Unsupervised Deep Learning via Affinity Diffusion

Jiabo Huang[1], **Qi Dong**[1], **Shaogang Gong**[1], **Xiatian Zhu**[2]
[1] Queen Mary University of London, [2] Vision Semantics Limited
{jiabo.huang, q.dong, s.gong}@qmul.ac.uk, eddy.zhuxt@gmail.com

Abstract

Convolutional neural networks (CNNs) have achieved unprecedented success in a variety of computer vision tasks. However, they usually rely on supervised model learning with the need for massive labelled training data, limiting dramatically their usability and deployability in real-world scenarios without any labelling budget. In this work, we introduce a general-purpose unsupervised deep learning approach to deriving discriminative feature representations. It is based on self-discovering semantically consistent groups of unlabelled training samples with the same class concepts through a progressive affinity diffusion process. Extensive experiments on object image classification and clustering show the performance superiority of the proposed method over the state-of-the-art unsupervised learning models using six common image recognition benchmarks including MNIST, SVHN, STL10, CIFAR10, CIFAR100 and ImageNet.

Introduction

Convolutional neural networks (CNNs) trained in a *supervised* fashion have significantly boosted the state-of-the-art performance in computer vision (Krizhevsky, Sutskever, and Hinton 2012; Girshick 2015; Long, Shelhamer, and Darrell 2015). Moreover, the feature representations of a supervised CNN (e.g. trained for classification on ImageNet (Russakovsky et al. 2015)) generalise to new tasks. Despite such remarkable success, this approach is limited due to a number of stringent assumptions. First, supervised model learning requires enormous labelled datasets to be collected manually and exhaustively (Dong, Gong, and Zhu 2018; Dong, Zhu, and Gong 2019). This does not always hold valid due to high annotation costs. Besides, the benefits of enlarging labelled datasets may have diminishing returns (Sun et al. 2017). Second, transfer learning becomes less effective when the target tasks significantly differ from the source task (Goodfellow, Bengio, and Courville 2016; Dong, Gong, and Zhu 2017). Due to the only need for access of unlabelled data typically available at scale, unsupervised deep learning provides a conceptually generic and scalable solution to these limitations (Caron et al. 2018).

There are an increasing number of recent attempts on unsupervised deep learning (Wu et al. 2018; Huang et al. 2019; Xie, Girshick, and Farhadi 2016; Caron et al. 2018; Ji, Henriques, and Vedaldi 2019; Haeusser et al. 2018; Yang, Parikh, and Batra 2016; Yang et al. 2017; Gidaris, Singh, and Komodakis 2018; Zhang, Isola, and Efros 2017; Noroozi and Favaro 2016; Doersch, Gupta, and Efros 2015). One intuitive strategy is joint learning of feature representations and data clustering (Xie, Girshick, and Farhadi 2016; Ji, Henriques, and Vedaldi 2019; Caron et al. 2018; Yang et al. 2017). It aims to automatically discover clustering solutions during training, with each cluster hopefully capturing a specific class concept. This objective is extremely hard due to the numerous combinatorial configurations of unlabelled data alongside highly complex inter-class decision boundaries. To avoid clustering errors as well as the following propagation, instance learning is proposed (Wu et al. 2018; Bojanowski and Joulin 2017) whereby every single sample is treated as an independent class. However, this simplified supervision is often rather ambiguous particularly around class centres, therefore, leading to weak class discrimination. As an intermediate representation, tiny neighbourhoods are leveraged for preserving the advantages of both data clustering and instance learning (Huang et al. 2019). But this method is restricted by the small size of local neighbourhoods. Another representative approach is designing effective pretext tasks with supervision labels produced automatically from unlabelled data (Gidaris, Singh, and Komodakis 2018; Zhang, Isola, and Efros 2017; Noroozi and Favaro 2016; Doersch, Gupta, and Efros 2015). Due to insufficient correlation with the target class supervision, these methods often result in less competitive models.

In this work, we aim to solve the algorithmic limitations of existing unsupervised deep learning methods. To that end, we propose a general-purpose *Progressive Affinity Diffusion* (PAD) method for training unsupervised models. Requiring no prior knowledge of class number, PAD performs model-matureness-adaptive data group inference in training for more reliably revealing the underlying sample-to-class memberships. This is achieved by progressively self-discovering strongly connected subgraphs on a neighbourhood affinity graph via faithful affinity diffusion and formu-

lating the group structure aware objective loss function.

The **contributions** of this work are summarised as the followings. **(1)** We propose a novel idea of leveraging strongly connected subgraphs as a self-supervision structure for more reliable unsupervised deep learning. **(2)** We formulate a *Progressive Affinity Diffusion* (PAD) method for model-matureness-adaptive discovery of strongly connected subgraphs during training through affinity diffusion across adjacent neighbourhoods. This strategy maximises the class consistency of self-discovered subgraphs therefore enhancing unsupervised model learning capability. PAD is end-to-end trainable. **(3)** We design a group structure aware objective loss formulation for more discriminative capitalising of strongly connected subgraphs in model representation learning. Comparative experiments have been extensively conducted on both image classification and image clustering tasks using popular benchmarks: MNIST (LeCun et al. 1998), SVHN (Netzer et al. 2011), STL10 (Coates, Ng, and Lee 2011), CIFAR10 and CIFAR100 (Krizhevsky and Hinton 2009) as well as ImageNet (Russakovsky et al. 2015). The results show that our method outperforms a wide variety of existing state-of-the-art unsupervised learning models, often by large margins.

