Neural Networks 133 (2021) 103–111

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Unsupervised feature learning for self-tuning neural networks

Jongbin Ryu [a,b], Ming-Hsuan Yang [d], Jongwoo Lim [c,*]

[a] Department of Computer Engineering, Ajou University, Republic of Korea
[b] Department of Artificial Intelligence, Ajou University, Republic of Korea
[c] Department of Computer Science, Hanyang University, Republic of Korea
[d] School of Engineering, University of California, Merced, United States of America

ARTICLE INFO

Article history:
Received 9 March 2020
Received in revised form 28 August 2020
Accepted 16 October 2020
Available online 22 October 2020

Keywords:
Self-tuning neural network
Unsupervised feature learning
Unsupervised transfer learning
Bagged clustering
Ranking violation for triplet sampling

ABSTRACT

In recent years transfer learning has attracted much attention due to its ability to adapt a well-trained model from one domain to another. Fine-tuning is one of the most widely-used methods which exploit a small set of labeled data in the target domain for adapting the network. Including a few methods using the labeled data in the source domain, most transfer learning methods require labeled datasets, and it restricts the use of transfer learning to new domains. In this paper, we propose a fully unsupervised self-tuning algorithm for learning visual features in different domains. The proposed method updates a pre-trained model by minimizing the triplet loss function using only unlabeled data in the target domain. First, we propose the relevance measure for unlabeled data by the bagged clustering method. Then triplets of the anchor, positive, and negative data points are sampled based on the ranking violations of the relevance scores and the Euclidean distances in the embedded feature space. This fully unsupervised self-tuning algorithm improves the performance of the network significantly. We extensively evaluate the proposed algorithm using various metrics, including classification accuracy, feature analysis, and clustering quality, on five benchmark datasets in different domains. Besides, we demonstrate that applying the self-tuning method on the fine-tuned network help achieve better results.

1. Introduction

In recent years, convolutional neural networks (CNNs) have been applied to numerous learning and vision problems with state-of-the-art performance. Among the various tasks, the networks for large-scale image classification are designed and trained thoroughly with a vast amount of labeled data to achieve the best performance. As these networks have learned generic and effective feature representations, there have been attempts to transfer them to different tasks, e.g., object detection, recognition, and tracking. Fine-tuning is one of the most widely-used methods for domain transfer. When fine-tuning a classification network, the labeled training data in the target domain are used to update the pre-trained network. Although the size of the training data used in fine-tuning is significantly less than that for training from scratch, still a significant amount of labeled data in the target domain is needed for high performance. In general, the performance of a fine-tuned network model for a new domain hinges on whether a training dataset with accurate labeling is available or not.

Constructing a large set of training data requires a significant amount of effort, and the labeling tasks are merely daunting. Besides, human error and bias can be introduced while collecting and labeling the data and, it becomes one of the reasons for poor performance. Thus recent efforts have been focused on reducing or eliminating the need for labeling training data (Ganin & Lempitsky, 2015; Long et al., 2016; Yang et al., 2020; Zhou et al., 2017). The above-mentioned issues are exacerbated for applying deep neural network models to a new application domain when there is no established training dataset or lack of expert domain knowledge. Therefore it is of great interest if a pre-trained network can be transferred to a new domain without using any training data or relying solely on unlabeled data.

In this paper, we propose a fully unsupervised self-tuning algorithm for the transfer learning of neural networks as Fig. 1. The algorithm starts with a pre-trained network in a source domain and a set of unlabeled training data in the target domain. Intuitively the distance in the embedded feature space by the network should better reflect the relevance of data points. In a supervised setup, the data labels are used to determine relevance. Data points sharing same class labels are considered as positive samples; otherwise, they are sampled by negatives with an anchor. The Siamese distance metric or the triplet loss updates the network to pull or push the training samples accordingly.

* Corresponding author.
E-mail addresses: jongbinryu@ajou.ac.kr (J. Ryu), mhyang@ucmerced.edu (M.-H. Yang), jlim@hanyang.ac.kr (J. Lim).

https://doi.org/10.1016/j.neunet.2020.10.011
0893-6080/

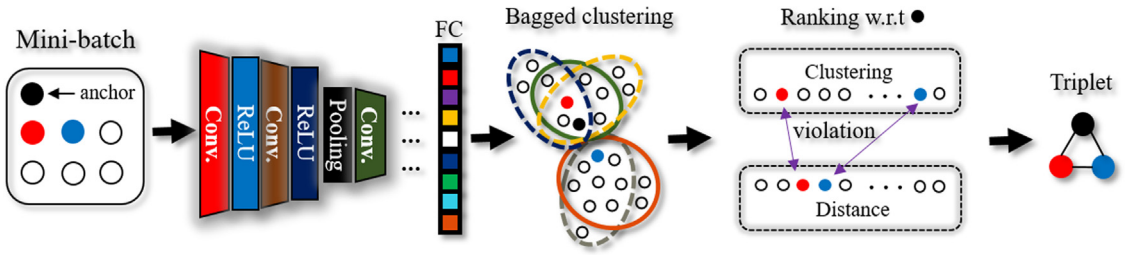


Fig. 1. Main steps of the proposed self-tuning method. We first feedforward mini-batch samples to obtain FC features. An ensemble of clusters is constructed on the FC features to measure the relevance score. Two rankings by the relevance scores and Euclidean distances on the FC features are computed for an anchor sample. The violation of the rankings is used to selects positive and negative samples to an anchor. Finally, the neural network is self-tuned by the gradient of the triplets.

In our fully unsupervised self-tuning method, the first intuition is that the relevance between unlabeled data can be inferred from the probability of whether the data points belong to the same cluster or not. For this relevance measurement, we use the unsupervised clustering algorithms with K-means (Duda et al., 2000) in randomly-selected subspaces or the random projection trees (RPT) (Dasgupta & Freund, 2008). The relevance estimation of the unlabeled training data is the most important part of our algorithm.

Even though the approximate relevance measure is available, tuning the network with this weak information is very challenging. It is desirable for the Euclidean distance in the feature space to reflect the relevance between data points, and our second intuition is to use the ranking inconsistency of the relevance and feature distance between data. The proposed algorithm selects the positive and negative samples whose distance rankings are significantly different from the relevance rankings. Then networks are updated via the sampled triplets without any supervision.

