Learning Invariance in Deep Neural Networks*

Han Zhang[1,2] and Tomasz Arodz[1][0000−0002−9215−5522]

[1] Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284, USA
tarodz@vcu.edu

[2] Department of Computer Science and Technology, School of Data Science and Artificial Intelligence, Dongbei University of Finance and Economics, Dalian, China
hanzhang@dufe.edu.cn

Abstract. One of the long-standing difficulties in machine learning involves distortions present in data – different input feature vectors may represent the same entity. This observation has led to the introduction of invariant machine learning methods, for example techniques that ignore shifts, rotations, or light and pose changes in images. These approaches typically utilize pre-defined invariant features or invariant kernels, and require the designer to analyze what type of distortions are to be expected. While specifying possible sources of variance is straightforward for images, it is more difficult in other domains. Here, we focus on learning an invariant representation from data, without any information of what the distortions present in the data, only based on information whether any two samples are distorted variants of the same entity, or not. In principle, standard neural network architectures should be able to learn the invariance from data, given sufficient numbers of examples of it. We report that, somewhat surprisingly, learning to approximate even a simple types of invariant representation is difficult. We then propose a new type of layer, with a richer output representation, one that is better suited for learning invariances from data.

Keywords: Invariant learning · Deep learning · Autoencoder.

1 Introduction

Machine learning deals with many application scenarios which pose difficulties for the learning method. These include supervised learning problems with highly skewed distribution of cardinalities across classes [9], and data with concept drift [5]. But even in the absence of the above problems, learning may be difficult – one example involves scenarios where data becomes distorted, in some unknown way, prior to being captured; that is, the same object can be represented by multiple different feature vectors [19]. Methods that provide invariance to specific types of distortions, such as image translation or rotation, have a long history in pattern recognition. Approaches based on Fourier transform [10], Zernike moments

* Supported by NSF grant IIS-1453658.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_6

[8], and Radon transform [1], have been used to generate invariant features that can be used in downstream learning methods. Invariant kernels have also been proposed [6]. In constructing deep learning models, convolutional and pooling layers [12] offer limited amount of invariance to translation [13], and dedicated methods for achieving more robust translational invariance [11, 18], or rotation invariance [3] have been proposed. What unites these approaches is that the type of invariance the model can handle is inherent in the construction, and is fixed. Learning the invariance from training data is an alternative approach. It is common in the language understanding domain, where word embeddings are learned, using large text corpus, to provide very similar vector representation to words that have the same meaning, but are different [14, 16, 15]. Outside of language modeling, the first method to learn invariances from training data, Augerino [2], has been proposed recently; it operates by specifying a parametric distribution over various augmentations of the input, and learning which augmentations, to which the model should be invariant, are useful. One should note that learning representations that are invariant over individual samples, as above, is different from learning representations that are invariant, with respect to some distribution characteristics, over several populations, often from different domains, as is common in domain adaptation approaches [20]. It is also distinct from the problem of learning equivariant representations, that is, representations in which variation on input leads to an equivalent variation in the representation [17].

Here, we propose a different method for learning invariant neural networks from data. Instead of defining a priori which transformations of input the network should ignore, or learning it through analyzing various input augmentations, we consider a scenario in which the training set consists not only of input samples, but also includes information which samples are the same, up to some transformation. For example, the training set may consist of several horizontal and vertical mirror version of an image, and based on the information that these images are essentially the same, the network should learn to become invariant to horizontal and vertical axial symmetry. More formally, for samples $x \in \mathcal{X}$, given an unknown family of transformations $\{v_\theta : \mathcal{X} \to \mathcal{X}\}$ parameterized by some vector θ, we aim to train a model $F_\beta(x)$, where β are trainable model weights, such that $F_\beta(x) = F_\beta(x')$ if and only if a θ exists such that $x' = v_\theta(x)$. That is, the representation resulting from model F should be the same for the same entity, irrespective of the distortion. On the other hand, two samples that are not distorted variants of the same underlying entity should have different representation. In order to construct network F that produces the invariant representation, we consider an auto-encoder architecture, consisting of an encoder and a decoder network. In our design, the intermediate layer resulting from an encoder network is partitioned into two parts: the invariant part representing the desired F_β, and the variance part that is needed during training for the decoder to reconstruct the auto-encoder's input and can be discarded after training. To facilitate learning the invariance, we assume the training set consists of triples $(x, x', d) \in \mathcal{X} \times \mathcal{X} \times \mathbb{R}_+$, where d captures whether samples x and x' are the same,

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_6

up to a distortion, or not; we use this information to equip the auto-encoder with an invariance-promoting loss.