Related Work

In the literature, most existing unsupervised deep learning methods fall generally into four groups: (i) joint clustering (Caron et al. 2018; Xie, Girshick, and Farhadi 2016; Yang et al. 2017; Haeusser et al. 2018; Ji, Henriques, and Vedaldi 2019; Huang et al. 2019), (ii) instance learning (Wu et al. 2018; Bojanowski and Joulin 2017), (iii) pretext task designing (Gidaris, Singh, and Komodakis 2018; Zhang, Isola, and Efros 2017; Noroozi and Favaro 2016; Wang, He, and Gupta 2017; Dahun Kim 2018; Doersch, Gupta, and Efros 2015; Zhang, Isola, and Efros 2016), and (iv) generative model formulation (Radford, Metz, and Chintala 2016; Donahue, Krähenbühl, and Darrell 2016).

Joint clustering methods aim to integrate the classical idea of data grouping (Aggarwal and Reddy 2013) into the end-to-end optimisation of unsupervised learning models. Often, the cluster labels self-formed are treated as concept annotations and supervised learning techniques such as softmax cross-entropy criterion are then adopted for model optimisation. Typical methods require the access of ground-truth class number which is usually unavailable in realistic scenarios. The training process may involve multiple stages continuously (Haeusser et al. 2018) or iteratively (Caron et al. 2018; Xie, Girshick, and Farhadi 2016; 2016). This approach is rather challenging due to the error-prone division of samples, especially when dealing with complex structures and distributions as typically encountered in images and videos. Our PAD method also aims to discover label consistent groups. However, unlike the clustering operation that enforces each and every sample to be clustered with a specific cluster, we take a more robust grouping scheme which associates strongly similar samples alone so that erroneous memberships can be minimised.

Instance learning strategy leaps to the other extreme by regarding every single sample as a unique class (Wu et al. 2018; Bojanowski and Joulin 2017). It is motivated by the observation that supervised learning can naturally encode global manifold information over different classes from end-to-end optimisation. However, by assigning samples from the same classes with different labels, this approach will push their features away, therefore, tends to dilate per-class regions and achieve less discriminative class boundaries. Evidently, this problem is clearly solved in our method due to the ability of searching label consistent groups of samples with the same class concept. Besides, instance learning can be considered as a special case of our method with the subgraph size as one.

Pretext task designing has been shown as strong alternative methods for unsupervised deep learning. In general, existing methods mainly differ in the auxiliary supervision formulation, typically hand-crafted, in order to exploit information intrinsic to training data. Representative examples include spatial context (Gidaris, Singh, and Komodakis 2018; Doersch, Gupta, and Efros 2015; Noroozi and Favaro 2016), spatio-temporal continuity (Wang, He, and Gupta 2017; Dahun Kim 2018) and colour distributions (Zhang, Isola, and Efros 2016; 2017). Conceptually, existing methods present only weak linkages between auxiliary supervision and class concepts, yielding inferior models. Whilst the potential of this approach is great, how to design more discriminating pretext task remains unsolved and lacks sufficient guidance. In design, our method is highly complementary as no pretext task is involved.

Generative model formulation is another important approach in unsupervised learning. Fundamental methods include Restricted Boltzmann Machines (RBMs) (Tang, Salakhutdinov, and Hinton 2012; Lee et al. 2009), Autoencoder (Vincent et al. 2010; Ng 2011) and Generative Adversarial Networks (GANs) (Radford, Metz, and Chintala 2016; Donahue, Krähenbühl, and Darrell 2016). As compared to discriminative models like ours, generative methods generally present weaker discrimination power due to lack of class discriminative learning. However, it is also this difference that may make their representations potentially highly complementary.

Progressive Affinity Diffusion

In unsupervised learning, we have access to a set of N *unlabelled* training image samples $\mathcal{I} = \{\boldsymbol{I}_1, \boldsymbol{I}_2, ..., \boldsymbol{I}_N\}$. We aim to train a CNN model f_θ for extracting class discriminative features $\boldsymbol{x}$, i.e. $f_\theta : \boldsymbol{I} \rightarrow \boldsymbol{x}$, where θ denotes the model parameters. The *key* in unsupervised learning is to bridge the semantic gap between low-level imagery appearance (e.g. pixel intensity) and high-level semantic concepts (e.g. object class). It is intrinsically challenging due to large intra-class variety and inter-class similarity in addition to unknown variations (e.g. background, pose, illumination) and high-dimensional data representation. Besides, variation factors are often confounding arbitrarily (Fig 2(b)), further increasing the modelling difficulty drastically. This complexity exhibited in typical image collections implies the necessity to infer highly non-linear class decision boundaries in unsupervised learning.

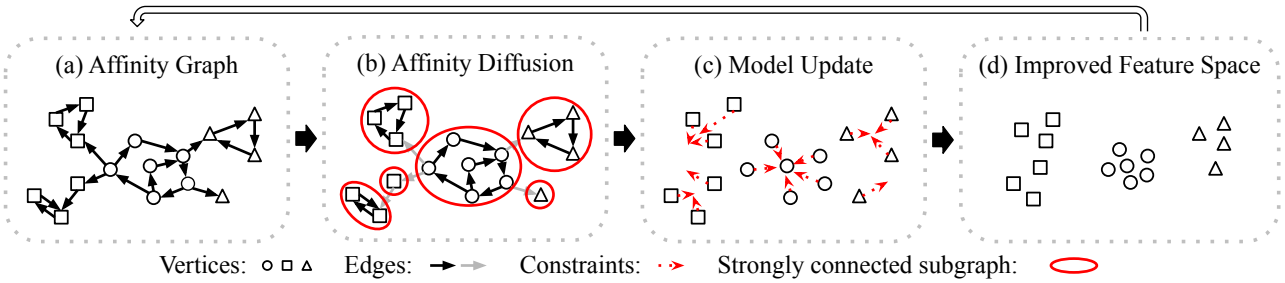

Figure 1: Overview of our *Progressive Affinity Diffusion* (PAD) method. **(a)** Affinity graph construction by k nearest neighbourhoods. **(b)** Affinity diffusion across neighbourhoods to discover label consistent *strongly connected subgraphs*. **(c)** Progressive model update with self-discovered subgraphs, leading to **(d)** improved feature representations. The model is trained *iteratively*.

