AMERICAN UNIVERSITY OF BEIRUT

UNSUPERVISED DEEP LEARNING: INTER-MODEL INTERPRETATION AND DEEP CLUSTERING

by
AHMAD MAAMOUN MUSTAPHA

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Electrical and Computer Engineering
of the Maroun Semaan Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
May 2021

AMERICAN UNIVERSITY OF BEIRUT

UNSUPERVISED DEEP LEARNING: INTER-MODEL INTERPRETATION AND DEEP CLUSTERING

by

AHMAD MAAMOUN MUSTAPHA

Approved by:

Prof. Wassim Masri, Professor Electrical and Computer Engineering	Advisor  May 7, 2021 *Signing on behalf of all committee members
Prof. Wael Khreich, Assistant Professor Suliman S. Olayan School of Business	Co. Advisor
Prof. Hazem Hajj, Professor Electrical and Computer Engineering	Member of Committee
Prof. Mariette Awad, Professor Electrical and Computer Engineering	Member of Committee

Date of thesis defense: May 3, 2021

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name:	Mustapha	Ahmad	Maamoun
	Last	First	Middle

⊗ Master's Thesis ○ Master's Project ○ Doctoral Dissertation

☐ I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

☒ I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One ___ year from the date of submission of my thesis, dissertation or project.**
Two ___ years from the date of submission of my thesis , dissertation or project.
Three X_ years from the date of submission of my thesis , dissertation or project.

Signature *Ahmad M. Mustapha* Date 08 - 05 - 2021

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

Acknowledgements

I want to thank my research co-supervisor Prof. Wael Khreich for all the time he invested in mentoring me and tolerating my stochastic mood shifts. Prof. Khreich came from traditional machine learning background where rigor experimentation is still a principle. He pushed me to apply the same principle in my research obsessing over every hyperparameter, asking for ablations, requiring to run every experiment multiple times with randomization, and trying to know why a neural network learns what it learns. This has set my progress in a rabbit hole for a while but it eventually paid off triumphantly!

I want to thank my advisor Prof. Wes Masri. Prof. Masri has a relaxed and composed demeanor that I have never seen before in a researcher who maintains a high research profile, impactful publications, and exposure in his field. Should I become a full-time researcher I want to be like him. I want to thank Prof. Hazem Hajj and Prof. Mariette Awad for being part of my master committee. Both unarguably play a major role in AUB's Artificial Intelligence research leadership, and they are doing it with style and distinction.

And most importantly, I want to thank my parents Maamoun and Samia. They have graciously accepted the fact that their son is doing yet another master degree rather than supporting himself and the family. They have always been a constant inspiration and motivation.

An Abstract of the Thesis of

Ahmad Maamoun Mustapha for Master of Engineering
Major: Electrical and Computer Engineering

Title: Unsupervised Deep Learning: Inter-model Interpretation and Deep Clustering

Deep Learning has risen recently as the de-facto machine learning approach for complex and rich-data domains. However, this has been established so far mostly in the setting of supervised learning. Supervised learning requires a tremendous amount of labeled data that is expensive, time-consuming, and not scalable. To overcome those limitations, researchers have invested recently in unsupervised deep learning approaches which make use of the huge amount and easily available unlabeled data. In this dissertation, our work in the domain of unsupervised deep learning was two-fold. We first apply a novel model interpretation technique to study the learned concepts of different unsupervised deep learning approaches and how they compare to each other. Second, we considered one of the proposed approaches, Deep Cluster, proposing an explanation on why it works, supporting it with experiments, and following it by a series of ablation experiments to understand some parameters-performance dynamics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since Krizhevsky [1] first demonstrated the superiority of Deep Convolutional Neural Networks in image classification tasks over the ImageNet dataset, and the field of deep neural networks has been exponentially growing. Deep learning have been proven to be highly performant in different complex and rich-data domains. Such in computer vision, speech recognition, natural language processing , etc. It have been able to mimic mappings of complex nature such in autonomous vehicles, and robotics.

However, all this have been achieved through supervised learning in which the deep model is fed with training data along with targets. Targets can be of any form such as labels in image recognition, and bounding boxes in object detection. Those targets are usually generated and crowd sourced by human annotators which is a time consuming, expensive, and unscalable process.

Training of CNNs requires a considerable amount of labeled data. Collecting such an amount of labeled data is exhaustively expensive, unscalable, unpractical for individuals/organizations with limited resources, and it also doesn't make use of the massively large and easily available unlabeled data. This called for a new unsupervised deep learning approach that learns strong hierarchical visual feature representations with no need for labels.

Although the publicity of supervised Deep Learning has shadowed the unsupervised deep learning movement, unsupervised deep learning has always existed alongside supervised learning. While Yann Lecun [3] was working on applied supervised deep learning using a convolutional neural network, Geoffrey Hinton [4], [5] was working on learning representations without supervision using generative models. When his student Krizhevsky was working on AlexNet for ImageNet, Hinton was resistant to the idea as he thought that machines should learn labels by themselves.

Since 2015 the domain of unsupervised deep learning has gained researchers' attention and the number of research papers quickly increased. The approaches tackling the domain started to form sub-domains each tackling it from a different perspective. We can divide the domain into 4 categories of research: (1)

Clustering-Based, (2) Sample-Specificity-Based, (3) Self-Supervised, and (4) Generative Models. Note that those categories are not exclusive and they do overlap. In recent years self-supervised deep learning approaches achieved competitive results compared to supervised learning [6] casting previous unsupervised approaches out of favor.

Given the unsupervised nature, one would wonder what approach is used to benchmark and compare different unsupervised deep learning approaches. The current state of the art approach is to train models in an unsupervised context by solving a challenge that make use of the unlabeled data (we will call this a *pretext task* in what follows) resulting in a pretrained model. Then a simple machine learning model will be trained over the frozen features of the pretrained model in a supervised context over a given task (we will call this a *downstream task* in what follows). The performance on the downstream task is considered as an evaluation of the unsupervised training approach.

Summarizing performance by a couple of evaluation metrics such as accuracy doesn't tell so much about how two models are similar or different. A model achieving a higher accuracy than another model on one downstream task might perform poorer on another. In this manuscript we explored a complementary approach that outputs fine grained details about the actual semantic concepts learned by the model by directly inspecting the learned features. The approach is rooted in the interpretable deep learning models domain and is based on a recent model interpretation technique called Dissect [7]. We introduce the notion of *inter-model interpretation* which studies how a group of models are similar or different and whether they complement each other or not. Our experiments allowed us to understand what type of concepts each downstream task requires. And wether the models complement each other or not. This allows for the formulation of better ensembles by utilizing the complementary nature of the models.

On the other hand, we investigated a recent Clustering-Based Unsupervised Deep Learning technique called Deep Cluster [8]. The technique has shown to be very competitive achieving a high transfer learning score on ImageNet [9] and Places [10] Datasets and still competes with newer techniques. Deep Cluster is end-to-end, scalable, and interestingly simple. It follows the mainstream supervised learning setting, while different only in the fact that the labels, which are not available, are generated through clustering the entire dataset and using assignments as pseudo-labels.

We proposed an explanation of why Deep Cluster works based on the initial alignment of the first pseudo-labels with ground-truth. We found that it is highly sensitive to random initialization and to the number of clusters parameter. We followed with a series of ablation studies to understand more the parameters-performance dynamics. One of the main finding is that a key component in Deep Cluster pipeline, which is clustering, doesn't add value after a certain number of epochs.

The rest of the dissertation is structured as the following: Chapter 2 positions

the research and pass through related literature. Chapter 3 introduce the Dissect interpretability technique and follows up with our experiments on self supervised models and findings. Chapter 4 introduces Deep Cluster in details and follows up with our experiments and findings. Chapter 5 concludes the manuscript.

Chapter 2

Literature Review

2.1 Unsupervised Deep Learning Techniques

As previously mentioned in the introduction, we recognize four major directions in the domain of Unsupervised Deep Learning. The movement started initially with Restricted Boltzmann Machines (RBMs) [11] and Deep Belief Networks (DBNs) [5, 12] as generative models to extract hierarchical representation of data. Then by 2015 the Clustering-Based approaches followed along with Self-Supervised movement happening in parallel. A handful of Sample-Specificity-Based approaches have been published. The sample specificity approaches forked a new self-supervised approach under the term of contrastive learning.

2.1.1 Clustering-Based Approaches

Clustering-Based Unsupervised Deep Learning simply extends traditional clustering methods by adopting neural networks to learn clustering-friendly representations. We can see them as a neural network that incorporates two losses rather than one, a Network Loss and a Clustering Loss. A hyper-parameter is usually introduced to trade-off between the losses:

$$L = \lambda L_c + (\lambda + 1)L_n$$

The choice of the network loss and the clustering loss leads to a variety of approaches. Several clustering losses have been used, Yang et al [13] used k-means loss, Xie et al [14] used clustering assignment hardening loss, Yang et al [15] used agglomerative clustering loss, Haung et al [16] used locality-preserving loss and group sparsity loss along with k-means, Chen et al [17] used non-parametric maximum margin clustering, etc.

From the Network loss perspective the majority used Autoencoders reconstruction loss (see Figure 2.1) [13, 16, 18, 19, 20], others used Variational Autoencoders (VAE) and Generative Adversarial Networks (GANs) losses [21, 22, 23, 24].

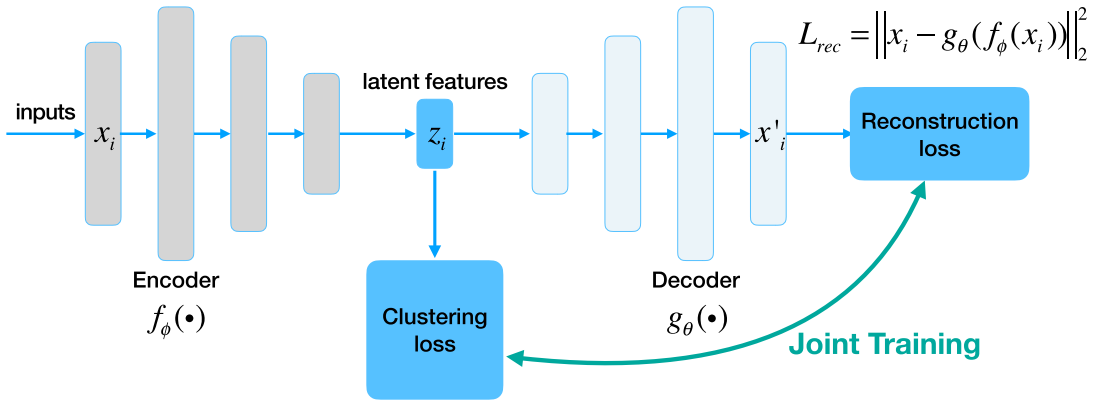


Figure 2.1: Autoecnoder-Based Deep Clustering. Image from [1]

On the other hand, some neglected the network loss and used cluster assignments to fine-tune the network [8, 25, 14, 15]

A comprehensive survey for clustering-based approaches can is provided by Aljalbout et al [26, 27].