The rest of the paper is organized as follows. In Section 2, we describe the autoencoder architecture in more detail. In Section 3, we provide experimental evidence that autoencoders constructed using standard neural network building blocks neural network have difficulties in learning invariance from data. In Section 4, we propose a new, richer layer that leads to much more effective learning of invariance from data. In Section 5, we show that an autoencoder built using the new layers can learn nontrivial types of invariance.

2 Architecture for Learning Invariant Representations

An auto-encoder is a neural network that attempts to copy its inputs x to its outputs y, that is $y \approx x$. Internally, it has a latent intermediate layer z that describes a code used to represent some aspect of the input. Since the network's output y is supposed to be similar to input x, we typically are most interested in z, the intermediate layer, not in y. An auto-encoder consists of two parts: an encoder f that transforms the inputs x to intermediate code z ($z = f(x)$) and a decoder g that produces a reconstruction of inputs from the intermediate code $y = g(z) = g(f(x))$. Function f and g are represented by multi-layer network. We want the outputs y to be as close to the inputs x as possible; to achieve that, we use mean-squared error of the reconstruction as the loss

$$L_{auto)}(x, y) = L(x, g(f(x))) = \|x - g(f(x))\|_2^2. \tag{1}$$

The idea of auto-encoders can be applied to dimensionality reduction, feature learning, and pre-training of a deep neural network – in all these cases, the useful part is the encoder, which is for example trained to produce low-dimensional, feature-rich representation of the input. The decoder is added to ascertain that all the relevant information from the input is represented in the result of the encoder, z.

We extend the auto-encoder to include an additional loss ℓ_{inv} operating on the intermediate code $z = f(x)$, to train the network to learn representations with desired properties. Here, we want some part of the code z to be invariant. At the same time, the remaining part of z will capture the information about the distortion, since auto-encoder needs full information about the sample in order to reconstruct it faithfully. In this way, the invariant representation will not be trivial, for example, all null.

Fully-connected feed-forward neural networks with at least one hidden layer and with a nonlinear activation function are universal approximators [7, 4], and are thus capable of modeling arbitrary well-behaved functions. In principle, we should be able to train and encoder-decoder pair that provides invariance in the intermediate code $z = f(x)$, although the network may need to be wide.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_6

3 Learning Invariant Representations is Hard

Before attempting to training networks to learn unknown invariance based on triples $(x, x', d) \in \mathcal{X} \times \mathcal{X} \times \mathbb{R}_+$, we test whether training the network to produce an invariant representation can be achieved using standard building blocks in the simplified scenario when the desired invariance is known, and the invariant representation can be pre-defined by the network designer. That is, given any input x, the desired code $z = f(x)$ is known in advance. In this simplified scenario, the network does not have to come up with the invariant representation on its own. Instead of triples $(x, x', d) \in \mathcal{X} \times \mathcal{X} \times \mathbb{R}$, the network is given pairs (x, z^*), where z^* is the what we desire the code z to be, and can use them to learn the invariant mapping in a supervised way.

To test the ability of standard neural architectures to learn pre-defined invariant representations, we focused on invariance to circular translation. The design of an encoder-decoder with the latent code that is invariant to this transformation is straightforward: the discrete Fourier transform can act as an encoder that transforms input vector x into the frequency domain vector z, where the modulus is invariant to circular shift in the input, and the phase is not invariant. Then, the inverse Fourier transform can be used as a decoder that can put back the modulus and phase to reconstruct the original input. That is, the additional loss operating on the output of the encoder is just

$$\ell_{inv}(z, x) = \|z - \mathrm{DFT}(x)\|_2^2,$$

where $z^* = \mathrm{DFT}(x)$ can be pre-calculated numerically and provided to the network as a supervised training signal. The complete training loss for the network $g \circ f$ is then composed of the auto-encoder and the invariance terms

$$L(x) = \|x - g(f(x))\|_2^2 + \lambda \|f(x) - \mathrm{DFT}(x)\|_2^2,$$

where λ are user-defined coefficients, we used $\lambda = 8$ in the experiments, indicating that we focus more on the intermediate layer than on the auto-encoder reconstruction error.