To overcome the challenges, we formulate ***Progressive Affinity Diffusion*** (**PAD**). PAD aims to discover sample groups each representing one of the underlying semantic concepts as self-supervision, hence facilitating class boundary inference. For avoiding clustering errors, it is particularly designed to exploit individual pairwise relationships with capricious variations in a purely data-driven manner.

Approach overview. An overview of our PAD method is depicted in Fig 1. Specifically, PAD is an iterative unsupervised model learning process including three components: **(1)** Affinity graph construction for representing the global structure of training data, **(2)** Affinity diffusion across neighbourhoods for self-discovering groups of samples with the same semantics, **(3)** Progressive model update by formulating group structure aware objective loss function. They are integrated into a *multi-stage* procedure. In each stage the model mines only the reliable data groups that have emerged thus far in the affinity graph (i.e. *model-matureness-adaptive*) other than clustering all the samples, which then feed into the subsequent model training stages sequentially. We describe the model training details at the end of this section.

Affinity Graph Construction

Progressive affinity diffusion is carried out through graphs $\mathcal{G}$ of unlabelled training samples. To construct the graph, we leverage per-sample k-nearest neighbourhoods $\mathcal{N}_k$ defined as:

$$\mathcal{N}_k(\boldsymbol{x}) = \{\boldsymbol{x}_i \mid \mathcal{S}(\boldsymbol{x}_i, \boldsymbol{x}) \text{ is ranked top-}k \text{ among } X\} \quad (1)$$

where X denotes the feature set of all training samples extracted by the up-to-date model and $\mathcal{S}$ represents the cosine similarity function. The neighbourhoods represent a kind of data grouping (Huang et al. 2019). However, without class label supervision, such structures convey either limited (small k) or noisy (large k) affinity information between training samples/vertices (Fig 2(a)). Using directly $\mathcal{N}_k$ for self-supervision is therefore restricted.

Affinity Diffusion across Neighbourhoods

To address the aforementioned problem, we propose affinity diffusion across neighbourhoods. This aims to model the correlation of different $\mathcal{N}_k$ in order to break through their

(a) k-nearest neighbours (b) strongly connected subgraph

Figure 2: **(a)** k-nearest neighbours vs. **(b)** strongly connected subgraph. *Blue box*: the anchor. *Green box*: with the same class as the anchor. *Red box*: with a different class against the anchor.

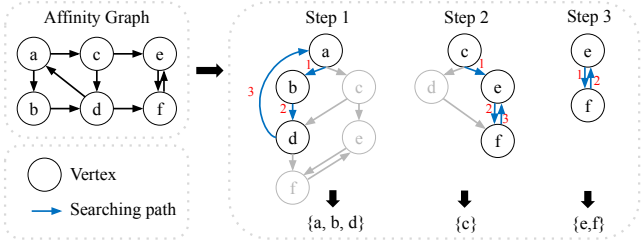

Figure 3: Illustration of searching *strongly connected subgraphs* in an neighbourhood affinity graph. In this example, the scale threshold s is set to 3.

barriers and spread the class identity of one sample through. Conceptually, this leverages global data manifold (Belkin, Niyogi, and Sindhwani 2006) formed collectively by all neighbourhoods. As each neighbourhood may encode visual affinity information in a distinct perspective, linking them is equivalent to joining different types of variations of a concept, therefore enabling model learning to capture more comprehensive concept boundaries.

For discriminative diffusion, it is critical that error introduction and propagation are minimised. To this end, motivated by graph theory (Duda, Hart, and Stork 2012; Raghavan and Yu 1981) we propose to search **strongly connected subgraphs** (SCS) of the affinity graph $\mathcal{G}$ for revealing underlying semantic concepts. A SCS structure is defined as a set of vertices (images) where each vertex can be reached from any others through neighbouring edges. This

means that all vertices of a SCS are highly similar w.r.t. some variation criteria.

SCS structures are formed as shown in Fig 3. Starting with a random vertex $\boldsymbol{x}_s$, we construct a tree-like structure based on the edges. Then, we traverse the vertices along edges in a depth-first search strategy. We assign nearest neighbours with the highest priority for maximising the average pairwise affinity of a SCS. Depth-first search can easily achieve this condition. It is constrained that each vertex can be visited only one time to avoid repeated traversing. To ensure reachability of any two vertices, we enforce a **cyclic constraint** (Fig 2(b)). In the tree-like structure, one cycle that loops back to the start vertex $\boldsymbol{x}_s$ is necessary, whilst any other cycles allow to exclude $\boldsymbol{x}_s$ as long as partly overlapped with the former. In the search process, we aim to find disjoint SCS structures. Hence, we remove all the vertices of a SCS once found from $\mathcal{G}$ to simplify and accelerate the subsequent searching. This process repeats until no SCS exists. Note that at the end, a number of isolated vertices are likely left (outside any SCS).