The main contributions of this work are as follows.

(i) We propose a novel fully-unsupervised self-tuning algorithm for neural networks. The triplet sampling algorithm using the ranking inconsistency in the Euclidean distance, and the relevance measure performs well with the unlabeled data. The relevance model by bagged clustering in random subspaces of the feature space is novel and effective.

(ii) We carry out extensive experimental evaluations to verify the proposed self-tuning algorithm. Five pre-trained CNN models and five different domains are used, and the performance is evaluated with various classification and clustering metrics. Experimental results demonstrate that our algorithm shows significant improvement, considering that it operates in a fully unsupervised manner.

(iii) We show that self-tuning on the fine-tuned neural networks also improves the performance significantly. As most domain adaptation is carried out by fine-tuning, it is interesting that the performance of a fine-tuned network can be further enhanced by self-tuning with the same training data without the labels. From the fact that the fine-tuned network also contains notable ranking inconsistency, we notice that what self-tuning learns from unlabeled data is different from the conventional supervised algorithms.

2. Related work

We discuss the transfer learning algorithms, unsupervised feature learning and pseudo-labeling based metric learning as the relevant works of the proposed method.

2.1. Transfer learning

Transferring knowledge from source to specific target domains where the distribution of data to the source differs significantly is a very challenging problem. In many cases, the overfitting problem occurs due to the different data distribution of the source and target. Adversarial learning (Ganin & Lempitsky, 2015; Liu et al., 2019), kernel (Tao et al., 2016), and ensemble based methods (Ryu et al., 2020a, 2020b) have been introduced to overcome the overfitting problem. Most of them suppose that we have labeled data for the source domain and unlabeled data for the target domain. The main issue for this setting is reducing the domain shift using both domain data. Although, we does not have labels of the target domain, we can label them as 'target' for the domain label; of course, the source data should be labeled as 'source'. From this domain label, the source and target domain distribution can be aligned. On the other hand, training neural networks from a few samples is also widely studied by the few-shot learning algorithm. Since the scarcity of the training data often causes the overfitting problem also, we need to prepare additional data source or transfer knowledge from another model. Augmenting training data (Antoniou et al., 2017; Hariharan & Girshick, 2017) and knowledge distillation (Bhardwaj et al., 2019; Yu et al., 2017) are effectively applied to escape the overfitting problem under the few-shot learning setup.

Recently, domain generalization, which uses multiple source domains without any target domain data, is also studied in several works (Balaji et al., 2018; Ryu et al., 2020b). It learns a model that reduces the domain-shift between multiple source domains, so that the domain-shift is minimized with any unseen domain not used for the learning. Thus, the model generalizes the classification task regardless of the domains.

2.2. Unsupervised feature learning

Learning robust features is a difficult task in an unsupervised environment. The most serious difficulty is to define the objective function without true labels. However, these methods (Bo et al., 2013; Boureau et al., 2011; Dosovitskiy et al., 2016; Zou et al., 2012) are different from the proposed approach for two reasons. First, they use the class labels to train a classifier. Although they improve the robustness of training features without the class labels, their methods are evaluated on a classifier trained by the ground-truth class labels. Second, most works focus on supervised visual recognition tasks, specified by a certain network architecture. However, we do not use class labels to train a network or classifier to measure classification accuracy using K-nearest neighbors. Further, we evaluate the proposed self-tuning algorithm by four visual domains with five network architecture. We think that this extensive experiment validates the generalized performance of the proposed self-tuning algorithm.

The fully unsupervised method is very important in real-world application scenarios. On the web and cloud, there exist millions

of data, but they cannot be directly utilized in training neural networks due to the lack of labels. Fully-unsupervised algorithms enable the networks to be autonomously updated and improved using the massive unlabeled data. It tries to find the effective features in the pre-trained base network for the new data in a different domain, and update the network to enhance them as well as learn new features. Since there is no classification label available, we focus on the feature space so that similar data are mapped closer, and different ones are dispersed apart. To best of our knowledge, the proposed self-tuning method is the first attempt in this direction. Without any information on the target domain, we propose using the ranking inconsistency between feature distance and clustering results, and experimentally show that this idea is effective in the extensive experiments.

3. Proposed algorithm

In this section, we describe the proposed self-tuning algorithm on the pre-trained neural networks using unlabeled data in the target domain. The proposed self-tuning algorithm uses the triplet loss function similar to Chechik et al. (2010), Ryu et al. (2020b), Schroff et al. (2015) and Wang et al. (2014a). The main idea of triplet loss is to keep the distance between samples of the same class close and the distance between samples of different classes far. For an anchor, samples of the same class become positive, and samples of the other class become negative. In a supervised setting, it is easy to define triplets using class label information, but in our fully-unsupervised setting, a well-designed selection algorithm is required.

In our algorithm, for a given (unlabeled) training set T, a set of triplet X is defined as:

$$X = \{\langle x_a, x_+, x_-\rangle\} \text{ such that } \rho\left(x_a, x_+\right) > \rho\left(x_a, x_-\right), \tag{1}$$

where x_a, x_+ and $x_- \in T$ are the anchor, positive, and negative samples respectively, and $\rho(\cdot)$ denotes a relevance score function. We denote the distance in the feature space as

$$D(x_i, x_j) = \|f(x_i) - f(x_j)\|_2^2, \tag{2}$$

where $f(x)$ is the feature descriptor output by the neural network and x_i, x_j are data points. The training process aims to minimize the distance of a positive pair $\langle x_a, x_+\rangle$, and at the same time to maximize that of a negative pair $\langle x_a, x_-\rangle$. Naturally, the loss function for the triplet set is defined as:

$$\mathcal{L} = \sum_{\langle x_a, x_+, x_-\rangle \in X} \max\left\{0,\ \alpha - D\left(x_a, x_-\right) + D\left(x_a, x_+\right)\right\}, \tag{3}$$

where α denotes a margin constraint that is a user input hyper-parameter. Minimizing the above loss function pushes the more relevant data x_+ closer to the anchor and, at the same time, the less relevant data x_- farther from it in the feature space. Thus the main goal is to build an adequate triplet set from the unlabeled data to transfer the pre-trained neural network $f(\cdot)$ to the target domain.