2.1.2 Self-Supervised Approaches

In a parallel community to Clustering-Based Approaches, Self-Supervised Deep Learning Approaches have been also gaining attention and momentum. Rather than introducing new losses to deep models, self-supervised approaches generate pseudo labels for the models to be trained on. The idea is to propose a pretext task – a task that doesn't require human annotation (i.e. coloring) - for the network to solve. By training the network on the proposed task objective functions it learns features that are relevant for an actual downstream task.

To the date of writing this manuscript, several pretext tasks have been proposed. Recently, Jing et al [1] have compiled a comprehensive survey on self-supervised learning. As the range of proposed tasks varies considerably we will stick to the tasks that solely depend on 3-channel Images. In other words, we will ignore the ones that depend on videos, artificially generated data (using game-engines), artificially generated labels (using hardcoded programs), and more-than-3-channels images (e.g. images with depth).

Initially Doersch et al[28] have proposed to train the network to predict the relative position between image patches (see Figure 2.3). Larsson et al [29] and Zhang et al , [30] proposed coloring grayscale images. Pathak et al [31] proposed in-painting occluded image patches. Hu et al [32] proposed in-variance against augmentation. Zhange et al[33] proposed cross channel prediction (color from grayscale and vice-versa). Wang et al [34] proposed preserving transitivity between instance inter-variance (same class different instance) and instance

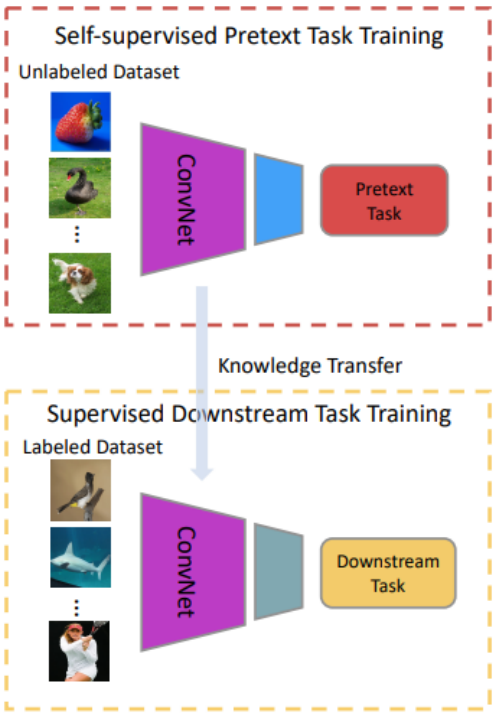


Figure 2.2: Self-supervised deep learning general workflow. Image from [1]

intra-variance (same instance different viewpoints, illumination, ...). Noroozi et al [35, 36] proposed solving a jigsaw puzzle and counting objects. Gidaris et al [37] proposed predicting image rotation.

To evaluate the performance of an SSL algorithm, the learning goes as following: (1) The model is pretrained in an unsupervised fashion over the pretext task. (2) The learned features are frozen and then used in a supervised setting over what is called a downstream task. The task is usually training a linear classifier but other tasks such as object detection or segmentation are performed also. The best SSL algorithm is the one that outperforms the others on a wide range of downstream tasks. As in Figure 2.2.

2.1.3 Generative Models Approaches

Generative models were the first to be proposed in the domain of unsupervised deep learning. Rather than learning by discrimination, the models are trained to reproduce images and are penalized by the quality of generated images.

Earlier methods included Deep Belief Networks (DBNs) and Restricted Boltzmann Machines RBMs [12, 4, 38, 39, 40]. DBNs have fallen out of favor and are rarely used [41] compared to other Generative Models such as Autoencoders [42, 43], Variational Autoencoders [44], and Generative Adversarial Networks (GANs) (see Figure 2.4) [45, 46, 47].

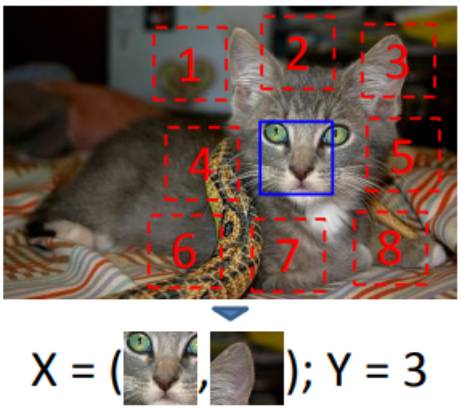


Figure 2.3: Learning by context prediction. The network is trained to predict the relative positions between image patches. Image from [1]

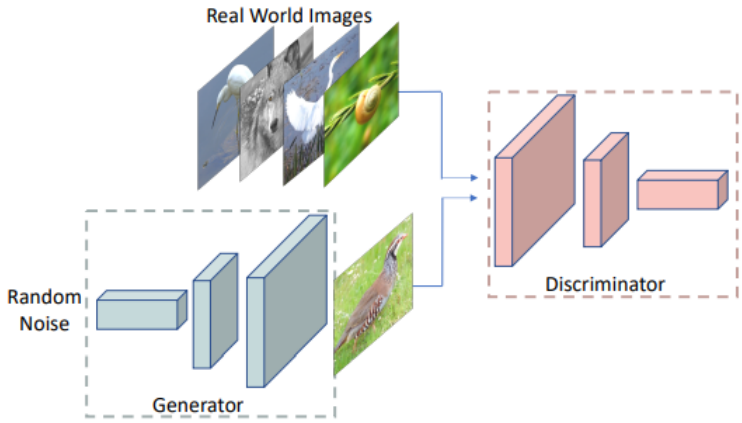


Figure 2.4: Generative Adversarial Network (GAN). Image from [1]

Although the motivation behind these approaches is to model the data and be able to generate it, the representation learned during training generative models can be exploited for downstream tasks.

2.1.4 Sample-Specificity Based Approaches

Sample-Specificity-Based approaches have gone with clustering to the extreme where they consider each instance a class that the network should learn to discriminate. The number of publications concerning this approach is small.

Initially Dosovitskiy et al [48] proposed Exemplar-CNN in which they extract patches from images and then create a surrogate class for each patch by augmenting it. The model is trained to discriminate between those surrogate classes. Bojanowski et al [49] proposed to map image instances to a fixed dis-

tribution of random vectors. Wu et al [50] directly transferred the supervised learning approach by giving a label for each instance however they used a non-parametric softmax alternative and approximated it using noise-contrastive estimation (NCE). This paper helped inspire a new self-supervised category of approaches termed "Contrastive Learning".

2.2 Studying Self Supervised Models

The work we present in this manuscript doesn't introduce a new unsupervised representation learning approach but it analyses an existing one - Deep Cluster. It aligns with the work of other researchers who also studied self-supervised models. Asano et al [51] studied the effect of minimizing the dataset size and quality on the performance of different approaches and found that it is possible to maintain the same performance on the lower layers of the network while training on one rich image. Caron et al [52] studied the performance of Deep Cluster itself on non-curated data. Goyal et al [6] studied the effect of scaling datasets on different approaches. Kolesnikov et al [53] studied the effect of different choices considered in the learning pipeline like models and evaluation. Bansal et al [54] hinted at the interesting small generalization gap between training and testing performance that characterizes self-supervised models.

2.3 Deep Learning Models Interpretability

After Deep Learning applications spurred in safety-critical applications, concerns have risen about neural networks' black box nature and lack of interpretability. In response to these concerns, the neural network's interpretability domain started to form. The domain has grown significantly and diverse in recent years.

The mainstream interpretability are: a) feature visualization which tries to generate examples that maximally activate a part of the network let it be a unit, a channel, a layer, or class probability [55, 56, 57]. b) feature attribution which tries to explain which part of the image contributed mostly to the network decision [58, 59, 60]. A survey on visual interpretation of CNNs is presented by Zhang et al [61]. And Olah et al provided an interactive toolbox [62]

Other researchers went on to explain predictions by case, i.e. by proposing a case that is believed by the targeted neural network to be closest to a query case [63, 64]. Or by generating textual explanation [65, 66]. A comprehensive survey is presented by [67].

As far as we know we are the first to consider inter-model interpretability. However, we made use of the recent interpretability technique "Dissect" [7].

Chapter 3

Inter-model Interpretation: Self-Supervised As a Use Case

3.1 Motivation

The current established approach for evaluating Deep Learning algorithms is to measure how they perform on certain task based on metrics such as accuracy or precision. While such metrics are useful they doesn't capture the full picture of how two models are similar or different. In other words two models that performs similarly from accuracy perspective might have learned different different complementary concepts. Knowing such information will help us build better ensembles or understand what combinations of concepts (objects, materials, colors, etc) different downstream tasks or dataset requires. For instance, recently it have been shown that increased performance on ImageNet doesn't add any value to performance over a medical set when transferred. This means that the medical dataset requires different nature of learned concepts [68].

To address this problem we hereby introduce the notion of *inter-model interpretability*. Unlike the traditional interpretability efforts that studies how a model makes it's decision, what are the learned concepts, or what fraction of the input that drives the decision [67], inter-model interpretability studies how a group of models relates to each other. Have the models learned similar concepts? Does the learned concepts complement each other or redundant?

To investigate this interpretability we build upon a recent approach called "Dissect" [7]. Dissect scalably inspects each unit (neuron) in a given network to understand the semantic concept it has learned and assign it to it. Concepts such as objects like horses and cars or materials such as fur and skin. It is possible to aggregate all the concepts learned by units in the final layer to come up with a learned concepts profile. We made such information for use by projecting models into a Learned Concepts Embedding (LCE) that reveals the dissimilarities and similarities between the models from the learned concepts perspective.