With the above search method, however, we find that the resulting SCS tends to be over large even when the graph is not dense. This ends up with mixed samples of different class concepts. To address this issue, we further impose an **operational size (scale) threshold** s on SCS. This simple constraint works in our case considering that we are *not* seeking a *complete* group of samples per class which can be extremely challenging and risky as in clustering methods, due to complex observation variations. Even splitting a single class into multiple SCS structures, we are still able to obtain a large amount of intra-class variation information, whereas the risk of class mixture is significantly reduced.

Progressive Model Updating

Next, we describe how the self-discovered SCS structures can be used for progressive model update. For formulation ease, we treat individual samples as special SCS structures each with one sample. We propose to further convert affinity measurements into probability distributions, so that maximum likelihood-based learning objective functions (Goodfellow, Bengio, and Courville 2016) can be adopted. Inspired by (Huang et al. 2019; Wu et al. 2018), we define the probability that any two samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are drawn from the same class as:

$$p_{i,j} = \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{x}_j/\tau)}{\sum_{k=1}^{N}\exp(\boldsymbol{x}_i^\top \boldsymbol{x}_k/\tau)} \tag{2}$$

where τ is a temperature parameter controlling the distribution concentration degree (Hinton, Vinyals, and Dean 2014). This quantity is naturally compatible with SCS formation, both relying on pairwise affinity. To reinforce the SCS structural information into model learning, we maximise the same-class possibility of samples per SCS. We therefore formulate a group (subgraph) structure aware objective function as:

$$\mathcal{L}_{\text{scs}} = -\frac{1}{n_{bs}}\sum_{i=1}^{n_{bs}}\log\Big(\sum_{j\in\mathcal{C}(\boldsymbol{x}_i)} p_{i,j}\Big) \tag{3}$$

where n_{bs} specifies the mini-batch size and $\mathcal{C}(\boldsymbol{x}_i)$ the samples of the SCS structure including $\boldsymbol{x}_i$. This encourages affinity maximisation of samples within SCS. For isolated samples, $\mathcal{C}(\boldsymbol{x}_i) = \{\boldsymbol{x}_i\}$, $\mathcal{L}_{\text{scs}}$ turns to the instance loss function (Wu et al. 2018). Eq (3) is also analogous to Neighbourhood Component Analysis (NCA) (Goldberger et al. 2005) when $\mathcal{C}(\boldsymbol{x}_i)$ is replaced with labelled sets.

With summation $\sum_{j\in\mathcal{C}(\boldsymbol{x}_i)} p_{i,j}$, one possible weakness of Eq (3) is that less similar neighbours can be overwhelmed due to over small quantity. To calibrate their importance, we introduce a **hard positive enhancement** strategy. Given a SCS, for a sample $\boldsymbol{x_i}$ we define its hard positive sample as the one $\boldsymbol{x}_{hp}$ with the smallest affinity. In the case of isolated sample, we use a randomly transformed variant as its hard positive. For calibration, we minimise the Kullback-Leibler (KL) divergence of model predictions of $\boldsymbol{x}_i$ and $\boldsymbol{x}_{hp}$ as:

$$\mathcal{L}_{\text{hpe}} = \frac{1}{n_{bs}}\sum_{i=1}^{n_{bs}}\sum_{j=1}^{N} p_{i,j}\log\frac{p_{i,j}}{p_{hp,j}} \tag{4}$$

We obtain the final loss function of our model by weighted summation as:

$$\mathcal{L}_{\text{pad}} = \mathcal{L}_{\text{scs}} + \lambda\mathcal{L}_{\text{hpe}} \tag{5}$$

where the weight λ modulates their importance balance.

Model Training

PAD starts with a randomly initialised CNN. For efficiency, we update the affinity graph and SCS per epoch, and use a memory to keep track of per-sample representations required by loss computation (Eq (2)). The memory is updated for mini-batch samples by exponential moving average (Lucas and Saccucci 1990):

$$\tilde{\boldsymbol{x}}_i = (1-\eta)\cdot\tilde{\boldsymbol{x}}_i + \eta\cdot\boldsymbol{x}_i \tag{6}$$

where η denotes the update momentum, $\boldsymbol{x}_i$ and $\tilde{\boldsymbol{x}}_i$ the up-to-date and memory feature vectors respectively. This can summarise the historical knowledge, learned in the past training iterations, to solve the incomplete memory limitation of neural networks (Weston, Chopra, and Bordes 2014). The whole model training procedure is summarised in Algorithm 1.

Algorithm 1 Unsupervised deep learning via progressive affinity diffusion.

Input: Training data $\mathcal{I}$, training epochs N_{ep}, iterations per epoch N_{it};
Output: A class discriminative CNN feature representation model;
for epoch $= 1$ **to** N_{ep} **do**
  Construct the kNN based affinity graph (Eq (1));
  Search strongly connected subgraphs (Fig 3);
  for iter $= 1$ **to** N_{it} **do**
    Mini-batch feed-forward through the network;
    Objective loss computation (Eq (5));
    Network back-propagation and model weights update;
    Memory feature refreshing (Eq (6)).
  end for
end for

Experiments

Datasets. ***CIFAR10(/100)***: A natural image dataset containing 50,000/10,000 train/test images from 10 (/100) object classes. ***ImageNet***: A large scale 1,000 classes object dataset with 1.2 million images for training and 50,000 for test. ***SVHN***: A Street View House Numbers dataset including 10 classes of digit images. ***STL10***: An ImageNet adapted dataset containing 500/800 train/test samples from 10 classes as well as 100,000 unlabelled images from auxiliary unknown classes. ***MNIST***: A hand-written digits dataset with 60,000/10,000 train/test images from 10 digit classes.