Existing transfer learning or domain adaptation methods mainly use class labels of the training examples to determine data relevance (Chechik et al., 2010; Schroff et al., 2015; Wang et al., 2014b). The positive samples are those with the same label as the anchor, and otherwise negative. In the supervised or semi-supervised settings where the labels of training data are available, this is a natural and effective way to construct the triplet set. However, in a fully unsupervised scenario, such label-based relevance functions cannot be used as the labels of training data.

Therefore, we propose the clustering-based relevance model for the fully unsupervised network learning. This approach is motivated by the pseudo-label based unsupervised learning models (Liang et al., 2018, 2019). They propose pseudo-label generation methods to learn networks without label supervision. Liang et al. create pseudo labels of target domain data with unsupervised random projection. They use the property that labels belonging to the same cluster have a high probability of having the same class. Inspired by this property, we propose the relevance model using an unsupervised clustering method. Suppose that the data points are clustered into several groups according to different criteria. If two points are more relevant than the others, they are more likely and more often to be in the same clusters. In our formulation, if two data points are closely related, repeated clustering with random feature selection should reflect such relevance. Samples with the same class label will frequently belong to the same cluster, and the distance between their features should be close. Therefore, we reduce the inconsistency between the relevance score and euclidean distance by a ranking model. For a reference data point, the rankings according to the feature distance and the relevance score of other data points are computed. When a network is not tuned to the target domain, there will be many inconsistencies in the two rankings; thus we can tune the network to reduce the inconsistency. Then the sampled triplets are used for updating the network to move the positive data toward the anchor and the negative data away from the anchor, as carried out in the conventional triplet loss algorithms (Fig. 2).

3.1. Unsupervised relevance model

To design the relevance score function, we exploit the data clustering algorithms which are well-established unsupervised learning methods for categorizing data according to their affinity. If two data are relevant to each other, they are likely to appear in the same cluster, and thus the clustering results can be used as a measure of relevance. However, it is not appropriate to apply it directly to the feature output of the training data since the feature space of the pre-trained network is tuned to the relevance in the source domain, rather than the target domain where the training data belongs to.

If the source domain is completely different from the target domain, it is not realistic to expect the pre-trained feature space to work in the target domain. However, the deep neural networks for large-scale object classification are constructed using a precisely-labeled huge training data on very diverse categories, and the features obtained from these networks have been used as the bases in various fine-tuning and transfer learning algorithms. What the proposed algorithm tries to do is tuning the generic feature embedding of the base network so that its discriminative power in the target domain is enhanced in a fully unsupervised way.

We formulate the relevance metric as the expected probability whether two data points belong to the same clusters via the ensemble of clustering methods, i.e.,

$$\begin{aligned}\rho(x_i, x_j) &\propto Pr\left(\mathcal{C}(f(x_i)) == \mathcal{C}(f(x_j))\right)\\ &\simeq E_k\left[\mathbb{1}(\mathcal{C}_k(f(x_i)) == \mathcal{C}_k(f(x_j)))\right],\end{aligned} \tag{4}$$

where $\mathcal{C}(\cdot)$ returns the cluster label, $\mathbb{1}(\cdot)$ is an indicator function, and E_k denotes expected value of k clusters. If a data pair is clustered together more often by different clustering algorithms, it can be considered more relevant than other data pairs. However, if the clustering is fully determined by the embedded feature vectors, no additional information is discovered by clustering, and there will be no update to the network. We verify this claim by the experiment using the K-means with data bootstrapping, where the affinity is measured by the Euclidean distance in the original feature space. Different clustering results are generated

Fig. 2. Visualization of FC features of the Flower102 dataset using t-SNE (Maaten & Hinton, 2008). It shows that the self-tuned result is much improved over the pre-trained and comparable to the fine-tuned result in terms of the discriminative ability.

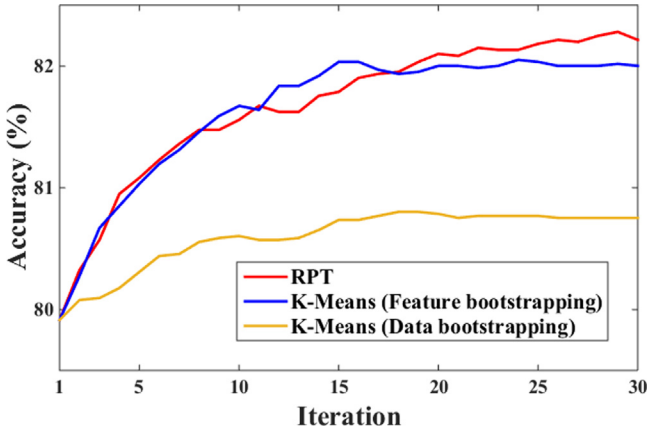


Fig. 3. Performance comparison of the three relevance models by clustering: K-means with data or feature bootstrapping and RPT. It shows K-means with data bootstrapping saturates prematurely compared to the other two methods. The reason can be because data bootstrapping uses the same feature space as the pre-trained network, whereas the others use differently weighted features.

by running it on randomly-selected subsets of data, and the co-incidence frequency is counted. As expected, it quickly converges to a local minimum without significant improvement, as shown in Fig. 3.

To avoid this problem, we propose two bagged clustering methods: K-means with feature bootstrapping and random projection tree (RPT).

(i) **K-means with feature bootstrapping:** Instead of randomly sampling the data for cluster bagging, the subspaces of the feature space are randomly sampled, and the K-means clustering is performed in the sampled subspace. By random projections, different subsets of features are used in clustering, and the subsets effective in the target domain will generate informative clustering results. Combined with the proposed triplet sampling method Section 3.2, this weak information is effectively utilized for self-tuning .

(ii) **Random projection tree:** The RPT clustering constructs a random projection tree, and groups samples in the same leaf node as one cluster. Each non-leaf node n of a RPT has a randomly sampled vector $\mathbf{w}_n$ and a threshold λ_n, and the data falls to the left branch if $f(x) \cdot \mathbf{w}_n < \lambda_n$ or to the right branch otherwise. In RPT, the set of random projection vectors $\{\mathbf{w}_n\}$ plays a similar role as the random feature subspaces in the K-means algorithm with feature bootstrapping.

