invariant RBM, using a supervised SVM classifier loss: we replaced our inference process with the SVM classifier to infer the dominant orientation during training. In Table III, we show the result of this test. Although using a classifier loss minimally improves on *mnist-rot*, the performance of the unsupervised and supervised approaches are the same on the *zalando mnist-rot*. This shows that it is important to make a consistent decision in predicting the dominant orientation during training.

As a further test, we trained our rotation-invariant RBM using the actual ground truth rotations, without employing any (un)supervised processes. This test establishes an upper bound performance of our rotation-invariant RBM. Test accuracy is reported in Table III. Overall, the performance of our unsupervised approach edges with the upper bound computed using ground truth rotations. Although our unsupervised method may make errors in predicting the actual rotation, its performance is comparable to both the supervised and the upper bound performance. Overall, using the actual ground truth rotations improves the test accuracy of $\sim 2\%$.

VII. CONCLUSIONS

Finding suitable features for the task at hand is considered hard in computer vision. We presented a novel method to extract rotation invariant features, extending the original model for Restricted Boltzmann Machines (RBMs). The core part of our method is the inference of the dominant orientation of the input, that is done by minimizing the reconstruction error. In order to have more robust inference of such a dominant orientation, we regularize the learning process with a term

TABLE III
COMPARISON OF OUR UNSUPERVISED DOMINANT ORIENTATION INFERENCE METHOD WITH TWO SUPERVISED METHODS. *2nd row*: THE INFERENCE METHOD IN SECTION V WAS REPLACED WITH AN SVM TRAINED DIRECTLY ON GROUND TRUTH ROTATIONS. *3rd row*: RBM IS TRAINED ON GROUND TRUTH ROTATIONS.

	Method	mnist-rot [8]	zalando mnist-rot
Uns.	Ours	92.12%	77.14%
Sup.	Ours using SVM	93.32%	77.14%
	Ours with GT rotations	93.96%	79.38%

derived from the KL-Divergence. This regularizer enforced a prior distribution over the dominant orientation in the dataset.

We evaluated our method on three publicly available datasets and it outperformed baseline and state-of-the-art approaches in all the cases. Our approach scored $\gamma = 0.98$, demonstrating full rotation invariance. Furthermore, we also showed that our method can learn highly discriminative features in the case of a reduced training set. Our ablation experiments showed that our method benefits from the sharing gradient method [14], as well as the regularizer based on the KL-Divergence. This allows our method to compete with supervised methods, such as SVM trained to infer dominant orientations. This was further demonstrated by a consistency analysis, where rotated images obtain the correct dominant orientation prediction w.r.t. the unrotated version. We also showed that the training of the network is not affected by errors in the estimation of the dominant orientation.

We showed that our approach outperforms the other shallow state-of-the-art methods, although is still challenged by the size of the input image. A future research direction is to embed our method within a deep network, where a fully-connected layer can be extended with a third-order tensor to accommodate a set of rotations $\mathcal{S}$ to extract rotation-invariant features.

In conclusion, our proposed method explicitly factorizes rotational nuisance from the training set, learning high discriminative and compact features. In fact, experimental evidence showed that explicit unsupervised feature learning performs better than the others (e.g., [14], [16]). Our python implementation is available at `https://bitbucket.org/tuttoweb/rotinvrbm`.

APPENDIX A
PROOF OF LEMMA 1

Proof. We will proceed by induction over the iteration t. For the base case $t = 0$, we impose that:

i. $\tilde{\mathbf{W}} \in R^{V \times H}$ is a matrix initialized with e.g., Glorot Gaussian method [45],[7]
ii. $\mathbf{W}_s^{(0)} = R_s(\tilde{\mathbf{W}}), \forall s \in \{0, 1, \ldots, S-1\}$.

This means that all slices in $\mathbf{W}^{(0)}$ are initialized as rotated versions of $\tilde{\mathbf{W}}$, which initially can be any matrix. Now, let us suppose that the lemma is true until $t-1$, and demonstrate it for t. Then we have:

[7]We observed that the base case of the induction can be relaxed. Experimental evidence showed that by initializing W with random numbers drawn from a normal distribution it is still possible to have rotation-invariant features.

$$
\begin{aligned}
R_\kappa(\mathbf{W}_s^{(t)}) &= \underbrace{R_\kappa\left(\mathbf{W}_s^{(t-1)} + \eta \nabla \overset{\circ}{\mathbf{W}}{}_s^{(t-1)}\right)}_{\text{From eq. (8)}} \\
&= \underbrace{R_\kappa\left(\mathbf{W}_s^{(t-1)}\right) + \eta R_\kappa\left(\nabla \overset{\circ}{\mathbf{W}}{}_s^{(t-1)}\right)}_{\text{Linearity property (c.f. Section IV)}} \qquad (20) \\
&= \underbrace{\mathbf{W}_{s'}^{(t-1)}}_{\text{By induction}} + \eta \underbrace{\nabla \overset{\circ}{\mathbf{W}}{}_{m(s,\kappa)}^{(t-1)}}_{\text{From (13)}}.
\end{aligned}
$$