Evaluation protocol. We considered two test protocols for unsupervised learning of discriminative representations.

(1) *Image classification* (Caron et al. 2018; Wu et al. 2018), where the ground-truth class labels of the training images are used *after* unsupervised model learning for enabling image categorisation. We tested two classification models, Linear Classifier (LC) with conv5 features and Weighted kNN with FC features. LC was realised by a fully connected layer. The kNN classifier is based on weighted voting of top-k neighbours $\mathcal{N}_k$ as $s_c = \sum_{i \in \mathcal{N}_k} \delta(c, c_i) \cdot w_i$ where $\delta(c, c_i)$ is a Dirac function returning 1 if label $c = c_i$, and 0 otherwise. We computed the weight w_i as $w_i = \exp(s_i/\tau)$ with $\tau = 0.1$ the temperature parameter and s_i the cosine similarity. Without an extra classifier learning post-process involved, kNN reflects *directly* the discriminative capability of the learned feature representations.

(2) *Image clustering* (Ji, Henriques, and Vedaldi 2019; Xie, Girshick, and Farhadi 2016) where k-means is applied if needed for clustering the samples represented by any unsupervised model into the ground-truth number of clusters. To measure the accuracy, we adopted the clustering accuracy (Xie, Girshick, and Farhadi 2016) which measures the proportion of samples correctly grouped. Note that both the training and test datasets are utilised for model learning in the standard clustering setting, unlike the standard classification setting where only the training dataset is used.

Implementation details. We used AlexNet as the backbone (Krizhevsky, Sutskever, and Hinton 2012). We adopted SGD with Nesterov momentum at 0.9, the epoch number at 200, the initial learning rate at 0.03 with a decay of 0.1 every 40 epochs after the first 80 ones. We set $k = 5$ (Eq (1)) for graph construction. The maximum size s of SCS is set to 10. We set the weight $\lambda = 0.8$ (Eq (5)) and the memory update rate $\eta = 0.5$ (Eq (6)). Data augmentation includes horizontal flipping, cropping and colour jittering. In practice, the SCS searching is conducted on memory features for efficiency concern and the memory bank takes around 600MB for ImageNet. Implemented in Tarjan framework (Tarjan 1972), the worst-case time complexity of our SCS searching is $\mathcal{O}(N^2)$. All the experiments run on Tesla P100 GPU.

Hyper-parameters selection. All our parameters are tuned on CIFAR10 and applied to all the other datasets. There is no per-dataset tuning. In particular, due to no validation set for cross-validation, we used common parameter settings: (1) we set k (Eq (1)) and s (maximum SCS size) so that there are neither too many isolated samples nor too large SCS units for error control; set λ (Weight of hard positive enhancement) so that no loss dominates; and set η (memory update rate) following (Wu et al. 2018; Huang et al. 2019). (2) For the SGD optimiser, we set the parameters so that the training loss can drop properly. This tuning process is easy to conduct in practice. Instead of exhaustively tuning the parameters, we used a *single* setting for all the experiments to test its scalability and generalisation.

Image Classification

In Table 1, we compared our PAD method with the representative works of *clustering analysis* (DeepCluster (Caron et al. 2018), AND (Huang et al. 2019)), *self-supervised learning* (RotNet (Gidaris, Singh, and Komodakis 2018)), and *sample specificity learning* (Instance (Wu et al. 2018)) on four benchmarks. We make three observations:

(1) PAD surpasses all competitors under either classification model, often by a large margin. This suggests the performance superiority of our method thanks to its strong capability of discovering underlying semantic boundaries.

(2) When compared with kNN classifier, linear classifier tends to yield better results due to using extra parameters. This is particularly so for the pretext task based model RotNet. This is because the pretext task is less relevant to classification, leading to weaker representation as compared to grouping based methods like AND and PAD.

(3) As an intermediate representation between clusters and instances, tiny neighbourhoods are exploited in AND for revealing class boundaries and improves the results in most cases. However, this method is restricted by the small size of neighbourhoods. PAD addresses this limitation by affinity diffusion across adjacent neighbourhoods.

Dataset	CIFAR10	CIFAR100	SVHN	ImageNet
Classifier/Feature	Weighted kNN / FC			
Random	34.5	12.1	56.8	3.5
DeepCluster	62.3	22.7	84.9	26.8
Instance	60.3	32.7	79.8	<u>31.3</u>
RotNet	72.5	32.1	77.5	9.2
AND	<u>74.8</u>	<u>41.5</u>	<u>90.9</u>	<u>31.3</u>
PAD	**81.5**	**48.7**	**91.2**	**35.1**
Classifier/Feature	Linear Classifier / conv5			
Random	67.3	32.7	79.2	14.1
DeepCluster	77.9	41.9	92.0	<u>38.2</u>
Instance	70.1	39.4	89.3	35.6
RotNet	<u>84.1</u>	<u>57.4</u>	92.3	36.5
AND	77.6	47.9	**93.7**	37.8
PAD	**84.7**	**58.6**	<u>93.2</u>	**38.6**

Table 1: Image classification results of unsupervised learning models. The 1st/2nd best results are indicated in **bold**/<u>underline</u>.