3.2. Triplet sampling using rank inconsistency

As we discuss in the previous section, the relevance between data pairs can be estimated using a clustering algorithm with random feature selection. However, finding the positive and negative pairs using the relevance scores is not straightforward since it only provides very weak information on data similarity, and simplistic approaches such as thresholding may not find a good training set.

We propose to use the ranking inconsistency in finding the positive and negative sample to build a triplet for training. For a randomly selected data point (the anchor) in the mini-batch, other data can be sorted according to the two metrics: the Euclidean distance in the feature space and the relevance score by clustering coincidence. The Euclidean feature distance represents the data similarity in the *current* feature space, and the relevance score indicates the *desired* distance ordering with respect to the anchor. At the moment, all data except the anchor have two ranks, one in ascending order of Euclidean distance (lower rank = closer), and the other in descending order of relevance score (lower rank = more relevant). If the two ranks are the same for all data, it is in the optimal state since the Euclidean feature distance matches with the relevance. If the Euclidean distance rank of a data is lower than its relevance rank, it means that the data is located closer than desired; thus it needs to be pushed away from the anchor (a negative sample), and vice versa. Hence for anchor data, we can choose positive and negative samples by using the rank inconsistency.

For fast and effective learning, it is important to select the hard negative and hard positive examples. We start with finding the hardest negative example for an anchor:

$$x_- = \underset{x}{\operatorname{argmin}} \left\{ \sigma_{D\uparrow}(x;\, x_a) - \sigma_{\rho\downarrow}(x;\, x_a) \right\}, \tag{5}$$

where $\sigma_m(s; x_a)$ denotes the index of s in the sorted list of the mini-batch by ascending($\uparrow$) or descending($\downarrow$) order in terms of the metric $m \in \{D, \rho\}$ w.r.t the anchor x_a. Thus, $\sigma_{D\uparrow}(x;\, x_a)$ denotes the index of sorted list for the distance metric D in ascending order and $\sigma_{\rho\downarrow}(x;\, x_a)$ represents that for the relevance score ρ in descending order. Then, we select a negative sample x_- using the gap between these two criteria.

Note that the data with a negative ranking gap in the above equation can be negative examples, and a positive ranking gap means positive candidates. Therefore, x_- is the one whose ranking gap is negatively largest when the data is sorted according to the relevance and distance with respect to the anchor x_a.

Next, a hard positive sample is selected. We note if the hardest positive and hardest negative data are sampled at the same time, the learning is likely stuck to local minima at the early stage.

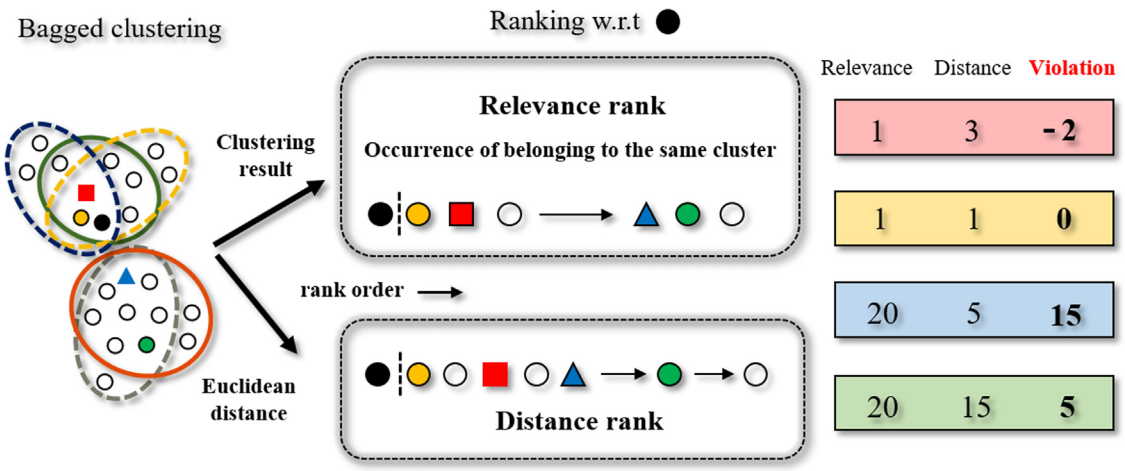


Fig. 4. Example of the triplet selection process. Data points are ranked on a randomly-selected anchor (black) by the relevance scores and Euclidean distances. The blue data point never belongs to the same cluster with the anchor while they are close to each other in Euclidean space. Since the ranking violation of the blue one is the largest in the mini-batch, it is sampled as the negative. The yellow data point, however, has a minimum ranking violation, but cannot be selected as positive because its distance is farther than the negative sample. Instead, the red one is selected as positive because it has the smallest ranking violation among the samples satisfying the distance condition. For more details refer to the text in Section 3.2.

Therefore, we select a positive sample of the hard positives that satisfy the following condition:

$$x_+ = \underset{x}{\operatorname{argmax}} \left\{ \sigma_{D\uparrow}(x;\, x_a) - \sigma_{\rho\downarrow}(x;\, x_a) \right\}, \quad \text{subject to } D(x,\, x_a) < D(x_-,\, x_a). \tag{6}$$

where x_+ denotes the positive sample. We constrain the distance between the positive and anchor to be closer than the distance between negative and anchor. This constraint ensures stable learning of the network, which can also be found in a method to prevent model collapse in Schroff et al. (2015).

In practice, it is similar to the supervised triplet sampling approaches (Schroff et al., 2015; Wang et al., 2014a). The sampled triplet $\langle x_a, x_+, x_- \rangle$ are used to self-tune a neural network. Fig. 4 illustrates this process with examples.

The self-tuning algorithm removes the last FC layer and the softmax layer in the pre-trained models and connects the proposed triplet loss layer at the end. All layers in the network are updated by backpropagation until convergence using the automatically generated triplets from the unlabeled training data according to the proposed sampling algorithm.