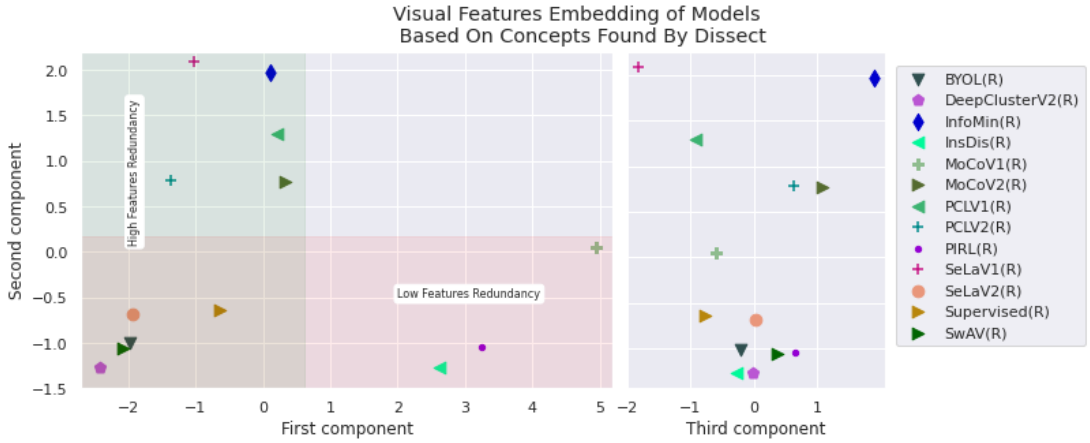


Figure 3.1: The Learned Concepts Embedding (LCE) space of the 13 top performing self-supervised models based on their dissect profile. Each dot represents a model. Three main groups emerges in the space.

3.2 Dissect

Dissect [7] is a model interpretation technique that directly inspects the semantic knowledge a certain filter (unit) in a given network holds. To understand what activates a unit a current practice in the literature is to feed the network a set of random images and manually inspect the images that highly activated it. This qualitative approach is useful but not scalable as it needs human interaction. The approach presented by Dissect scales this out by using a pretrained image segmentation model to replace humans inspection.

Consider the input image in figure 3.2 and the corresponding up-scaled activation of a certain neuron on this image. It is evident that the neuron is looking for tree-like structures. But how would Dissect know that? It uses a pretrained image segmentor that segments the input image into different concepts, in the figure it is segmenting the tree concept. It then computes the intersection over union (IoU) ratio between the neuron activation and the segmentor segmentation. To avoid chance scenarios it applies the same procedure to a large number of images in a validation dataset. If the IoU ratio is high over many images then this neuron is probably looking for the corresponding concept. Dissect computes the IoU ratio against 1,825 concepts available by the segmentor. And designate the concept that has the highest ratio to the neuron. If the highest ratio is under a certain value, an IoU threshold hyperparameter, the neuron is reported to not belong to any concept. Check Appendix A.0.1 to see concepts that achieved top IoU for each considered model with images showing activations.

What we care about in this paper, is that given a certain layer in a certain pretrained network, Dissect will output all the concepts that it captured in this

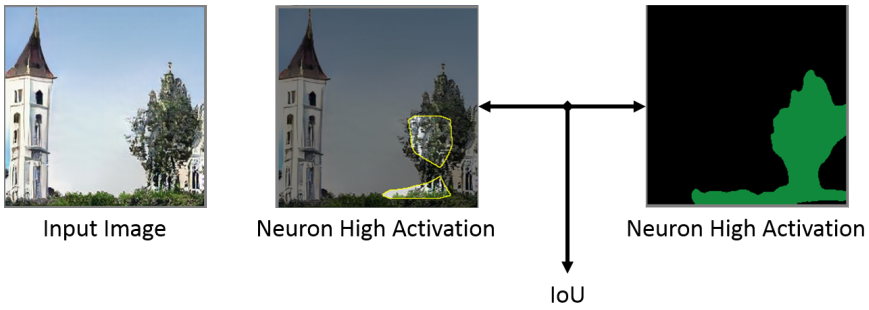


Figure 3.2: Dissect Basic Building Block

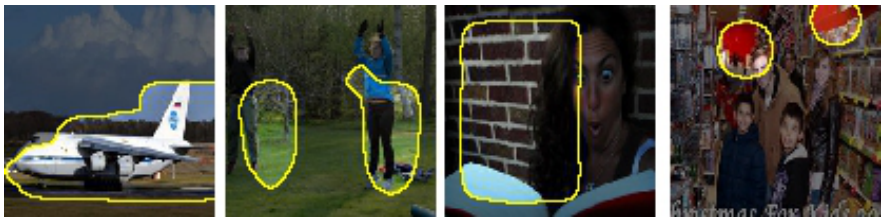


Figure 3.3: Example of unites activating over different concept categories from left to right: airplane(object), leg(object part), brick(material), red(color)

layer along with the number of units that corresponds to each concept. The Dissect version we used in this paper captures concepts of 4 classes: (1) Objects such as horse and person. (2) Object parts such as mountain-top and person-left. (3) Materials such as fur and skin. (4) Colors.

3.3 Experiment Setup

In our experiment, we applied Dissect on 13 models pretrained using top-performing Self Supervised algorithms. All the models use ResNet50 [69] as a backbone and are pretrained over ImageNet. We also considered a ResNet model trained in supervised fashion over ImageNet for comparison and a random one. We used Dissect authors code as is including the validation Places dataset [10], the Unified Perceptual Parsing image segmentation network [70], and the IoU threshold as 0.04. We considered in our experiment only the final layer in ResNet50 of size 2048 in all models.

The models used in our experiment are InsDis [50], Moco V1/V2 [71], PIRL [72], SimCLR V1/V2 [73], InfoMin [74], and BYOL [75] as contrastive SSL. PCL V1/V2 [76], SeLa V1/V2 [77], DeepCluster V2 [78], and SwAV [78] as Clustering SSL.

To enrich our data we crossed the generated Dissect profiles of the models with there performance on a set of different downstream tasks and datasets from [79]. The Datasets used are: FGVC Air-craft [80], Stanford Cars [81], Oxford 102 Flowers [82], Food-101 [83], and Oxford-IIIT Pets [84] as many-shots fine-grained object classification. ImageNet, Caltech-101 [85], CIFAR-10 [86], CIFAR-100 [86], and Pascal VOC2007 [87] as many-shots coarse-grained object classification. DTD [88] as many-shot texture classification. And SUN397 [89] as many-shot scene classification. CropDiseases [90], EuroSAT [91], ISIC2018 [92] and ChestX [93] as few-shots classification reporting on 5-10-50 shots for each. Pascal VOC2007 [94] as detection reporting Ap, 50AP, 75AP with and without fine-tuning. NYUv2 [95] as dense prediction of surface normal estimation. The reader is forwarded to [79] for detailed training procedures.

3.3.1 Downstream Tasks

We hint below to the definition of the four different downstream tasks considered.

Many Shot Classification. The widely targeted learning setting in which a model learns to classify a set of inputs into two or more classes while having an abundance of labeled inputs. They are usually trained by minimizing the cross-entropy between predictions and ground truth.

Few Shot Classification. A learning setting in which a model learns to classify a set of inputs into two or more classes while having a very limited set of labeled inputs usually ranges between 5 to 50 for each class. It considers the concept that reliable algorithms should be made to perform well using a small amount of training data. Here we consider prototypical networks in which pre-trained models are used as a non-linear mapping of the input into an embedding space and take a class's prototype to be the mean of its support set (training set for each class) in the embedding space. Prediction is done, given an embedded query point (an unlabeled input), by simply finding the nearest class prototype.

Object Detection. A learning setting in which a model learns to not only classify objects in images but also detect their bounding box. Different evaluation metrics exist each gives different priority for classification and localization.

Surface Normal Estimation. A learning setting in which a model learns to assign the normal for each pixel in an image. Given, for example, a 2-dimensional image of a table. The tabletop pixels will be assigned a normal perpendicular to the x-y plane.

3.4 Learned Concepts Embedding

In order to compute the LCE, we first found the super-set of all concepts found by Dissect in all aforementioned models. We then represent each model by a vector where each dimension represent a concept in the super-set and its value represents

the number of units Dissect matched to the concept in the model's last layer. We call this vector for each model a *Dissect Profile*. We finally normalized the values with the total number of concept found for each category. The super-set included 144 concepts distributed as following 67 objects, 54 object parts, 16 materials and 8 colors. Formally consider that we have a set of models $M := (m_1, m_2, ..,)$, a super-set of concepts $C := (c_1, c_2, ..)$. Let $D(m_i, c_j)$ be the number of units found by Dissect in model m_i that is matched to concept c_i. Let (t_o, t_p, t_m, t_c) be the total number of objects, object parts, materials, and colors respectively that composes the super-set C. Let $cat(c_i)$ be a mapping to the concept c_i category. Then the LCE will be computed over the following matrix:

$$\begin{bmatrix} \frac{D(m_1,c_1)}{t_{cat(c_1)}} & .. & .. & .. \\ .. & .. & .. & .. \\ .. & .. & \frac{D(m_i,c_j)}{t_{cat(c_i)}} & .. \\ .. & .. & .. & .. \end{bmatrix}$$

We computed a LCE through Principle Component Analysis (PCA) and considered only the first three components. The components explained 88% of the total variance. The scatter plot of the embedding can be seen in Figure 3.1 along with the different models considered in our experiment.

3.5 Results

3.5.1 Dissect Profiles

The abstracted Dissect profile of the considered models can be seen in figure3.4.

The Random model has a simple profile composed of a handful of concepts. However, visually inspecting corresponding unites activations shows that they are random and don't hold any semantics. Those random activations tend to intersect with segments of frequently occurring concepts such as sky, floor, and person. This points out a major weakness in Dissect algorithm. On the other hand, SimCLR models have yet a smaller profile than a random model. Again with found concepts being of coincidence. It seems that SimCLR encodes information in a different way than the other models. For the rest of the models, the number of unique concepts found doesn't vary a lot across the majority of models. MoCoV2 and SeLaV2 are leading. Insdis, PCLV1, and SeLaV1 are trailing.