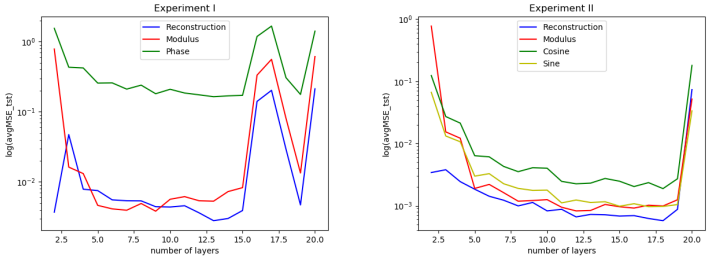
We constructed two datasets, one with 10 features, and one with 20 features. All feature values for all samples are sampled independently from a uniform univariate distribution on $[-1, 1]$. The target supervised signal for the intermediate code z, denoted by z^*, is calculated by performing discrete Fourier transform on each sample, $z^* = \mathrm{DFT}(x)$, which results in a vector of complex numbers of the same dimensionality as the input x. Then, we define four quantities, each being a real-valued vector of the same dimensionality as the input vector x

$$\begin{aligned} z^* &= \mathrm{DFT}(x), \\ \mathit{modulus} &= |z^*| \\ \mathit{phase} &= \mathrm{Phase}(z^*) \\ \mathit{cosine} &= \cos(\mathit{phase}) \\ \mathit{sine} &= \sin(\mathit{phase}). \end{aligned}$$

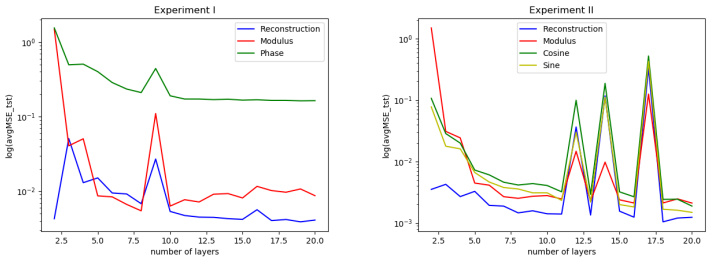
ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_6

Of these four vectors, the modulus is known to be invariant to input translation. We conducted two experiments, representing the desired input representation $z^* = \mathrm{DFT}(x)$ as $[modulus, phase]$ or as $[modulus, sine, cosine]$. In each experiment, the network's output z is compared to the desired output z^*, and the discrepancy in the form of the mean-square error is used as the loss that should be minimized during training. In addition to the training set consisting of 60,000 samples, we also created a separate test set of 10,000 samples following the same protocol. We use stochastic gradient descent (SGD) with batch size 128 to minimize the loss $L(x)$.

We tested four accessible nonlinear activation functions: unipolar sigmoid, bipolar sigmoid, ReLU, and SELU. We tested networks with depth varying between 2 and 20 layers, and we made the networks wide by using $8d$ neurons in each layer, where d is the dimensionality of input vectors x. The results Fig. 1 show that SELU activation function is better than the other three, but none of the four activation functions leads to low MSE.



(a) d=10 (b) d=10



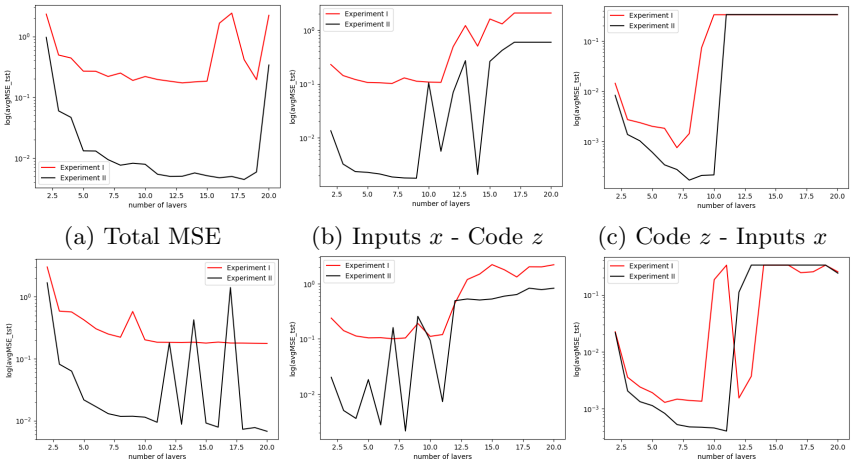
(c) d=20 (d) d=20

Fig. 1: MSE for each part of code z with activation function SELU. Experiment I (left) denotes learning modulus-phase, while Experiment II (right) denotes learning modulus-sine-cosine. We tested inputs with $d = 10$ and $d = 20$ input features.