4. Experimental results

4.1. Experimental setting

We evaluate the proposed self-tuning algorithm on the five datasets in different visual domains: generic objects (Caltech 101) (Fei-Fei et al., 2007), scenes (MIT Indoor) (Quattoni & Torralba, 2009), flowers (Flower 102) (Nilsback & Zisserman, 2008), and materials (FMD) (Sharan et al., 2009). We use the standard evaluation protocols for these datasets, including splitting the training and test sets.

For the pre-trained neural network models we use the following five networks: AlexNet (Krizhevsky et al., 2012), VGG-S (Chatfield et al., 2014), VGGVD-19 (Simonyan & Zisserman, 2014), ResNet-50 and ResNet-101 (He et al., 2016). We obtain features from the last FC layer for both pre-trained and self-tuned networks. The FC feature dimensions are 4096 for AlexNet, VGG-S, and VGGVD-19, and 2048 for ResNet-50 and ResNet-101. We do not apply any post-processing to the features.

4.2. Evaluation metrics

To evaluate the proposed algorithm, we perform extensive experiments on three tasks: classification, feature representation, and clustering. In all experiments, the class labels in the datasets are only used for evaluation purposes. The first task, classification performance, is measured by K-nearest neighbors (KNN) with $K = 1$. The classification performance is the most fundamental and important metric to evaluate learning algorithms, and thus we use the KNN, which does not need class labels in the training stage. Second, we use Fisher score and variance for the evaluation of feature representations learned by the proposed algorithm and others. Since it is known that the features with higher variance are more salient in the feature selection method (Guyon & Elisseeff, 2003), we also use the feature variances. Third, to measure the clustering performance by the learned features of the algorithms, we use the average purity, precision, and recall values after running the K-means clustering algorithm. The clustering task is also widely used to evaluate the robustness of the feature learning algorithm. Some of the evaluation metrics are described in more detail below.

(i) **Fisher score:** The discriminative power of a feature space is measured by the FisherScore $= \frac{\sum_k (\mu - \mu_k)^2}{\sum_k \sigma_k^2}$, where μ is the sample mean, and μ_k and σ_k^2 are the mean and variance of the kth cluster.

(ii) **Purity**: When each cluster is assigned with the label by the majority, the purity is the ratio of the correctly clustered data (i.e., data with the label same as the cluster label). The maximum value of purity is 1 when the clustering is perfect. That is, Purity $= \frac{1}{N} \sum_k \max_{l \in L} \sum_{s \in C_k} \mathbb{1}(\lambda(s) = l)$ where N is the number of all samples, C_k is the kth cluster, $\lambda(s)$ is the label of a sample s, and L is the set of all labels.

(iii) **Precision and recall:** In the context of clustering evaluation, all pairs of test data are checked, and the term positive/negative represents if a data pair belongs to the same cluster or not. Thus among the pairs in the same clusters, true positives (TP) are those with the same labels, and false positives (FP) are those with different labels. True negatives (TN) and false negatives (FN) can be defined accordingly for the pairs in different clusters. The precision and recall are defined as Precision $= \frac{\mathrm{TP}}{\mathrm{TP+FP}}$ and Recall $= \frac{\mathrm{TP}}{\mathrm{TP+FN}}$. In the experiment, we fix the number of clusters to 50.

Table 1
Evaluation results on the proposed self-tuning algorithm (Self) over the baseline pre-trained model (Pre) for classification accuracy and feature evaluation. KNN is used to measure classification accuracy, which shows significant improvement of Self in all cases. Feature representation evaluation by Fisher score and variance also shows that Self enhances the discriminative power (Fisher score) and transforms features more salient (variance). Overall the results demonstrate the effectiveness of domain adaptation by the self-tuning algorithm.

Net	Dataset	KNN (K = 1)		Fisher score		Variance	
		Pre	Self	Pre	Self	Pre	Slef
AlexNet	Caltech 101	78.12	**80.08**	0.40	**0.48**	0.43	**1.13**
	MIT Indoor	33.58	**36.04**	0.29	**0.34**	0.65	**1.30**
	Flower 102	63.95	**67.18**	0.37	**0.42**	2.03	**2.98**
	FMD	47.60	**49.40**	0.18	**0.19**	0.54	**0.61**
VGG-S	Caltech 101	79.91	**82.10**	0.41	**0.46**	0.20	**0.23**
	MIT Indoor	43.28	**43.28**	0.34	**0.38**	0.37	**0.75**
	Flower 102	65.21	**70.92**	0.39	**0.43**	0.84	**1.58**
	FMD	52.40	**55.60**	0.20	**0.21**	0.25	**0.29**
VGGVD-19	Caltech 101	79.06	**81.36**	0.41	**0.51**	0.23	**0.47**
	MIT Indoor	46.27	**49.55**	0.40	**0.43**	0.58	**0.75**
	Flower 102	55.31	**67.88**	0.38	**0.46**	1.40	**1.97**
	FMD	62.20	**66.00**	0.22	**0.22**	0.30	**0.34**
ResNet-50	Caltech101	84.29	**87.10**	0.45	**0.52**	0.10	**0.11**
	MIT Indoor	56.34	**58.81**	0.37	**0.38**	**0.08**	0.08
	Flower 102	70.87	**77.22**	0.41	**0.47**	**0.12**	0.09
	FMD	69.00	**72.00**	0.24	**0.24**	0.14	**0.14**
ResNet-101	Caltech101	85.16	**87.64**	0.45	**0.54**	0.12	**0.13**
	MIT Indoor	58.73	**60.37**	0.38	**0.39**	**0.09**	0.08
	Flower 102	70.16	**77.09**	0.42	**0.48**	**0.13**	0.09
	FMD	69.20	**72.00**	0.24	**0.24**	0.15	**0.16**

Table 2
Evaluation results on the proposed self-tuning (Self) algorithm over the baseline pre-trained model (Pre) for clustering analysis. Self-tuning improves the performance of the pre-trained model for purity in most cases. Precision rates also increase by the self-tuning while recall rates remain similar compared to the pre-trained model. This result also supports the self-tuning transforms the feature space of the pre-trained model into better representation.