The models differed more in the total number of concepts assigned to units. Multiple units can be assigned the same concept. This doesn't necessarily mean that all those units capture only the corresponding concept. It means that those have an activation pattern that mostly intersect with the concept but might also capture other concepts. For example, consider a unit that activates for blue flat surfaces. The unit will activate for swimming pool as well as for pool (the game)

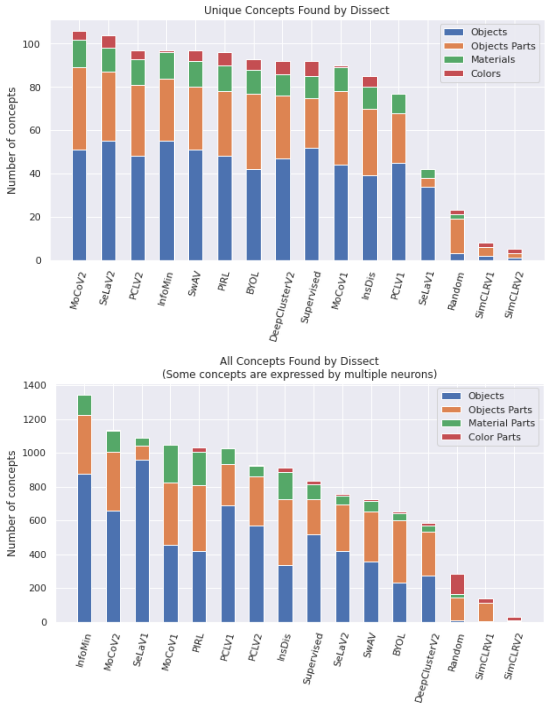


Figure 3.4: The number of unique/all concepts in each category found by Dissect for each model

tables. However Dissect end up assigning the concept to the swimming pool as it intersects mostly with it possibly due to the fact that it appears more in the validation dataset or that the segmentor doesn't have pool table concept. As far as we are applying the same procedure on all models the comparison is still valid. The less redundancy (multiple units for a concept) a model's Dissect profile has, the more the model units are specialized and hold more abstract detectors.

3.5.2 Explaining The Embedding

To understand the semantics of the three main axes of variance. We computed the correlation between each axis (the principal components) and an *abstracted Dissect profile*. The abstracted profile aggregates the concepts into their categories. The aggregations are the unique number of concepts in a category, and the total number of units in a category. The results are found in figure3.5.

The first component is highly correlated with the total number of materials and to the total number of object parts with a lesser degree. The same pattern occurs for the second component but it considers objects and color. On these axes, the number of unique concepts found by Dissect doesn't matter as much as how many units represents a given concept. The lower the number of units that

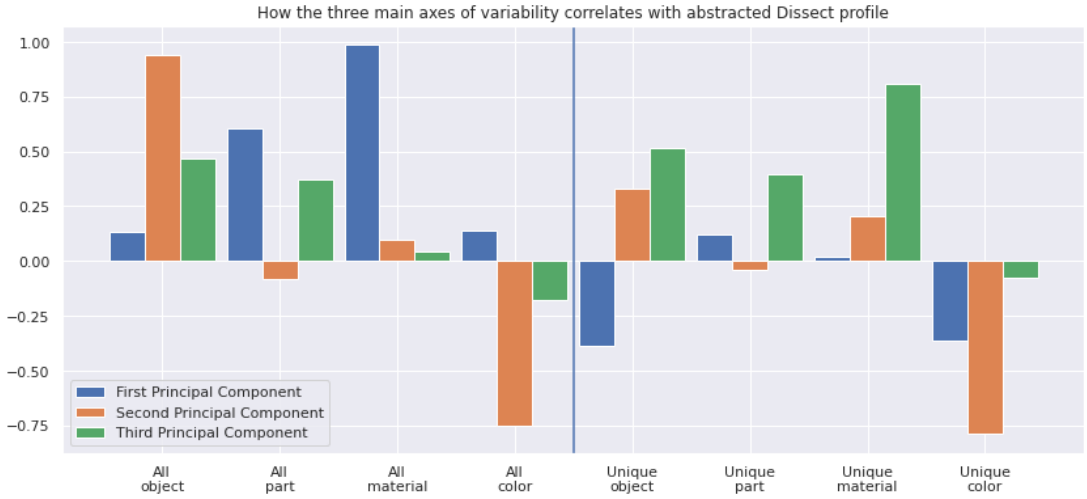


Figure 3.5: Correlation between abstracted Dissect profile and principal components

tackle a unique concept the model will be placed on the lower region. Moreover, the second component have a negative correlation with color which means the lower it is the more the model has color information. The third component correlates mostly with the number of unique materials found. Unlike the first two components, it considers the number of unique concepts to a certain degree.

In summary, the first component looks for materials and object parts and the second component looks for objects and colors. The variance on both axes represents how many units are matched to the corresponding categories of concepts. The third axis looks for the number of unique concepts found with more emphasis on materials.

Reading The Embedding

The projection of the different model's Dissect profiles resulted in a space of three clusters. The first cluster (A) resides in the lower region of the two principal components. Those models maintain their proximity to the third component. The second cluster (B) residing in the higher end of the first component and the lower end of the second component. These models are rich with low-level features suitable to detect low-level concepts such as materials. (C) residing in the higher end of the second component and the lower end of the first component. These models are rich with mid-level features suitable to detect high-level concepts such as objects. Nonetheless, both regions fall short for high-level abstract features that are available in region (A). The models in (A) have less redundancy which can be explained in light of Dissect as having more abstract semantic features.

Redundancy is a Dissect artifact however it reflects certain types of feature

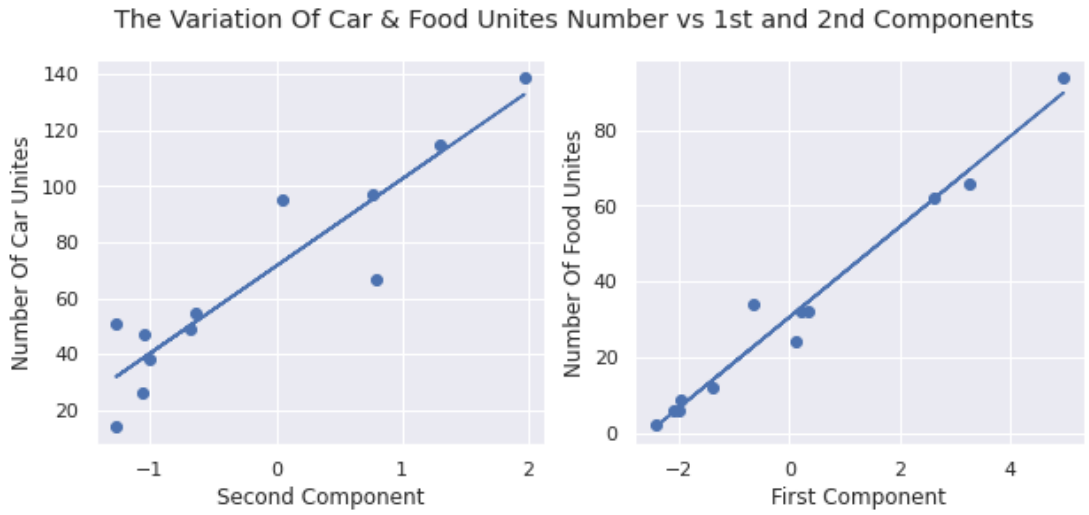


Figure 3.6: How the number of food and car unites varies with the 1st & 2nd components respectively.

detectors. To emphasis on this more, consider the concept of car(object) and food(material). In figure 3.6 we plot how the number of neurons matched to the food and car concepts varies with respect to the first and second components respectively. Some models have more units that detect cars because the model hasn't matured to a certain level of abstraction. In other words, units that detect wheels, smoothed edges, mirrors, or geometric shapes will end up being matched by Dissect to car concept for they mostly intersect with car concept. The same is true for the food concept on the second component.

From an algorithmic perspective Cluster (A) is dominated by clustering-based approaches mainly SeLaV2, SwAV, and DeepClusterV2. It also contains BYOL and Supervised. All these models learn by grouping instances despite the nature of grouping (clusters, labels, or augmentation) without using negative samples. The other approaches spread in clusters (B) and (C) are contrastive approaches that do use negative samples. Except for SeLaV1 which is clustering-based and precursor to SeLaV2. All the V2 models are derived from V1 models mainly by adding projection heads, more augmentations, and more training epochs. This operation has moved SeLa from the cluster (C) to cluster (A), MoCo from cluster (B) to (C), and has moved PCL mostly on the third component.

3.5.3 What combination of concepts each task require?

We computed the correlation between the three principal components and models' performance over different learning tasks and datasets. The goal of this experiment is to understand the importance of each embedding axis on different types

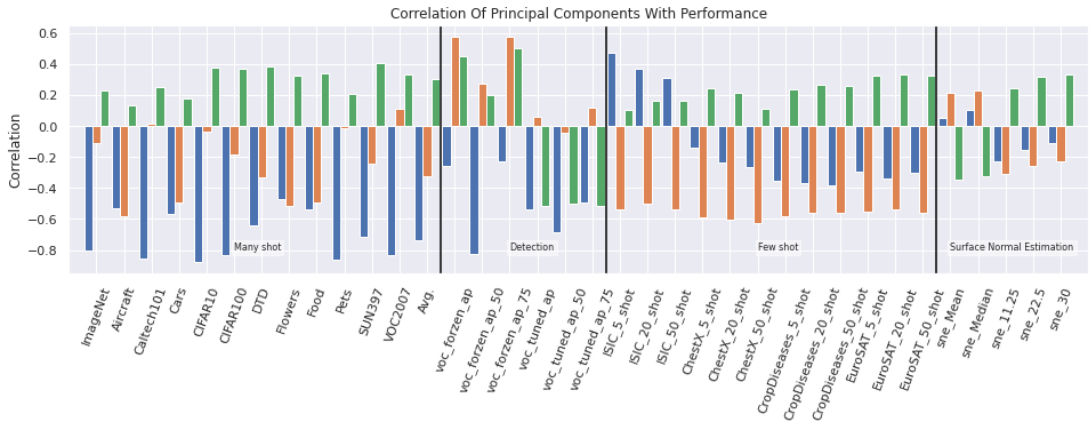


Figure 3.7: Correlation between embedding directions and performance

of tasks. In other words, to what combination of concepts each task require. The result are as in 3.7.

For Many Shot classification tasks performance has a negative correlation with the first component mainly. For such tasks, low-level features are not of importance. When the task is fine-grained such as in Aircraft and Cars, performance becomes less correlated with the first component and more correlated with the second component.

For detection, however, an interesting insight emerges. When we are training with an emphasis on classification more than on localization i.e. using AP50 metric, performance is correlated negatively with the first component. However, when we give more importance to localization using stricter metrics such as AP and AP75 performance become positively correlated with the second and third component. In other words, precise localization requires mid-level features. When we consider fine tuning low-level and mid-level features become less important.