To exclude the scenario where the joint task of learning the reconstruction and the invariance makes the problem challenging, we compared the MSE for the

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_6

full auto-encoder with the result for only training the encoder f to approximate $z^* = \mathrm{DFT}(x)$ using $\|f(x) - \mathrm{DFT}(x)\|_2^2$ as the only term in the loss. We also trained only the decoder g, that is, we minimized $\|x - g(\mathrm{DFT}(x))\|_2^2$. Results in Figure 2 show that the difficulty comes mostly from training the encoder.



(a) Total MSE (b) Inputs x - Code z (c) Code z - Inputs x

(d) Total MSE (e) Inputs x - Code z (f) Code z - Inputs x

Fig. 2: MSE for training the full auto-encoder (left), just the encoder (center), and just the decoder (right). Plots on top show the results of the dataset of 10 features and 20 features on the bottom.

The results of experiments in this section show that deep and relatively wide networks, with the most popular activation functions, with up to 20 layers and with width exceeding the input dimensionality by a factor of 8, are not suited well to learn invariant representations, even in a simplified scenario where the exact form of the representation is known a priori and can be used as a target training signal in a supervised way.

4 Proposed New Layer for Learning Invariances

We hypothesize that using layers with a linear transformation followed by a single, fixed nonlinearity applied element-wise is to restrictive to learn complicated invariant representation. To alleviate that problem, we define an extended layer

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_6

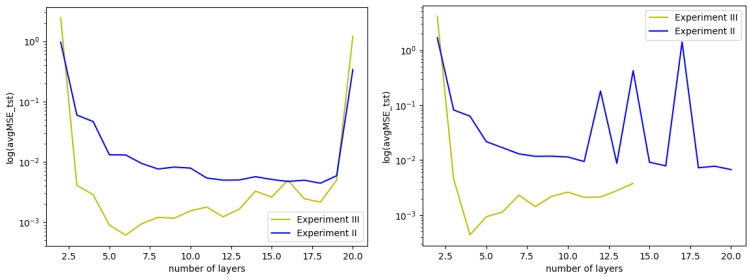
based on a richer set of transformations

$$layer(O_1) = \begin{cases} O_1 & = Output\ from\ previous\ layer, \\ O_2 & = \sqrt{X_{Even}^2 + X_{Odd}^2}, \\ O_3 & = T_{Even} \Big/ O_2, \\ O_4 & = T_{Odd} \Big/ O_2, \\ O_5 & = O_3 * O_4, \\ Outputs & = [O_1, O_2, O_3, O_4, O_5], \end{cases}$$

where

$$\begin{aligned} X &= O_1 W + b \\ X_{Even} &= Choose\ the\ even\ columns\ of\ X, \\ X_{Odd} &= Choose\ the\ odd\ columns\ of\ X. \end{aligned}$$

The layer consists of a skip connection similar to those used in residual networks (O_1), a 2-norm (O_2), a normalized linear transformation (O_3 and O_4), and a normalized quadratic transformation (O_5), concatenated together. Weights W and biases b are the trainable parameters of the layer. The number of columns of O_2, O_3, O_4, and O_5 is all half the number of O_1, thus the dimensionality of the output of the layer is three times the dimensionality of input.



(a) 10 features SELU vs. f_{mixed}

(b) 20 features SELU vs. f_{mixed}

Fig. 3: Comparison of an auto-encoder with standard layers that use SELU activation function (blue) and auto-encoder using the new proposed layers (yellow) on the task of supervised learning of translation invariance, for $d = 10$ (left) and $d = 20$ (right) input features.

The results in Figure 3 show that the new layer is much more capable of approximating the invariance. In the simple case when the invariant representation, z^*, is known a priori, the mean-squared error of approximating it is lower by more than an order of magnitude for the new layer (yellow in Fig. 3) compared to standard layer with SELU activation function (blue in Fig. 3), which

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_6

performed best compared to RELU and sigmoids. Notably, the best results with the new layer are achieved for relatively shallow networks, while for SELU-based layers a deep network is needed.

5 Learning Invariant Representations from Data

The experiments above assumed that we know what type of transformation – e.g., translation – is present in the input data, and thus we have a way of calculating the desired invariant representation and training the network in a supervised way. While this approach is useful in evaluating inherent ability of different architectures to capture invariance, it is far from our goal of learning invariance from data – if the desired transformation, for example DFT, is known a priori, there is no need for the network to learn to approximate it, it can be used directly as a pre-processing step.