We examined the training dynamics of SCS size and precision. Fig 4 shows that PAD starts with finding small SCS structures due to weak representation power, then explores larger ones at decreasing precision, and finally converges the size whilst increases the precision before both levels off. High precision of SCS is a key for enabling more discriminative unsupervised learning by PAD.

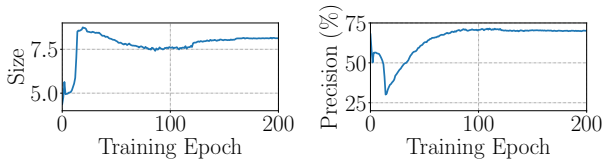

Figure 4: Statistics dynamics of SCS during training on CIFAR10: average size and precision.

Image Clustering

Apart from sample-wise image classification, we further tested the performance of our model on image clustering which reflects the representation quality in describing global data structures. We compared PAD with two groups of alternative methods, (1) *Clustering methods*: JULE (Yang, Parikh, and Batra 2016), DEC (Xie, Girshick, and Farhadi 2016), DAC (Chang et al. 2017), ADC (Haeusser et al. 2018) and IIC (Ji, Henriques, and Vedaldi 2019); (2) *Generic representation learning methods*: Triplets (Schultz and Joachims 2004), AE (Bengio et al. 2007), Sparse AE (Ng 2011), Denoising AE (Vincent et al. 2010), Variational Bayes AE (Kingma and Welling 2013), SWWAE (Zhao et al. 2015), DCGAN (Radford, Metz, and Chintala 2016) and DeepCluster (Caron et al. 2018). For the latter group including PAD, we further applied k-means to generate their clustering solutions. For PAD, we reported the average result over 10 runs. For competitors, we used the results from (Ji, Henriques, and Vedaldi 2019). Despite different modelling purposes, we performed both within and cross group comparisons. We made a couple of observations from Table 2:

(1) The first group of methods tends to produce higher clustering results, thanks to their joint learning of feature representations and clustering by using the ground-truth class number prior in end-to-end model training, i.e. consistent between training and test objectives. Among them, IIC achieves the best results.

(2) Without taking clustering as objective, the second group of methods is relatively inferior in modelling data group structures. However, PAD again reaches the best performance consistently in this group. Crucially, our model is on par with all the dedicated clustering methods and even surpasses them on CIFAR10/100 with significant margins, regardless of the disadvantage on STL10 which, we conjugate, is due to some distracting impact from auxiliary unknown categories. This indicates the efficacy of our unsupervised learning method in capturing the holistic data distribution. We attribute this advantage to the favourable ability of our method in seeking the latent class consistent groups with high variations of individual concepts.

Component Evaluations and Further Analysis

To provide insights into our model, we conducted a sequence of detailed component evaluations and performance analysis on the image classification task with kNN as classifier.

Effect of affinity diffusion. To investigate the effect of affinity diffusion, we tested the performance of models in which $\mathcal{C}(\boldsymbol{x})$ for each sample was replaced by its k-nearest neighbours $\mathcal{N}_k(\boldsymbol{x})$ as shown in Fig 2(a) and the size k was set to 10. According to Table 3, discovering consistent sample groups according to affinity diffusion across adjacent neighbourhoods with necessary constraints clearly benefits the discriminative learning of models in all cases.

Methods	MNIST	STL10	CIFAR10	CIFAR100
JULE	96.4	27.7	27.2	13.7
DEC	84.3	35.9	30.1	18.5
DAC	97.8	47.0	52.2	23.8
ADC	**99.2**	53.0	32.5	16.0
IIC	98.4	**59.8**	**57.6**	**25.5**
Random CNN †	48.1	20.1	18.6	10.3
Triplets †	52.5	24.4	20.5	9.9
AE †	81.2	30.3	31.4	16.5
Sparse AE †	82.7	32.0	29.7	15.7
Denoising AE †	83.2	30.2	29.7	15.1
Variational Bayes AE †	83.2	28.2	29.1	15.2
SWWAE †	82.5	27.0	28.4	14.7
DCGAN †	82.8	29.8	31.5	15.1
DeepCluster †	65.6	33.4	37.4	18.9
PAD (Ours) †	**98.2**	**46.5**	**62.6**	**28.8**

Table 2: Comparing image clustering results of unsupervised learning methods. †: Used k-means. The results of existing methods are adopted from (Ji, Henriques, and Vedaldi 2019).

Diffusion	CIFAR10	CIFAR100	SVHN
✗	77.5	34.5	89.5
✓	**81.5**	**48.7**	**91.2**

Table 3: Effect of affinity diffusion.

Cyclic and scale constraints. We examined the effect of *cyclic* and *scale* constraints and observed from Table 4 that: (1) Cyclic constraint brings consistently performance gain, particularly in the most challenging CIFAR100 test. This is because of the presence of subtle visual discrepancy between fine-grained classes, which leads to more wrong association in affinity diffusion. (2) Scale constraint is clearly necessary for ensuring the effectiveness of our model in all cases. Without it, different classes would be mixed up in diffusion due to complex visual patterns exhibited in images.

Cyclic	CIFAR10	CIFAR100	SVHN
✗	73.3	30.6	91.1
✓	**81.5**	**48.7**	**91.2**
Scale	**CIFAR10**	**CIFAR100**	**SVHN**
✗	20.2	1.8	20.3
✓	**81.5**	**48.7**	**91.2**

Table 4: Effect of (**Top**) cyclic and (**Bottom**) scale constraints in forming SCS structures.