On contrary to many shot classification, few-shot classification is more negatively correlated with the second component. For such tasks, mid-level features result in lower performance. ISIC dataset in particular has a weak positive correlation with the first component. The means low-level features are important for the dataset. This is expected for the ISIC dataset, skin lesion images, is visually very different from natural images in ImageNet.

Performance on surface normal estimation has a weak correlation with all components. This task requires certain features that are not captured by Dissect. Features such as edges and depth of field.

3.5.4 How components complement each other

In this section, we try to find whether we can use the LCE to derive ensemble selection. For each couple of models, we build a weighted soft voting ensemble. The models' probability predictions are weighted based on the performance of individual models. Let a, and b with $a > b$ be the performance of models m_1 and m_2 respectively. Then the prediction of models ensemble

$$E_{m_1,m_2}(X) = (0.5 + |a - b|)m_1(x) + (0.5 - |a - b|)m_2(x)$$

. We build an ensemble for each couple of models and tested their performance on DTD, Aircraft, and Caltech101 datasets. In figure 3.8 we report the results. The ensemble's performances are aggregated over the corresponding groups A, B, and C. In other words, all the possible ensembles between group A models and group B models are built and the corresponding change in performance is computed for each. We then report the average of change into the cell (A, B).

The results are as in figure 3.8.

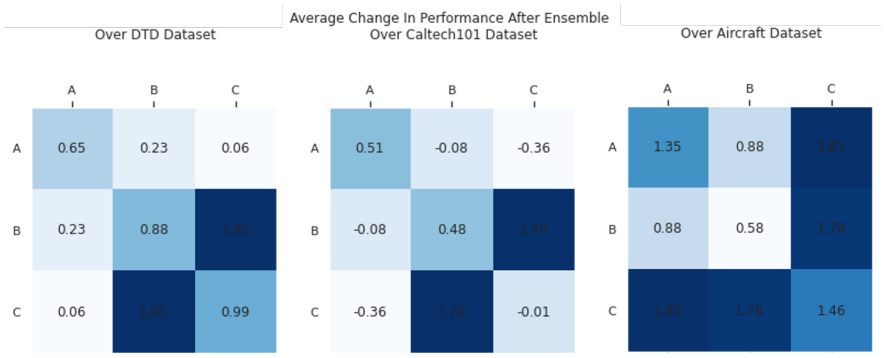


Figure 3.8: The Performance Gain/Loss After Ensemble

Interestingly the models in (B) and (C) apparently result in increased performance over the three datasets. This is means that the features in the two groups complements each other. For the Aircraft dataset, which is a fine-grained one, the ensembles between models in group A and group B increases performance too. This means that for fine-grained datasets mid-level features tends to be useful.

Chapter 4

Revisiting Deep Cluster

4.1 Motivation

Deep Learning has been topping benchmarks leader boards across different domains. Domains such as image recognition [96], speech recognition [97], activity recognition [98], image generation [99], etc. In computer vision, this success has been charged by ImageNet [9]. When an application dataset is limited and small, practitioners bootstrap their learning routines with pretrained models trained over ImageNet.

However, pretrained models over ImageNet holds features that are limited to the classes present in the dataset and don't generalize to non-natural images - except for the lower layers that learn generic Gabor filters [100, 68]. Moreover, settling this problem by pretraining on larger datasets is not efficient as labeled data is expensive, especially if domain expert annotators are needed. On the other hand, unlabeled data is cheap and freely available. Is it possible to have scalable unsupervised pertaining of models?

Unsupervised deep learning has been recently gaining researchers' attention. A recent approach is Deep Cluster [8]. Deep Cluster is significant for its simplicity and scalability. However, as far as we know, no one has tried to do analysis regarding why Deep Cluster works and what is the effect of different hyperparameters on its performance. In this manuscript, we try to understand the internals of Deep Cluster and how different hyper-parameters effect performance. This is important, especially that several concerns have been raised regarding the lack of ablation studies or hyperparameters optimization in the field of Deep Learning [101]. Moreover, answering those questions will: a) Deepen our understanding of Deep Cluster. b) Allow us to separate it's core elements from auxiliary ones. c) Give us the tools to optimize both its learning and computational efficiencies.

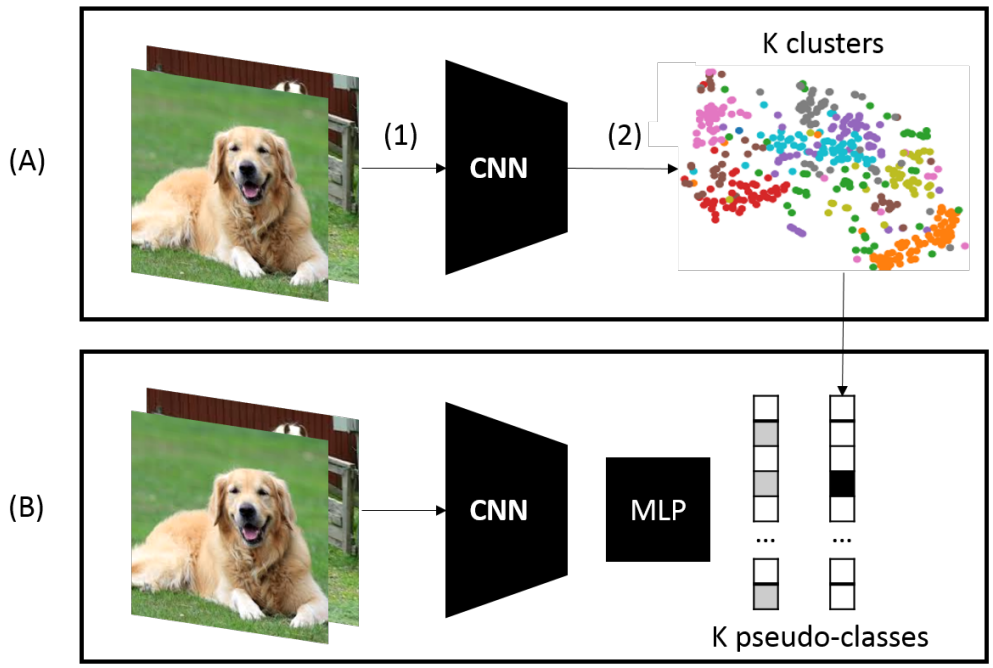


Figure 4.1: Deep Cluster pipeline

4.2 Deep Cluster

Deep Cluster is a very simple pipeline for deep unsupervised representation learning. It follows the same pipeline of supervised learning except for the labels being unavailable and generated on the go.

Figure 4.1 introduce the pipeline of Deep Cluster. The pipeline mainly iterates between (A) generating pseudo-labels and (B) training on these pseudo-labels. We denote N_c to be the number of cycles deep cluster run and N_e to be the number of epochs to train the network on pseudo-labels in each deep cluster cycle. The original authors found $N_e = 1$ to perform well in practice.

To generate pseudo-labels, (1) the entire dataset is fed forward to the network and a feature vector is got for each sample. In other words, the input is projected from Euclidean space into ConvNet feature space. (2) The vectors' dimensions are reduced using principal component analysis and then whitened and L2-normalized. A clustering algorithm - KMeans is used - is then applied to the processed feature space to assign each group of samples into a cluster. The assignments of samples are considered to be the pseudo-labels on which the model will be trained on. The reader is referred to the original paper [8] for more

details.

Statistical learning based computer vision requires a mapping that maps images from their raw space into a space that performs well with training. A space that holds images semantics. Convolutions networks are found to be useful as such mapping and outperform other approaches. Let f_θ be a ConvNet that maps images with θ being it's parameters. The goal of training such networks is to find θ that makes f_θ generates decent general purpose features. Given a set of n images $X = \{x_0, x_1, ..., x_n\}$ with a label y_i for each image x_i. Let $Y = y_i$ for $i \in \{0, 1, ..., n\}$. The network is trained by predicting labels using a parameterized classifier g_w over f_θ. The parameters θ and W are jointly learned through a loss function.

In Deep Cluster the labels are not available but presumed by clustering the ConvNet feature space. In other words for each x_i we compute $v_i = f_\theta(x_i)$, and the clustering is performed on $V = v_i$ for $i \in \{0, 1, .., n\}$. For each x_i we generate a pseudo-label y_i^p being the cluster assignment of the sample. Let $Y^p = y_i^p$ for $i \in \{0, 1, .., n\}$. For each Deep Cluster cycle c a pseud-label vector Y_c^p is generated and the network is trained upon it.

To study the learned feature space quality evolution 2 metrics have been proposed. Both metrics depend on Normalized Mutual Information (NMI), which is a value that is bounded by $[0, 1]$ and represents the dependence between two sets, zero means no dependence and one means that we can predict one set from the other. The metrics are as follows: (1) $NMI(Y_{c-1}^p, Y_c^p)$ which is the normalized mutual information between the cluster assignments of two consecutive clustering. This metrics tells us how much two consecutive clustering assignments are dependent to each other. If this metric converges to a certain value during training, this means that the model didn't collapse. (2) $NMI(Y_c^p, Y)$ which is the NMI between pseudo-labels and ground-truth labels. This metric hints whether Deep Cluster is learning class level features or not. If the dependence between the two groupings (pseudo-labels and ground-truth) is increasing as the model is trained then it must be learning class level features. Note that those metrics are used as a proof of concept as in real-world applications of Deep Cluster the ground-truth is not available. Deep Cluster authors found that as the algorithm executes $NMI(Y_c^p, Y)$ increases which means that class related features are being learned. Also they found that $NMI(Y_{c-1}^p, Y_c^p)$ converge to a certain value after a number of cycles.