Our goal is to show the network examples of input samples that are the same but have been transformed, and samples that are not the same. We want to train the network to discover what the invariant transformation is based on the above information alone, without defining the specific type of invariance upfront. To this end, the network is presented on input with triples $(x, x', d) \in \mathcal{X} \times \mathcal{X} \times \mathbb{R}_+$, where d is null if one sample is a transformed version of the other, and not null otherwise. We then train the auto-encoder $g(f(x))$, and we focus on part of the intermediate code $z = f(x)$, denoted $z_{inv}(x)$, to capture the invariant representation. We expect that given a triple (x, x', d), the intermediate codes $z_x = z_{inv}(x)$ and $z_{x'} = z_{inv}(x')$ for the two samples will have $\hat{d}(x, x') = \|z_x - z_{x'}\|$ similar to d. We are most interested in preserving small distances; thus, we use the inverse of squared Euclidean distance as the invariance-promoting term in the loss

$$\ell_{inv}(d, \hat{d}(x, x')) = \left(\frac{1}{\delta + \alpha d^2} - \frac{1}{\delta + \alpha \hat{d}(x, x')^2} \right)^2, \tag{2}$$

where δ and α are hyperparameters.

5.1 Experimental Validation on Translation Invariance

We conducted a series of experiments to validate the ability of the new layer to learn invariance from data.

Our first experiment involves learning translation invariance. We create a dataset in which samples come in pairs, the first sample is random as described before, and the second sample is a shifted version of the first sample, with the amount of shift selected randomly. For example, if we have a sample $1, 2, 3, 4, 5$, and we want to shift this sample by 2 positions, this sample is then changed to $4, 5, 1, 2, 3$. As the true distance, d^* we use the Euclidean distance between modulus of Fourier transforms of the samples; thus, we have null distance if two samples are shifted versions of each other, and positive distance otherwise.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_6

The results presented in Figure 4, left panel, show that in the absence of the true desired values of the intermediate code, and with access to pairwise distance data instead, the auto-encoder is still able to learn invariant representation equivalent to the Fourier transform of the input.

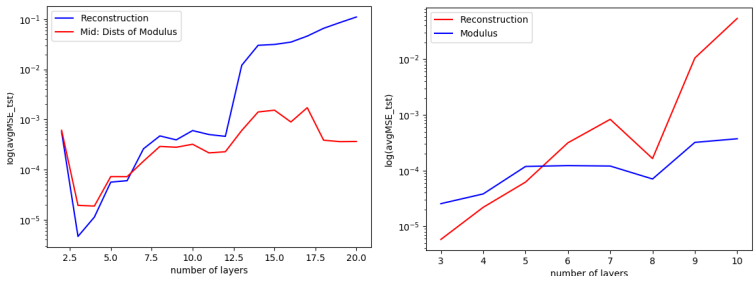


Fig. 4: MSE for auto-encoder reconstruction (red) and for learning invariant intermediate code (blue) for simple shift invariance (left) and for two-part shift invariance (right).

We also created a dataset in which each sample is composed of two parts, left and right, and a circular shift occurs independently within each part. The parts are of equal size, that is, if the sample has 10 features, each part consists of 5 features. If both parts of the sample are shifted version of another sample, the true distance d^* is null. We also create samples which are the same, concerning invariance, only in one part – those samples have $d^* > 0$ and allow us to detect if invariance for both parts is appropriately learned. The results in Figure 4, right panel, show that our architecture can successfully learn this type of invariance – the MSE is below 10^{-4}.

5.2 Experimental Validation beyond Translation Invariance

To move beyond simple shift-invariance, we tested invariance to an unknown set of permutations of dimensions. Specifically, prior to experiments with data of dimensionality d, we created a random cycle over a graph with d nodes, one per input dimension, and performed a cyclic shift of dimensions by a random number of steps along that cycle – the network should learn to be invariant to this set of permutations of data dimensions. As another example, we combined cyclic translation with multiplication of the input by a scalar. The results in Figure 5 show that both types of invariances are learned successfully.