Net	Dataset	Purity		Precision		Recall	
		Pre	Self	Pre	Self	Pre	Self
AlexNet	Caltech 101	0.64	**0.66**	0.60	**0.71**	**0.80**	0.74
	MIT Indoor	0.32	**0.32**	0.12	**0.13**	**0.19**	0.18
	Flower 102	0.39	**0.41**	0.19	**0.23**	**0.36**	0.34
	FMD	0.44	**0.45**	0.21	**0.22**	**0.16**	0.14
VGG-S	Caltech 101	0.66	**0.67**	0.67	**0.68**	**0.82**	0.79
	MIT Indoor	0.37	**0.38**	0.16	**0.17**	**0.24**	0.23
	Flower 102	0.38	**0.42**	0.19	**0.24**	0.34	**0.36**
	FMD	0.51	**0.54**	0.27	**0.32**	0.12	**0.14**
VGGVD-19	Caltech 101	0.64	**0.67**	0.57	**0.67**	**0.85**	0.84
	MIT Indoor	0.42	**0.42**	0.20	**0.21**	**0.30**	0.29
	Flower 102	0.32	**0.39**	0.14	**0.21**	0.28	**0.32**
	FMD	0.58	**0.59**	**0.33**	0.32	**0.23**	0.22
ResNet-50	Caltech101	0.68	**0.71**	0.67	**0.73**	0.84	**0.87**
	MIT Indoor	0.44	**0.46**	0.22	**0.25**	0.32	**0.33**
	Flower 102	0.40	**0.45**	0.21	**0.30**	0.37	**0.44**
	FMD	0.65	**0.67**	0.42	**0.50**	**0.21**	0.16
ResNet-101	Caltech101	0.66	**0.72**	0.58	**0.73**	0.82	**0.87**
	MIT Indoor	0.46	0.46	0.24	**0.25**	**0.35**	0.34
	Flower 102	0.40	**0.46**	0.21	**0.28**	0.37	**0.43**
	FMD	0.65	**0.67**	0.41	**0.53**	**0.20**	0.17

4.3. Self-tuning on pre-trained model

Tables 1 and 2 summarize the comprehensive experimental evaluations of the proposed self-tuning method. Compared to the baseline pre-trained model, the self-tuning method improves most of the classification, feature representation, and clustering performance measures significantly. When the average value of all datasets is obtained from ResNet-101, it shows that self-tuning has better performance in all cases than the pre-trained network, as shown in Table 3.

In classification tasks, the average improvement of KNN ranges from 1.94% to 4.40% across all five networks, and the purity and precision values in clustering analysis are significantly improved while the recall remains similar. In addition, the Fisher scores and variances show the enhancement in feature representation quality by the proposed algorithm. Compared to AlexNet and VGG, ResNet networks mark slightly lower variances than the original network, but the difference is not large.

In clustering evaluation, while the purity and precision show impressive boosting, the recall remains the same or decreases a little. According to the definition of recall in clustering, making more positives tends to yield higher recall, and when the cluster sizes are unbalanced the number of positives increase (in the extreme, recall = 1 if all data is in one cluster). As self-tuning refines the feature space, the clustering results improve, and the clusters will have similar sizes as the training dataset is built in such way. The slight decrease in the recall can be contributed to the above reason.

4.4. Self-tuning on mixed datasets

The four datasets we used in the experiments are constructed and carefully labeled. Although the labels are not used in training, the data quality is very high. For the unsupervised, the data may be noisy, corrupted, or even in completely different domains (outliers). To simulate such cases and to evaluate the generalization ability, the self-tuning algorithm is run on the three networks with various combinations of datasets. Table 5 shows the result of self-tuning using mixtures of datasets. This is a challenging task as the datasets are very diverse, from flowers to materials.

In this challenging scenario, the self-tuning successfully learns the heterogeneous domains in a fully unsupervised way. Classification and feature representation performances are significantly improved, except purity becomes slightly worse in AlexNet and VGG-S. This experiment shows that the proposed self-tuning algorithm is a generic framework that is quite robust to domain and variations.

4.5. Self-tuning on fine-tuned networks

Fine-tuning is a supervised learning algorithm that updates the pre-trained network using the labeled target domain data. It has been the state-of-the-art and used in many transfer learning applications. We present experimental results of comparing the proposed self-tuning algorithm with the supervised fine-tuning method and running the self-tuning on the fine-tuned network. Compared to the pre-trained and self-tuning results Tables 1 and 2, the fine-tuning results in Table 4 shows much improved performance since the fine-tuning uses labeled training data. The self-tuning results are in the middle of the pre-trained and the fine-tuned results, which is remarkable, considering that it is a fully unsupervised method.

More interestingly running self-tuning on the fine-tuned networks also boosts the performance as shown in Table 4 and Fig. 5. By simply running the self-tuning with the same dataset without the labels on the fine-tuned network all classification, feature representation, and clustering performance are considerably improved. This is quite surprising that we do not use any additional data (in fact, less information since labels are not used) to self-tune the network. We interpret this result as the self-tuning process reduces the overfitting in the fine-tuned networks by examining various combinations of learned features and fortifying more robust feature subsets. Thus, we believe the proposed self-tuning method is unique and different from the conventional learning methods, and it contributes to the field significantly.

Table 3
Evaluation results of the average performance of the classification, feature representation and clustering problems. It shows that the proposed self-tuning algorithm (Self) improves the performance of the pre-trained models (Pre) in all cases.

Classification		Feature representation				Clustering problem					
KNN (K = 1)		Fisher score		Variance		Purity		Precision		Recall	
Pre	Self	Pre	Self	Pre	Self	Pre	Self	Pre	Self	Pre	Self
70.81	**74.28**	0.37	**0.41**	0.12	**0.12**	0.54	**0.58**	0.36	**0.45**	0.44	**0.45**

Table 4
Performance comparison of the fine-tuning (Fine) and the self-tuning on the fine-tuned networks (F-A) of ResNet-101. Fine-tuning marks much improved performance compared to the pre-trained and self-tuning as it uses training labels. More interestingly running the self-tuning algorithm on the fine-tuned networks shows considerable enhancement over the fine-tuning. We think that the self-tuning can reduce the overfitting induced by supervised methods, and thus the self-tuning operates quite differently than the conventional learning methods.