At this point, we need to expand the modulo function. Applying (15), we obtain that $m(s,\kappa) = (s+\kappa) \bmod S = (s + s' - s) \bmod S = s'$. Thus, to conclude, eq. (20) becomes:

$$
\begin{aligned}
R_\kappa(\mathbf{W}_s^{(t)}) &= \mathbf{W}_{s'}^{(t-1)} + \eta \nabla \overset{\circ}{\mathbf{W}}{}_{m(s,\kappa)}^{(t-1)} = \\
&= \mathbf{W}_{s'}^{(t-1)} + \eta \nabla \overset{\circ}{\mathbf{W}}{}_{s'}^{(t-1)} = \mathbf{W}_{s'}^{(t)}.
\end{aligned}
$$

□

APPENDIX B
PROOF OF THEOREM 1

Proof. We have to prove that $\gamma_j = 1$, $j = 1, 2, \ldots, H$. From eq. (17), we will show that the numerator and denominator coincide. Now, starting from the definition of μ_j showed in eq. (16), we obtain:

$$
\begin{aligned}
\mu_j(\mathbf{x}) &= \frac{1}{S}\sum_{q=0}^{S-1} h_j(R_q(\mathbf{x})) \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \underbrace{\sigma\left(\sum_{t=0}^{S-1} r_t\left(b_j + R_q(\mathbf{x})^T W_{j,\cdot,t}\right)\right)}_{\text{From eq. (11)}} \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \underbrace{\sigma\left(b_j + R_q(\mathbf{x})^T W_{j,\cdot,s'}\right)}_{\text{From (10), } \exists s' \text{ s.t. } r_{s'}=1} \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \underbrace{\sigma\left(b_j + R_q(\mathbf{x})^T R_q(W_{j,\cdot,s})\right)}_{\text{From Lemma 1, } \exists s : R_q(\mathbf{W}_s) = \mathbf{W}_{s'}} \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \sigma\left(b_j + [\mathbf{R_q x}]^T \mathbf{R_q} W_{j,\cdot,s}\right) \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \underbrace{\sigma\left(b_j + \mathbf{x}^T \mathbf{R}_q^T \mathbf{R}_q W_{j,\cdot,s}\right)}_{(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T} \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \underbrace{\sigma\left(b_j + W_{j,\cdot,s}\mathbf{x}\right)}_{\mathbf{R}_q^T\mathbf{R}_q = \mathbf{R}_q\mathbf{R}_q^T = \mathbf{I}} \\
&= \frac{1}{S}\sum_{q=0}^{S-1} \sigma\left(\sum_{t=0}^{S-1} r_t\left(b_j + \mathbf{x}^T W_{j,\cdot,t}\right)\right) \\
&= \frac{1}{S}\sum_{q=0}^{S-1} h_j(\mathbf{x}) = h_j(\mathbf{x}).
\end{aligned}
$$

Since $\mu_j(\mathbf{x}) = h_j(\mathbf{x})$, then also their variance over all the samples in the training set is equal. Therefore $\gamma_j = 1$. □

REFERENCES

[1] Yoshua Bengio. Deep Learning of Representations for Unsupervised and Transfer Learning. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 17–36, Bellevue, Washington, USA, 2012. PMLR.

[2] Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 342–350. Curran Associates, Inc., 2009.

[3] M. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[4] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–54, 2006.

[5] Pascal Vincent and Hugo Larochelle . Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pierre-Antoine Manzagol. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

[6] Yann LeCun. Learning invariant feature hierarchies. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7583 LNCS(PART 1):496–505, 2012.

[7] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:991–999, 2015.

[8] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, pages 473–480. ACM, 2007.

[9] Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. In *ICCV*, pages 5048–5057, 2017.

[10] Josephine J. Cock, Dianne C. Berry, and E. A. Gaffan. New strings for old: The role of similarity processing in an incidental learning task. *The Quarterly Journal of Experimental Psychology Section A*, 47(4):1015–1034, 1994.

[11] Eric R. Kandel and Robert D. Hawkins. The biological basis of learning and individuality. *Scientific American*, 267(3):78–87, 1992.

[12] Diego Marcos, Michele Volpi, and Devis Tuia. Learning rotation invariant convolutional filters for texture classification. page 6, 2016.