The above experiments show that the proposed approach can be used to learn representations invariant to various transformations: shifting, scaling, and shuffling dimensions, and it can still achieve excellent performance even if the desired output of the invariant transformation is not known a priori, but has to be learned from examples.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_6

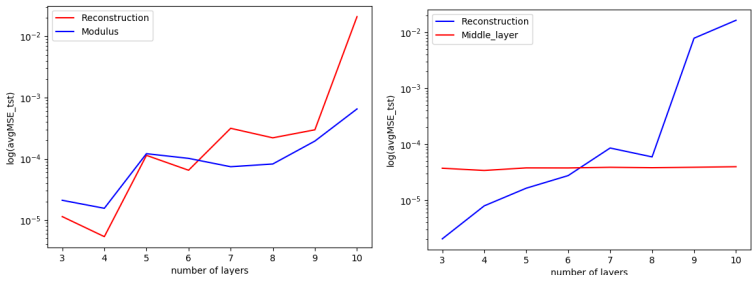


Fig. 5: MSE for auto-encoder reconstruction (red) and for learning invariant intermediate code (blue) for a set of fixed, unknown permutations of dimensions (left), and for translation and scaling (right).

6 Conclusions and Future Work

We focused on constructing neural networks that are invariant to transformations in the input samples. Instead of known, pre-defined type of invariance, we consider a more flexible scenario where invariance is learned from data. First, we showed that standard neural networks are poorly suited to capture invariance, leading to the need for approaches such as dataset augmentation with rotated, translated, or scaled versions of input images [2]. We then propose a new, richer layer, and show that it is more capable of learning invariance. We then show that the proposed new approach is effective in learning invariance form data, by utilizing information about which samples represent the same input subjected to some unknown transformation. These results open the avenue to creating neural networks that can be robust to various changes in the input – our future work will focus on exploring practical applications of this new type of networks. One possible application is in analyzing molecular profiles, such as gene expression, where several similar but different expression patterns can be functionally similar, for example if they represent utilization of two alternative biological pathways.

Acknowledgements

T.A. is supported by NSF grant IIS-1453658.

References

1. Arodź, T.: Invariant object recognition using radon-based transform. Computing and Informatics **24**(2), 183–199 (2012)
2. Benton, G., Finzi, M., Izmailov, P., Wilson, A.G.: Learning invariances in neural networks. In: Advances in Neural Information Processing Systems. vol. 33, pp. 17605–17616 (2020)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_6

3. Cheng, G., Han, J., Zhou, P., Xu, D.: Learning rotation-invariant and fisher discriminative convolutional neural networks for object detection. IEEE Transactions on Image Processing **28**(1), 265–278 (2018)
4. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems **2**(4), 303–314 (1989)
5. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys **46**(4), 1–37 (2014)
6. Haasdonk, B., Burkhardt, H.: Invariant kernel functions for pattern analysis and machine learning. Machine learning **68**(1), 35–61 (2007)
7. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural networks **4**(2), 251–257 (1991)
8. Khotanzad, A., Hong, Y.H.: Invariant image recognition by zernike moments. IEEE Transactions on pattern analysis and machine intelligence **12**(5), 489–497 (1990)
9. Krawczyk, B.: Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence **5**(4), 221–232 (2016)
10. Lai, J.H., Yuen, P.C., Feng, G.C.: Face recognition using holistic fourier invariant features. Pattern Recognition **34**(1), 95–109 (2001)
11. Laptev, D., Savinov, N., Buhmann, J.M., Pollefeys, M.: Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 289–297 (2016)
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
13. Lenc, K., Vedaldi, A.: Understanding image representations by measuring their equivariance and equivalence. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 991–999 (2015)
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems. pp. 3111–3119 (2013)
15. Panahi, A., Saeedi, S., Arodz, T.: word2ket: Space-efficient word embeddings inspired by quantum entanglement. In: International Conference on Learning Representations (2019)
16. Pennington, J., Socher, R., Manning, C.: GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. pp. 1532–1543 (2014)
17. Qi, G.J., Zhang, L., Lin, F., Wang, X.: Learning generalized transformation equivariant representations via autoencoding transformations. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)
18. Shen, X., Tian, X., He, A., Sun, S., Tao, D.: Transform-invariant convolutional neural networks for image classification and search. In: Proceedings of the 24th ACM international conference on Multimedia. pp. 1345–1354 (2016)
19. Wood, J.: Invariant pattern recognition: a review. Pattern recognition **29**(1), 1–17 (1996)
20. Zhao, H., Des Combes, R.T., Zhang, K., Gordon, G.: On learning invariant representations for domain adaptation. In: International Conference on Machine Learning. pp. 7523–7532. PMLR (2019)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_6