Dataset	KNN(N = 1)		Fisher score		Purity		Precision		Recall	
	Fine	F-A	Fine	F-A	Fine	F-A	Fine	F-A	Fine	F-A
Caltech 101	91.26	**92.39**	0.65	**0.70**	0.72	**0.75**	0.65	**0.71**	0.89	**0.91**
MIT Indoor	70.22	**70.60**	0.41	**0.42**	0.59	**0.62**	0.36	**0.38**	**0.50**	0.49
Flower 102	86.29	**87.14**	0.51	**0.55**	0.56	**0.57**	0.37	**0.37**	**0.62**	0.61
Average	82.59	**83.38**	0.52	**0.56**	0.62	**0.65**	0.46	**0.49**	0.67	**0.67**

Table 5
Experimental results on the heterogeneous dataset combination. To show the generalization ability of the proposed algorithm, the self-tuning method is performed with synthetically generated datasets on heterogeneous domains. The synthetic datasets are constructed by simply merging several existing datasets: C (Caltech 101), M (MIT Indoor), Fl (Flower 102), and FM (FMD). Even with mixed and inconsistent training data, the self-tuning improves the pre-trained network.

Net	Dataset				KNN (K = 1)		Fisher score		Purity	
	C	M	Fl	FM	Pre	Self	Pre	Self	Pre	Self
AlexNet	✓	✓			68.10	**69.96**	0.36	**0.39**	**0.49**	0.48
		✓	✓		60.03	**61.22**	0.26	**0.30**	**0.24**	0.23
		✓	✓	✓	58.29	**59.82**	0.27	**0.30**	**0.43**	0.41
VGG-S	✓	✓			72.35	**73.57**	0.40	**0.41**	**0.50**	0.48
		✓	✓		62.01	**63.92**	0.31	**0.33**	**0.24**	0.23
		✓	✓	✓	60.66	**62.30**	0.31	**0.33**	**0.45**	0.44
ResNet-101	✓	✓			79.74	**80.59**	0.45	**0.47**	0.44	**0.54**
		✓	✓		69.76	**71.79**	0.38	**0.39**	0.21	**0.23**
		✓	✓	✓	69.07	**70.61**	0.38	**0.39**	0.41	**0.43**

Table 6
Performance comparison of the self-tuning with previous unsupervised feature learning methods on Caltech-101. The self-tuned network of ResNet-101 achieves favorable results compared to previous state-of-the-art methods.

Previous method	Accuracy
Multiway local pooling (Boureau et al., 2011)	77.3
Slowness on vidoes (Zou et al., 2012)	74.6
Multipath HMP (Bo et al., 2013)	82.5
Ex-CNN (Small/Large) (Dosovitskiy et al., 2016)	79.8 / 87.1
Pre-trained (ResNet-50)	84.3 / 85.6
Self-tuning (ResNet-50)	87.1 / 88.8
Pre-trained (ResNet-101)	85.2 / 85.9
Self-tuning (ResNet-101)	87.6 / **89.3**

4.6. Performance comparison with unsupervised feature learning methods

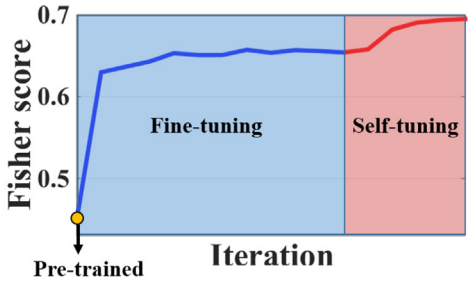
Since the proposed self-tuning is the first attempt to transfer the neural networks in a fully unsupervised manner, fair comparison with existing similar algorithms is not a simple task. In this section, we compare the proposed algorithm with the previous unsupervised feature learning methods such as pyramidal max pooling (Boureau et al., 2011), temporal slowness constraint (Zou et al., 2012), hierarchical matching pursuit (HMP) (Bo et al., 2013) and Exemplar CNN (Dosovitskiy et al., 2016). They use deep architectures to learn feature representation for visual classification tasks. Hence, we compare our KNN(K = 1 and 5) results of ResNet on the Caltech101 dataset to the best reported results of the previous works. Table 6 shows the self-tuning performs well compared to previous unsupervised feature learning methods. This result also supports the unsupervised feature learning ability of the self-tuning method.

5. Discussion

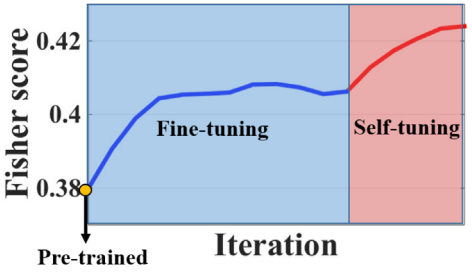
We propose a 'fully-unsupervised' network learning method. Therefore, to emphasize the complete unsupervised-ness, we summarize the differences between ours and the previous method with regard to the use of labels as shown in Table 7. Note that the proposed self-tuning algorithm does not use any labels in a training stage. In contrast, existing approaches, such as few-shot learning and domain adaptation, needs some supervision in learning process. Few-shot learning requires a small number of class labels in the target domain for training. To apply meta-learning to the few-shot approach, additional class labels are necessary. Domain adaptation also needs class labels of the training data in the source domain, although those of the target domain is not required. Therefore, we expect that the proposed approach can be utilized for many learning algorithms especially with large-scale data. Labeling large data is obviously very difficult, so the proposed approach will help train deep neural networks while reducing human labor. In addition, the proposed approach will be effective even if there are noisy labels or partially annotated labels in training data.

Table 7
Differences of four approaches in terms of label requirement. The self-tuning do not use any class labels in training.