To provide visual interpretation, we conducted a case study of affinity diffusion on STL10. Fig 5 shows that when the model is immature, wrong cross-class diffusion may happen frequently; the cyclic constraint can help detect this and early stop error accumulation. Besides, it is shown that

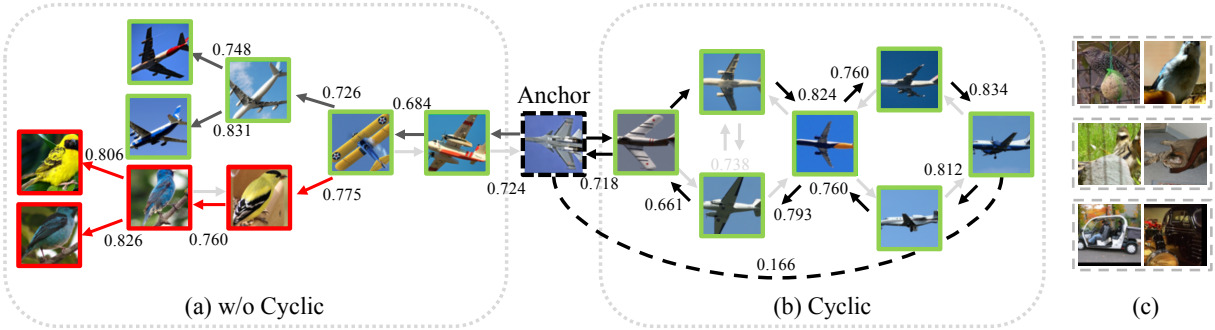

Figure 5: Case study on STL10. **(a)** *vs.* **(b)**: Cyclic constraint helps detect erroneous affinity diffusion. **(c)** Extremely hard positive pairs undiscovered by PAD. *Red solid box*: with a different class as the anchor; *Green solid box*: with the same class as the anchor. The numbers above arrows are the corresponding pairwise affinity scores.

SCS can better capture semantic similarity beyond pairwise affinity measurements (see dashed curve). However, as expected not all the hard positive pairs are discovered due to extreme viewing condition discrepancy as shown in Fig 5(c).

Hard positive enhancement. Table 5 (top) shows that constraining the prediction between hard positive pairs is clearly beneficial for model discriminative learning. This confirms the overwhelming effect among within-SCS samples when using normalised affinity measurements (Eq (2)) to quantify loss function (Eq (3)), and suggests the efficacy of our enhancement strategy. We also compared two loss designs: feature cosine similarity (FCS) *vs.* Kullback-Leibler (KL) divergence. Table 5 (bottom) suggests the superiority of KL over FCS. A plausible explanation is that KL can integrate with SCS loss $\mathcal{L}_{\text{scs}}$ (Eq (3)) in a more harmonious manner, as both are based on class posterior probability measurements.

HPE	CIFAR10	CIFAR100	SVHN
✗	69.8	30.9	80.9
✓	**81.5**	**48.7**	**91.2**
Design	CIFAR10	CIFAR100	SVHN
FCS	72.7	39.1	90.4
KL	**81.5**	**48.7**	**91.2**

Table 5: (**Top**) Effect of hard positive enhancement (HPE) and (**Bottom**) the HPE loss design comparison of feature cosine similarity (FCS) and Kullback-Leibler (KL) divergence.

Parameter analysis. We evaluated 3 parameters of PAD on CIFAR10: (1) Affinity graph density k, (2) SCS scale threshold s, and (3) weight λ of hard positive enhancement loss. Table 6 shows that the parameters are insensitive with a wide range of good values, indicating training robustness.

Computation cost of SCS searching. To validate the complexity of SCS searching, we tested the searching time on ImageNet: 4 minutes per epoch, i.e. 800 minutes among the overall 6 days training.

Conclusion

In this work, we presented a novel *Progressive Affinity Diffusion* (PAD) method for discriminative unsupervised deep learning. It is achieved by self-discovering class consistent strongly connected subgraphs in neighbourhood affinity graphs and formulating group structure aware objective loss function. This model can be trained end-to-end in a progressive multi-stage manner. Critically, PAD overcomes the notorious error propagation of clustering and the small locality limitation of neighbourhoods, whilst preserving and integrating their intrinsic strengths for more effective discriminative learning. Extensive experiments on image classification and image clustering tasks validate the superiority of PAD over a wide spectrum of state-of-the-art unsupervised deep learning methods.

k	1	3	5	10
Accuracy	78.3	79.3	**81.5**	79.7
s	5	10	50	100
Accuracy	80.2	**81.5**	80.7	73.5
λ	0.2	0.5	0.8	1.0
Accuracy	78.4	78.5	**81.5**	78.9

Table 6: Model parameter analysis on CIFAR10. **Top**: Affinity graph sparsity k; **Middle**: SCS scale threshold s; **Bottom**: Weight λ of hard positive enhancement (HPE) loss.

Acknowledgements

This work was supported by the China Scholarship Council, Vision Semantics Limited, the Alan Turing Institute Turing Fellowship, and the Innovate UK Industrial Challenge Project on Developing and Commercialising Intelligent Video Analytics Solutions for Public Safety (98111-571149)

References

Aggarwal, C. C., and Reddy, C. K. 2013. *Data clustering: algorithms and applications*. CRC press.

Belkin, M.; Niyogi, P.; and Sindhwani, V. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR* 7(Nov):2399–2434.

Bengio, Y.; Lamblin, P.; Popovici, D.; and Larochelle, H. 2007. Greedy layer-wise training of deep networks. In *Proc. NeurIPS*, 153–160.

Bojanowski, P., and Joulin, A. 2017. Unsupervised learning by predicting noise. In *Proc. ICML*, 1–10.