[13] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *ICML*, pages 1064–1071, New York, NY, USA, 2008. ACM.

[14] Mario Valerio Giuffrida and Sotirios A. Tsaftaris. Rotation-Invariant Restricted Boltzmann Machine Using Shared Gradient Filters. In *ICANN*, pages 480–488. Springer International Publishing, Cham, 2016.

[15] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[16] Kihyuk Sohn and Honglak Lee. Learning Invariant Representations with Local Transformations. *ICML*, pages 1311–1318, 2012.

[17] Antti Rasmus, Tapani Raiko, and Harri Valpola. Denoising autoencoder with modulated lateral connections learns invariant representations of natural images. *arXiv:1412.7210*, 2014.

[18] Ian Goodfellow, Honglak Lee, Quoc V. Le, Andrew Saxe, and Andrew Y. Ng. Measuring invariances in deep networks. In *NIPS 22*, pages 646–654. 2009.

[19] Zheng Shou, Yuhao Zhang, and H J Cai. A study of transformation-invariances of deep belief networks. In *IJCNN*, pages 1–8. IEEE, 2013.

[20] Jyri J Kivinen and Christopher K I Williams. Transformation Equivariant Boltzmann Machines. In *ICANN*, volume 6791, pages 1–9. Springer, 2011.

[21] William T. Freeman and Edward H Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):891–906, 1991.

[22] Eero P Simoncelli, William T Freeman, Edward H Adelson, and David J Heeger. Shiftable multiscale transforms. *IEEE transactions on Information Theory*, 38(2):587–607, 1992.

[23] Pietro Perona. Deformable kernels for early vision. *IEEE Transactions on pattern analysis and machine intelligence*, 17(5):488–499, 1995.

[24] M. Unser, N. Chenouard, and D. Van De Ville. Steerable pyramids and tight wavelet frames in $L_2(\mathbb{R}^d)$. *IEEE Transactions on Image Processing*, 20(10):2705–2721, Oct 2011.

[25] Michael Unser and Nicolas Chenouard. A unifying parametric framework for 2d steerable wavelet transforms. *SIAM Journal on Imaging Sciences*, 6(1):102–135, 2013.

[26] Dongyang Cheng, Tanfeng Sun, Xinghao Jiang, and Shilin Wang. Unsupervised feature learning using Markov deep belief network. In *2013 IEEE ICIP*, pages 260–264. IEEE, Sep 2013.

[27] David G Lowe. Distinctive Image Features from Scale-invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[28] Uwe Schmidt and Stefan Roth. Learning rotation-aware features: From invariant priors to equivariant descriptors. In *CVPR*, pages 2050–2057, 2012.

[29] Mohammad Norouzi, Mani Ranjbar, and Greg Mori. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. In *CVPR*, pages 2735–2742. IEEE, 2009.

[30] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray Kavukcuoglu. Spatial transformer networks. In *NIPS 28*, pages 2017–2025. Curran Associates, Inc., 2015.

[31] D. Laptev, N. Savinov, J. M. Buhmann, and M. Pollefeys. Ti-pooling: Transformation-invariant pooling for feature learning in convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 289–297, June 2016.

[32] Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Oriented response networks. In *CVPR*, July 2017.

[33] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *CVPR*, 2017.

[34] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

[35] Bernardino Romera-Paredes and Philip Hilaire Sean Torr. Recurrent Instance Segmentation. In *ECCV*, pages 312–329. Springer International Publishing, Cham, 2016.

[36] Mengye Ren and Richard S. Zemel. End-to-end instance segmentation with recurrent attention. In *CVPR*, 2017.

[37] Ammar Mahmood, Mohammed Bennamoun, Senjian An, Ferdous Sohel, Farid Boussaid, Renae Hovey, Gary Kendrick, and Robert B Fisher. Chapter 21 - Deep Learning for Coral Classification. In *Handbook of Neural Computation*, pages 383–401. Academic Press, 2017.

[38] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[39] Yoshua Bengio and Olivier Delalleau. Justifying and generalizing contrastive divergence. *Neural Computation*, 21(6):1601–1621, 2009.

[40] Honglak Lee, Chaitanya Ekanadham, and Andrew Y Ng. Sparse deep belief net model for visual area v2. In *NIPS 20*, pages 873–880, 2008.

[41] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, and et al. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.0, 2016.

[42] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint Optimization Framework for Learning with Noisy Labels. 2018.

[43] N. Alajlan, M.S. Kamel, and G.H. Freeman. Geometry-Based Image Retrieval in Binary Image Databases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1003–1013, jun 2008.

[44] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[45] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *AISTATS*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. PMLR, 2010.

[46] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, pages 833–840. Omnipress, 2011.