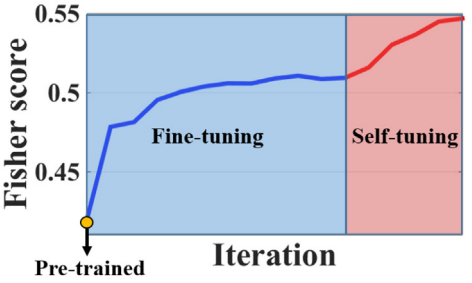
	Few-shot	Domain adaptation	Self-tuning
Labels of training data	Required only a few	**Required for source** not required for target	**Not required**
Category	Semi-supervised	**Unsupervised in target domain**, but supervised in source domain	**Fully unsupervised**



(a) Caltech 101



(b) MIT Indoor



(c) Flower 102

Fig. 5. Improvement of Fisher score by the tuning methods of Caltech 101 with ResNet-101. The yellow circle denotes the Fisher score value of the pre-trained model. Blue and red regions indicate the fine-tuning on the pre-trained model and self-tuning on fine-tuned one, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Conclusion

In this paper, we propose a fully-unsupervised self-tuning algorithm that can transfer a pre-trained neural network to a new target domain only using unlabeled training data. The self-tuning is conducted by minimizing the triplet loss function, and the triplets are automatically sampled based on the ranking violations of the relevance scores and Euclidean feature distances. The relevance of unlabeled data points is estimated by bagged clustering with feature bootstrapping. Extensive experiments are carried out to validate the ability of the proposed self-tuning method. Classification, feature representation, and clustering metrics on five domains with five neural networks show significant improvement over the pre-trained networks. Furthermore, we demonstrate that the self-tuning method improves the performance of the fine-tuned neural networks. These results are significant as the proposed self-tuning methods can also facilitate achieving state-of-the-art performance in various domains.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP), Korea grant funded by the Korea government(MSIT) (No. 2020-0-01373, Artificial Intelligence Graduate School Program(Hanyang University)), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Korea (NRF-2017R1A6A3A11031193), and the NSF CAREER, United States of America Grant #1149783.

References

Antoniou, A., Storkey, A., & Edwards, H. (2017). Data augmentation generative adversarial networks. arXiv preprint arXiv:1711.04340.

Balaji, Y., Sankaranarayanan, S., & Chellappa, R. (2018). Metareg: Towards domain generalization using meta-regularization. In *Neural information processing systems* (pp. 998–1008).

Bhardwaj, S., Srinivasan, M., & Khapra, M. M. (2019). Efficient video classification using fewer frames. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 354–363).

Bo, L., Ren, X., & Fox, D. (2013). Multipath sparse coding using hierarchical matching pursuit. In *IEEE conference on computer vision and pattern recognition* (pp. 660–667).

Boureau, Y.-L., Le Roux, N., Bach, F., Ponce, J., & LeCun, Y. (2011). Ask the locals: multi-way local pooling for image recognition. In *IEEE international conference on computer vision* (pp. 2651–2658).

Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531.

Chechik, G., Sharma, V., Shalit, U., & Bengio, S. (2010). Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, *11*(Mar), 1109–1135.

Dasgupta, S., & Freund, Y. (2008). Random projection trees and low dimensional manifolds. In *ACM symposium on theory of computing* (pp. 537–546).

Dosovitskiy, A., Fischer, P., Springenberg, J. T., Riedmiller, M., & Brox, T. (2016). Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(9), 1734–1747.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification*. John Wiley & Sons.

Fei-Fei, L., Fergus, R., & Perona, P. (2007). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, *106*(1), 59–70.

Ganin, Y., & Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *International conference on machine learning* (pp. 1180–1189).

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*(Mar), 1157–1182.

Hariharan, B., & Girshick, R. (2017). Low-shot visual recognition by shrinking and hallucinating features. In *IEEE international conference on computer vision* (pp. 3018–3027).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Neural information processing systems* (pp. 1097–1105).

Liang, J., He, R., Sun, Z., & Tan, T. (2018). Aggregating randomized clustering-promoting invariant projections for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(5), 1027–1042.

Liang, J., He, R., Sun, Z., & Tan, T. (2019). Exploring uncertainty in pseudo-label guided unsupervised domain adaptation. *Pattern Recognition*, *96*, Article 106996.

Liu, H., Long, M., Wang, J., & Jordan, M. (2019). Transferable adversarial training: A general approach to adapting deep classifiers. In *International conference on machine learning* (pp. 4013–4022).

Long, M., Zhu, H., Wang, J., & Jordan, M. I. (2016). Unsupervised domain adaptation with residual transfer networks. In *Neural information processing systems* (pp. 136–144).

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(Nov), 2579–2605.

Nilsback, M.-E., & Zisserman, A. (2008). Automated flower classification over a large number of classes. In *Indian conference on computer vision, graphics and image processing* (pp. 722–729).

Quattoni, A., & Torralba, A. (2009). Recognizing indoor scenes. In *IEEE conference on computer vision and pattern recognition* (pp. 413–420).

Ryu, J., Bae, J., & Lim, J. (2020). Collaborative training of balanced random forests for open set domain adaptation. arXiv preprint arXiv:2002.03642.

Ryu, J., Kwon, G., Yang, M.-H., & Lim, J. (2020). Generalized convolutional forest networks for domain generalization and visual recognition. In *International conference on learning representations*. URL: https://openreview.net/forum?id=H1lxVyStPH.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *IEEE conference on computer vision and pattern recognition* (pp. 815–823).

Sharan, L., Rosenholtz, R., & Adelson, E. (2009). Material perception: What can you see in a brief glance? *Journal of Vision*, *9*(8), 784.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Tao, J., Hu, W., & Wen, S. (2016). Multi-source adaptation joint kernel sparse representation for visual classification. *Neural Networks*, *76*, 135–151.

Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., & Wu, Y. (2014a). Learning fine-grained image similarity with deep ranking. In *IEEE conference on computer vision and pattern recognition*.

Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., & Wu, Y. (2014b). Learning fine-grained image similarity with deep ranking. In *IEEE conference on computer vision and pattern recognition* (pp. 1386–1393).

Yang, J., Cao, J., Wang, T., Xue, A., & Chen, B. (2020). Regularized correntropy criterion based semi-supervised ELM. *Neural Networks*, *122*, 117–129.

Yu, R., Li, A., Morariu, V. I., & Davis, L. S. (2017). Visual relationship detection with internal and external linguistic knowledge distillation. In *Proceedings of the IEEE international conference on computer vision* (pp. 1974–1982).

Zhou, T., Brown, M., Snavely, N., & Lowe, D. G. (2017). Unsupervised learning of depth and ego-motion from video. In *IEEE conference on computer vision and pattern recognition* (vol. 2) (no. 6) (p. 7).

Zou, W., Zhu, S., Yu, K., & Ng, A. Y. (2012). Deep learning of invariant features via simulated fixations in video. In *Neural information processing systems* (pp. 3203–3211).