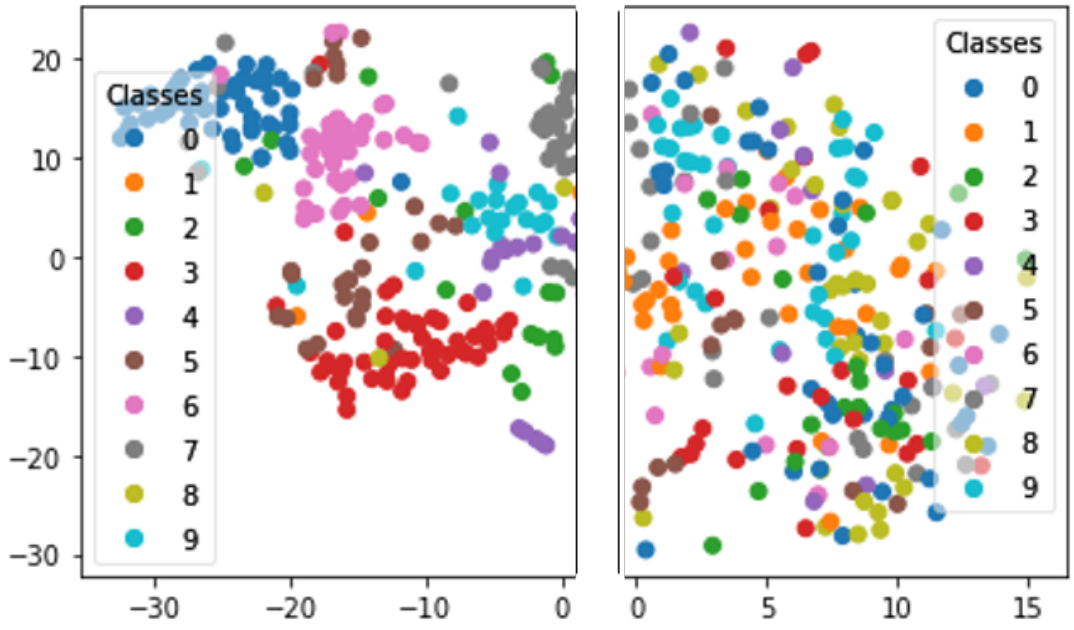


Figure 4.2: The projection of MNIST and CIFAR10 datasets through randomly initialized LeNet and AlexNet respectively. t-SNE [2] is used for dimensionality reduction. Each point represents an image an its color correspond to its ground-truth.

4.3 Why Deep Cluster Works

As far as we know no one have tried to understand why Deep Cluster works out. Intuitively what Deep Cluster do is grouping samples in proximity together and train the model to discriminate between different groups. According to the authors Deep Cluster make use of the randomly initialized filters of the convolutional layers to work as a strong prior to the input signal due to their convolutional structure [102]. It makes sense to think less of the randomly initialized weights of a convolutional layer as numerical matrices, and think more of them as random convolutional filters that hold loose spatial signal.

However, this raises the question on how much does the random filters quality effects the final results of Deep Cluster on downstream tasks or on it's representation learning capabilities and what does the role the number of clusters plays in this process. In this manuscript we argue that Deep Cluster success strongly depends on both the random filters quality and to the choice of the number of clusters. The more the random filters are suitable for the images domain the more projected samples in proximity hold similar semantics. The more number of clusters we use the more feature consistent clusters we get.

Consider the following illustrative example. (A) We fed a sample of MNIST

dataset into a randomly initialized LeNet network. (B) We feed a sample of CIFAR10 dataset into a randomly initialized AlexNet network. In figure 4.2 we plot the final convolutional layer activations for both networks reduced using t-SNE [2] along with the ground truth. The random filters are pretty good for digit images domain as we can see that similar digits fell in the same regions in the features space. This is not the case for CIFAR10, however while similar classes of images doesn't fell in wide regions we can see a handful of them near each other. The number of clusters chosen can then resolve the weakness in the random filters suitability for CIFAR10 classes. Hereby we introduce the notion of *initial alignment* which quantify how much the initial pseudo-labels align with the ground truth. This alignment depends on both the chosen number of clusters and the random initialization of the network weights.

This proposal explains why the original authors found that Deep Cluster performed better on downstream tasks when choosing 10k clusters then when choosing 1k clusters over the ImageNet dataset (which contain 1k classes). And why transforming images using Sobel filter also performed better - it is because the random filters are better as shape detectors than in pattern detectors. We further support this claim with a set of experiments.

4.4 Effects of Initial Alignment

The number of clusters and random initialization of wights (k, θ) plays together regarding the performance of Deep Cluster - θ represents the randomly initialized parameters of a given model. We formulated a metric to measure the suitability of this (k, θ) combination for the targeted dataset. The metric works as following. Let $f_\theta(x)$ be a mapping representing a fully convolutional neural network initialized using θ. Let L be the ground-truth labels of the dataset samples - L is not used during Deep Cluster iterations but we use it here for analytical purposes. Let P be the pseudo-labels obtained by applying KMeans over the dataset features, produced by feeding the dataset into $f_\theta(x)$, using k clusters and considering assignments as pseudo-labels. Note that L takes values between 0 and C where C is the number of classes in the dataset while P take values between 0 and K. Then the metric IA is:

$$IA(\theta, k) = NMI(L, P)$$

where NMI is the Normalized Mutual Information. The higher this metric is, the more the clusters are consistent and align with the ground-truth labels.

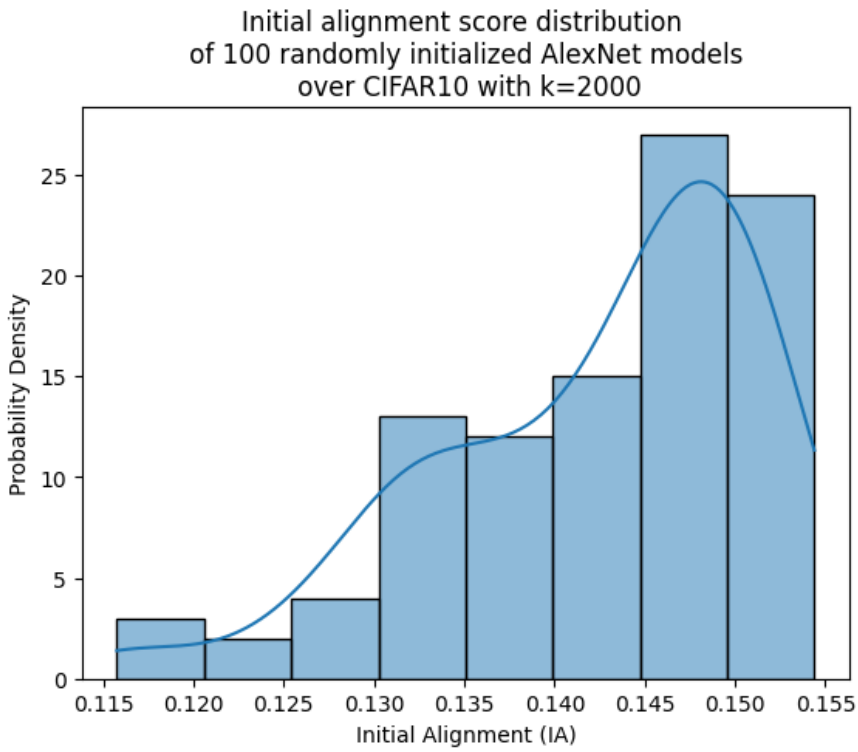


Figure 4.3: Initial alignment score distribution over CIFAR10 using a randomly initialized Al vexNet with k=2000

4.4.1 Changing θ

In this experiment, we measure the effect of changing θ while fixing k on Deep Cluster performance. We considered the CIFAR10 dataset and AlexNet model. We fixed all hyper-parameters with $k = 2000$ except for θ. The model parameters are initialized randomly as following. For convolutional layers weights are chosen from a normal distribution with mean 0 and standard deviation $\sqrt{\frac{2}{n}}$ where n is the layer size $height * width * channels$. Linear layers weights are chosen from a normal distribution with mean 0 and standard deviation 0.01. In case Batch Normalization layers are used there weights are set to 1. All biases are set to 0.

We first inspected the initial alignment score distribution of 100 randomly initialized models. The results are present in figure 4.3. The distribution is right-skewed with the majority of models achieving a score between 0.14 and 0.15. To correlate score with downstream task performance we selected 10 different models that collectively span the entire initial alignment score distribution range. We then measured each selected model downstream task performance by running Deep Cluster for 100 epochs and then evaluating performance over the CIFAR10 dataset as a downstream task. Figure 4.4 shows the results. The higher the

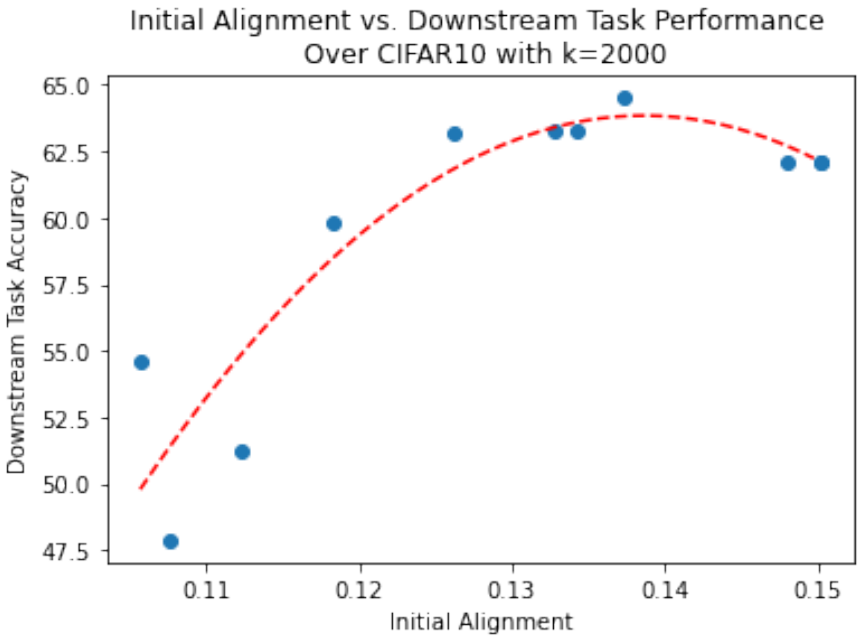


Figure 4.4: Space Quality vs. Deep Cluster Performance

initial alignment is the better Deep Cluster performs. This experiment provides concrete evidence that supports our claim. The take-away of this experiment is that Deep Cluster should be performed multiple times to get an accurate estimate of performance mean and standard deviation. It also calls for heuristics to choose better performing initialization as running Deep Cluster multiple times is very time and resources consuming.

4.4.2 Changing k

In this experiment, we study the effect of changing the chosen number of clusters over performance. For this we considered both FMNIST [103] and SVHN [104] datasets to train a LeNet-5 model and a down-scaled AlexNet model respectively. For FMNIST we varied the number of clusters through (5, 7, 10, 12, 20, 200, and 500) for each number of clusters we run 30 experiments fixing everything except for the random seed. For SVHN we varied the number of clusters through (10, 50, 100, and 500) for each number of clusters we run 5 experiments fixing everything but the random seed. Note that both datasets contain 10 classes of images. The results can be seen in Figure 4.5.