Caron, M.; Bojanowski, P.; Joulin, A.; and Douze, M. 2018. Deep clustering for unsupervised learning of visual features. In *Proc. ECCV*, 1–18.

Chang, J.; Wang, L.; Meng, G.; Xiang, S.; and Pan, C. 2017. Deep adaptive image clustering. In *Proc. ICCV*, 5879–5887.

Coates, A.; Ng, A.; and Lee, H. 2011. An analysis of single-layer networks in unsupervised feature learning. In *Proc. AIStats*, 215–223.

Dahun Kim, Donghyeon Cho, I. S. K. 2018. Self-supervised video representation learning with space-time cubic puzzles. In *Proc. AAAI*.

Doersch, C.; Gupta, A.; and Efros, A. A. 2015. Unsupervised visual representation learning by context prediction. In *Proc. ICCV*, 1422–1430.

Donahue, J.; Krähenbühl, P.; and Darrell, T. 2016. Adversarial feature learning. In *Proc. ICLR*, 1–18.

Dong, Q.; Gong, S.; and Zhu, X. 2017. Multi-task curriculum transfer deep learning of clothing attributes. In *WACV*.

Dong, Q.; Gong, S.; and Zhu, X. 2018. Imbalanced deep learning by minority class incremental rectification. *TPAMI*.

Dong, Q.; Zhu, X.; and Gong, S. 2019. Single-label multi-class image classification by deep logistic regression. *AAAI*.

Duda, R. O.; Hart, P. E.; and Stork, D. G. 2012. *Pattern classification*. John Wiley & Sons.

Gidaris, S.; Singh, P.; and Komodakis, N. 2018. Unsupervised representation learning by predicting image rotations. In *Proc. ICLR*, 1–16.

Girshick, R. 2015. Fast r-cnn. In *Proc. ICCV*, 1440–1448.

Goldberger, J.; Hinton, G. E.; Roweis, S. T.; and Salakhutdinov, R. R. 2005. Neighbourhood components analysis. In *Proc. NeurIPS*, 513–520.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep learning*. MIT press.

Haeusser, P.; Plapp, J.; Golkov, V.; Aljalbout, E.; and Cremers, D. 2018. Associative deep clustering: Training a classification network with no labels. In *Proc. GCPR*, 18–32. Springer.

Hinton, G.; Vinyals, O.; and Dean, J. 2014. Distilling the knowledge in a neural network. *Proc. NeurIPS*.

Huang, J.; Dong, Q.; Gong, S.; and Zhu, X. 2019. Unsupervised deep learning by neighbourhood discovery. In *Proc. ICML*.

Ji, X.; Henriques, J. F.; and Vedaldi, A. 2019. Invariant information distillation for unsupervised image segmentation and clustering. In *Proc. ICCV*, 1–10.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Proc. NeurIPS*, 1097–1105.

LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P.; et al. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Lee, H.; Grosse, R.; Ranganath, R.; and Ng, A. Y. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proc. ICML*, 609–616.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, 3431–3440.

Lucas, J. M., and Saccucci, M. S. 1990. Exponentially weighted moving average control schemes: properties and enhancements. *Technometrics* 32(1):1–12.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. In *NeurIPS workshop on deep learning and unsupervised feature learning*, number 2, 5.

Ng, A. 2011. Sparse autoencoder. 1–19.

Noroozi, M., and Favaro, P. 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proc. ECCV*, 69–84.

Radford, A.; Metz, L.; and Chintala, S. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proc. ICLR*.

Raghavan, V. V., and Yu, C. 1981. A comparison of the stability characteristics of some graph theoretic clustering methods. *TPAMI* (4):393–402.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *IJCV* 115(3):211–252.

Schultz, M., and Joachims, T. 2004. Learning a distance metric from relative comparisons. In *Proc. NeurIPS*, 41–48.

Sun, C.; Shrivastava, A.; Singh, S.; and Gupta, A. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *Proc. ICCV*, 843–852.

Tang, Y.; Salakhutdinov, R.; and Hinton, G. 2012. Robust boltzmann machines for recognition and denoising. In *Proc. CVPR*.

Tarjan, R. 1972. Depth-first search and linear graph algorithms. *SIAM journal on computing* 1(2):146–160.

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; and Manzagol, P.-A. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR* 11(Dec):3371–3408.

Wang, X.; He, K.; and Gupta, A. 2017. Transitive invariance for self-supervised visual representation learning. In *Proc. ICCV*, 1329–1338.

Weston, J.; Chopra, S.; and Bordes, A. 2014. Memory networks. In *Proc. ICLR*.

Wu, Z.; Xiong, Y.; Stella, X. Y.; and Lin, D. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proc. CVPR*.

Xie, J.; Girshick, R.; and Farhadi, A. 2016. Unsupervised deep embedding for clustering analysis. In *Proc. ICML*, 478–487.

Yang, B.; Fu, X.; Sidiropoulos, N. D.; and Hong, M. 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proc. ICML*, 1–14.

Yang, J.; Parikh, D.; and Batra, D. 2016. Joint unsupervised learning of deep representations and image clusters. In *Proc. CVPR*, 5147–5156.

Zhang, R.; Isola, P.; and Efros, A. A. 2016. Colorful image colorization. In *Proc. ECCV*, 649–666.

Zhang, R.; Isola, P.; and Efros, A. A. 2017. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proc. CVPR*, 1–11.

Zhao, J.; Mathieu, M.; Goroshin, R.; and Lecun, Y. 2015. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*.