For FMNIST - which is a dataset of down-sampled grayscale images of 10 apparel items like shoes, pullovers, coat, etc - increasing the number of clusters doesn't contribute a lot to the final performance. It is also better to choose a small number of clusters even below the number of classes. This is because of

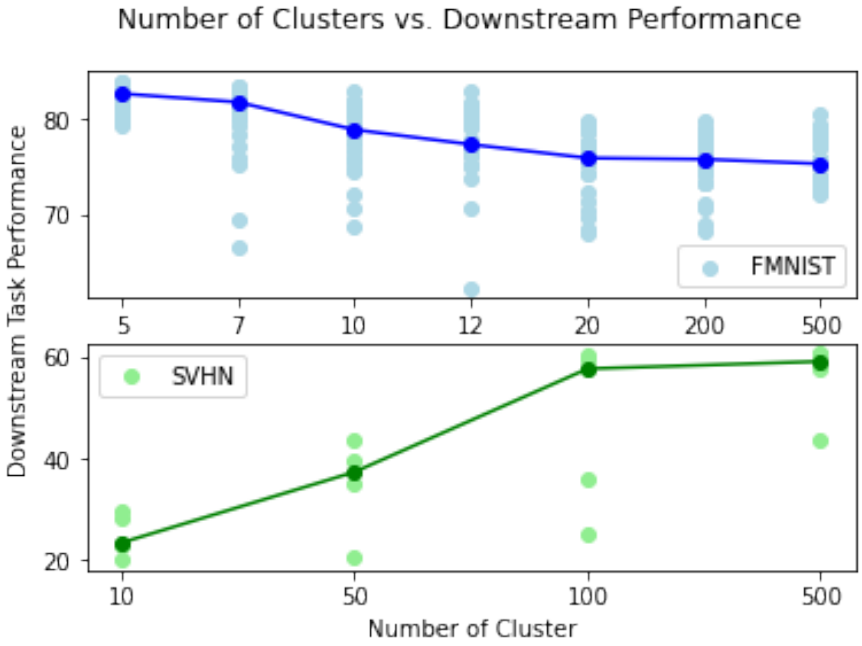


Figure 4.5: Number of Clusters vs. Deep Cluster Performance

the simple nature of the images. Even if different items are grouped together the model is still able to learn meaningful representation and learn the different modalities of the group.

The good performance when using a lower number of clusters is also related to another factor. In the experiment, we run Deep Cluster 30 times for each number of clusters and report the median in the figure. For a smaller number of clusters, the performance variance of the 30 runs is small while for higher numbers it gets larger (This is true for SVHN also). This fact drags the median of performance in a higher number of clusters lower.

The small variance in performance when choosing a lower number of clusters is related to the random nature of k-means. When we have a small number of clusters no matter what random seeds we used the k-means output will converge to a couple of identical groupings. However, when we have a larger number of clusters the output will converge to a much more possible grouping (we can say that using a larger number of clusters degrades k-means to a proximity preserving space partitioning routine). This diversity of groupings will then lead to diverse initial alignment scores and in turn diverse performance. This again hints at Deep Cluster sensitivity to random initialization.

On the contrary to FMNIST, to increase performance on SVHN we need to choose a higher number of clusters. As the number of clusters increases the performance significantly increases. SVHN is a dataset of RGB images of street

Halt Clustering Position	Downstream Task Accuracy
15	62.00
30	64.06
50	65.18
100	66.98

Table 4.1: Clustering Contribution

view house numbers distributed over 10 digits from 0 to 9. The nature of SVHN is much more complex than FMNIST.

This experiment reinforces more the claim that Deep Cluster performance is sensitive to the initial alignment score that is based on both the parameters' initialization and the choice of the number of clusters. It also proves that we should choose the number of clusters based on the images' nature complexity and not the number of ground-truth classes in the first place.

4.5 Clustering Contribution

According to the original authors Deep Cluster converge after some iterations. This convergence is measured by the fact that the mutual information between consecutive pseduo-labels become constant through Deep Cluster iterations after a certain number of iterations. This fact raises a question about clustering contribution in Deep Cluster performance.

To measure this contribution we run multiple Deep Cluster experiments on CIFAR10 Dataset using AlexNet while fixing everything including the random seed. The only difference was halting Deep Cluster clustering part at different cycles. In other words after a certain number of cycles we fix pseudo-labels and only the training part is reserved. After halting Deep Cluster degrades to normal supervised learning setting while using pseudo-labels as ground-truth. We ran Deep Cluster for 150 epochs and halted clustering at 15, 30, 50 and 100. Table 4.1 shows the results.

It turns our that continuous clustering only added about five degrees of accuracy. This finding significantly decrease the computational requirements of Deep Cluster. This is because Deep Cluster performs a full feed forward of the entire dataset and then apply clustering over the features each cycle. Considering ImageNet this means feeding the network approximately 1 million images and then perform k-means over them in each cycle. Further more, the authors reported that the majority of Deep Cluster execution time was in the clustering phase.

PCA	Downstream Task Accuracy
32	64.97
64	64.80
128	65.50
256	65.47

Table 4.2: PCA effect on performance

4.6 Better Performance with Less Computation

In this description we present the experiments we performed in order to quantify the effect of different parameters on Deep Cluster. Although the accuracy improvement might seem small, a 1 percent improvement is considered significant.

4.6.1 Principal Component Analysis

The output feature size of modern ConvNets is relatively large. For example in case for AlexNet the output dimension is 2048. It is very expensive to perform Clustering over such height dimensional space especially that the clustering is being performed over the entire dataset. To resolve this problem the dimensionality reduction technique Principal Component Analysis (PCA) is used. This raise a new hyper-parameter.

To measure the effect of PCA on performance we run deep cluster over CIFAR10 using ALexNet. We set $k = 2000$ and fixed all other parameters except for PCA which is varied between 32, 64, 128, and 256. Table 4.2 show the result. The PCA effect on downstream task accuracy is negligible. A reduction from a 256-dimensions space into a 32-dimensions space significantly speeds up clustering.

4.6.2 Fitting KMeans

In Deep Cluster the clustering module is trained on the entire dataset. This is expensive and to reduce it, is is possible to train the clustering module on a partial number of data and then predict other data labels - in KMeans this means to find the centroids using a small number of data points and then assign the other points using the learned centroides. We performed an experiment to study the effect of fitting KMeans using portion of the data on Deep cluster performance. We found that training the clustering module on significantly small portion of the entire data actually performed better.

The experiment is performed on CIFAR10 using AlexNet with $k = 2000$ while varying only the percentage of data to train KMeans on between 5,10,20, ..., and 100.

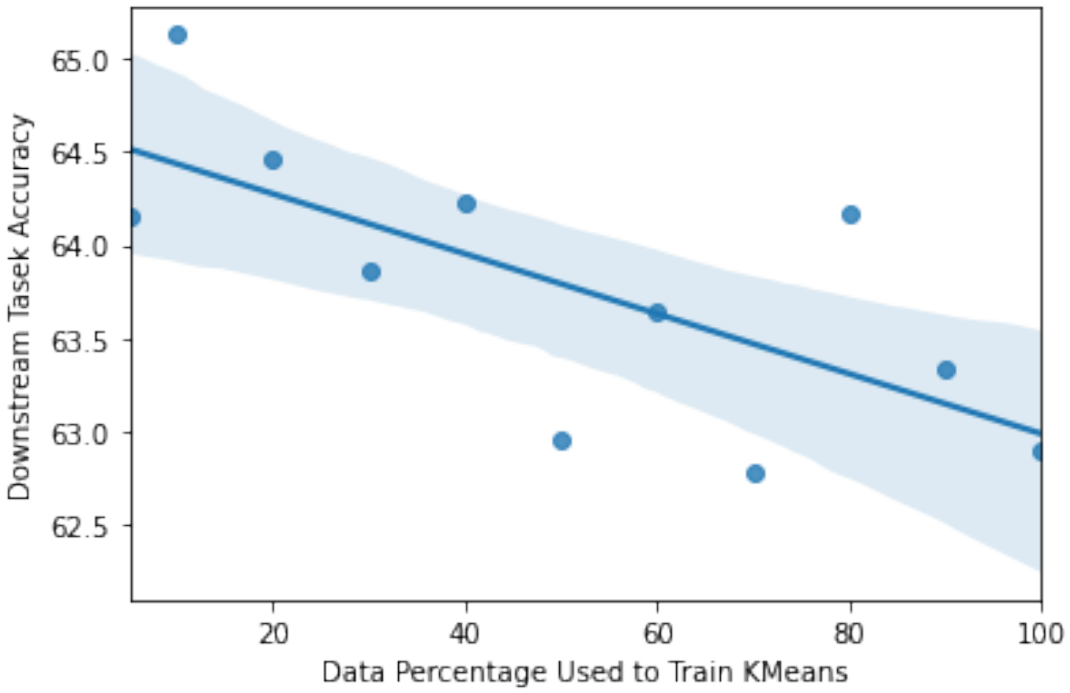


Figure 4.6: Data Percentage for Fitting KMeans vs Deep Cluster Performance

Figure 4.6 show the results. Interestingly the data percentage used to train KMeans is inversely proportional to Deep Cluster Performance. Using 10% of the data achieves the best results with a difference of approx. 2 degrees of accuracy with the least performing value which is 70%.

This finding can also be related to the aforementioned performance variance under large number of clusters due to the random nature of k-means. By fitting k-means over a much more smaller number of samples decreases the randomization effect while preserving the number of clusters benefit.

Chapter 5

Conclusion and Future Work

In this dissertation, we tackled two question in the domain of unsupervised deep learning for visual data.

First we introduced the notion of *inter-model interpretability* to understand how top performing self-supervised deep learning models are similar or dissimilar. We achieved this by projecting the models into a Learned Concepts Embedding (LCE) that reveals their similarities and dissimilarities form learned concepts perspective. The embedding was computed by representing each model by a vector learned concepts through utilizing the recent interpretability approach "Dissect".

We find that the models can be divided into three main groups. And that the embedding axes can be explained by a combination of concepts categories. Crossing the embedding with models performance over a range of different downstream tasks and datasets allowed us to figure out what combination of concepts each task requires. We finally find that two groups constantly complement each other and can be used to formulate better ensembles.

Though Dissect has captured important axes of difference between different models it still a partial representation of the entire concept spectrum learned by the deep learning model. In particular, it focuses on semantic concepts. An interesting future work direction would be to increase Dissect coverage by including more semantic concepts and to include non-semantic concepts such as blur, depth of field, style, etc. A better-balanced validation dataset must be assembled to prevent coincidence matching to frequently occurring objects. On the other side applying Dissect on a larger pool of models with different training procedures, architectures, training data, and training task is interesting. This will enhance the LCE quality and give more insights into the interaction between performance and learned concepts space. It is also of interest to study the effect of different hyperparameters on the placement in the LCE. Hyperparameters such as dropout, normalization, capacity, activation functions, and others.

Second we studied a recent self supervised algorithm called Deep Cluster. Deep Cluster follows the same pipeline of conventional supervised learning ex-

cept for the labels being generated every cycle by applying k-means over the entire dataset after projecting it into the network feature space and using clustering assignments as pseudo-labels. We proposed an explanation about why Deep Cluster works and we supported the proposal by a set of experiments that validates it. The proposal validation revealed Deep Cluster's high sensitivity to random initialization and to the choice of the number of cluster.

We followed the explanation by a series of ablation studies to understand the effect of different parameters such as:

- The importance of continuous clustering which we found to marginally important. We recommended to halt clustering after a couple of early iterations to reduce Deep Cluster computational resources consumption.
- The effect of reducing the number of components used after reducing the feature space through PCA. We found that we can reduce the chosen number of components by 8 folds with negligible effect on performance while benefiting form computational resources reduction.
- The percentage of samples to fit the k-means over. We found that considering much smaller sample than the entire dataset lead to better performance.

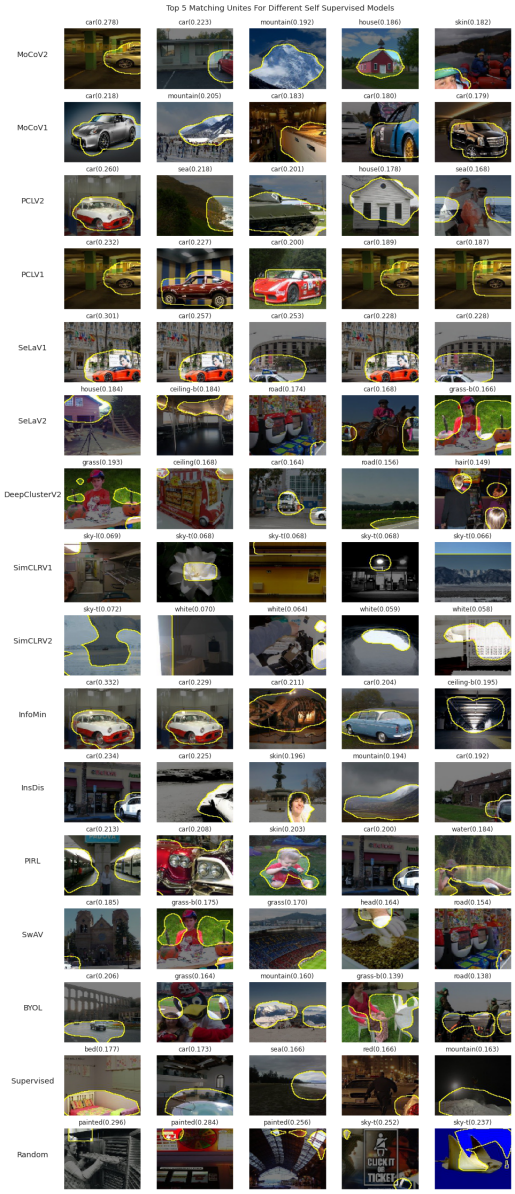
Now that we know that Deep Cluster is sensitive to random initialization, a future work direction would be to investigate possible ways to decrease this effect and to formulate heuristics or procedures to select random initialization that leads to the highest possible accuracy over the downstream task.

Appendix A

Dissect

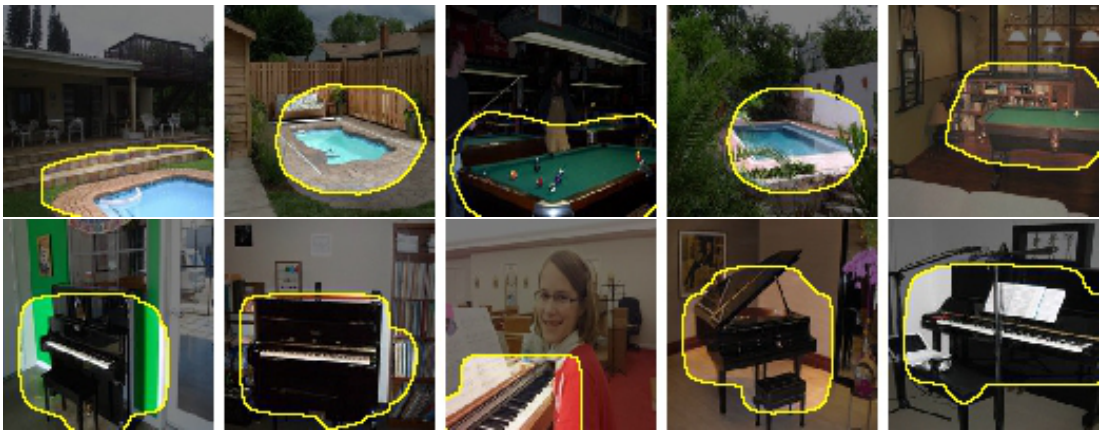
A.0.1 Top Matching Unites

Below we report the top 5 matching units along with the concept they were matched to and the IoU value using dissect for each model. Random and SimCLR models units tend to be matched coincidentally.



A.0.2 Exemplary Activations

The first row of images below corresponds to a unit in the supervised model. The unit is matched to the swimming pool concept. The second row corresponds to a unit in PCLV1 model. The unit is matched to no concept by Dissect.



A.0.3 Principal Components Coefficients

After applying Principal Component Analysis on normalized vectors of Dissect concepts we consider the top three components which represent 87% of data variance. The below figure represents the detailed concept composition of each component. As you can see the first component has the materials fur and skin as major concepts along with other object parts. While the second component has more emphasis on objects. The best performing models tend to minimize both components balancing between different abstractions.

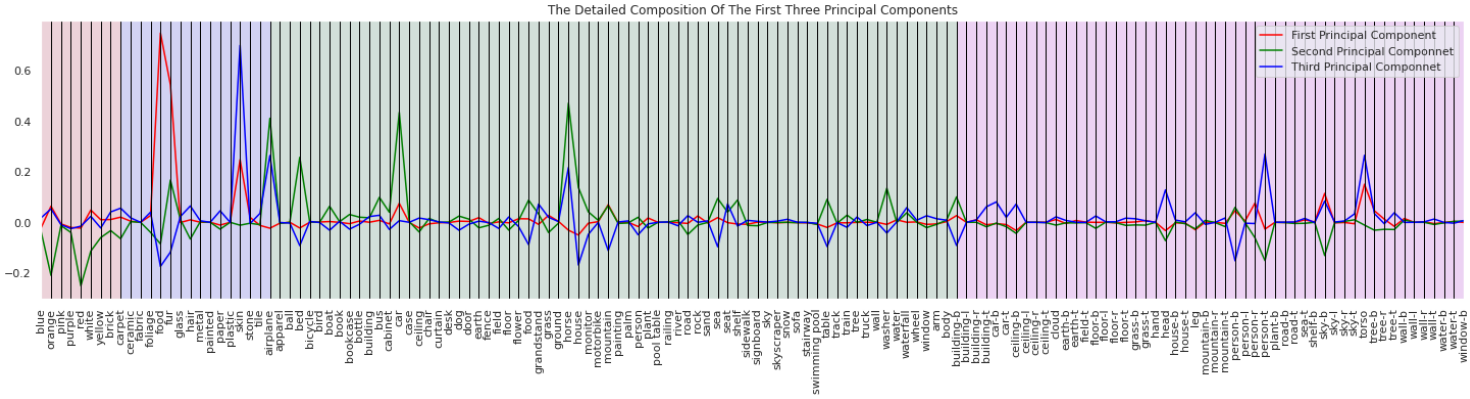


Figure A.1: The Detailed Composition Of Dissect Main Axes Of Variance

Bibliography

[1] L. Jing and Y. Tian, "Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey," *arXiv:1902.06162 [cs]*, Feb. 2019. arXiv: 1902.06162.

[2] L. v. d. Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov. 1998. Conference Name: Proceedings of the IEEE.

[4] G. E. Hinton, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, pp. 504–507, July 2006.

[5] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, pp. 1527–1554, July 2006.

[6] P. Goyal, D. Mahajan, A. Gupta, and I. Misra, "Scaling and Benchmarking Self-Supervised Visual Representation Learning," pp. 6391–6400, 2019.

[7] D. Bau, J.-Y. Zhu, H. Strobelt, A. Lapedriza, B. Zhou, and A. Torralba, "Understanding the Role of Individual Units in a Deep Neural Network," *Proceedings of the National Academy of Sciences*, vol. 117, pp. 30071–30078, Dec. 2020. arXiv: 2009.05041.

[8] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep Clustering for Unsupervised Learning of Visual Features," *arXiv:1807.05520 [cs]*, Mar. 2019. arXiv: 1807.05520.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009. ISSN: 1063-6919.

[10] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning Deep Features for Scene Recognition using Places Database," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[11] G. E. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines," in *Neural Networks: Tricks of the Trade: Second Edition* (G. Montavon, G. B. Orr, and K.-R. Müller, eds.), Lecture Notes in Computer Science, pp. 599–619, Berlin, Heidelberg: Springer, 2012.

[12] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, (Montreal, Quebec, Canada), pp. 1–8, ACM Press, 2009.

[13] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering," *arXiv:1610.04794 [cs]*, June 2017. arXiv: 1610.04794.

[14] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised Deep Embedding for Clustering Analysis," *arXiv:1511.06335 [cs]*, May 2016. arXiv: 1511.06335.

[15] J. Yang, D. Parikh, and D. Batra, "Joint Unsupervised Learning of Deep Representations and Image Clusters," *arXiv:1604.03628 [cs]*, June 2016. arXiv: 1604.03628.

[16] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep Embedding Network for Clustering," in *2014 22nd International Conference on Pattern Recognition*, pp. 1532–1537, Aug. 2014. ISSN: 1051-4651.

[17] G. Chen, "Deep Learning with Nonparametric Clustering," *arXiv:1501.03084 [cs]*, Jan. 2015. arXiv: 1501.03084.

[18] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, "Deep Subspace Clustering Networks," *arXiv:1709.02508 [cs]*, Sept. 2017. arXiv: 1709.02508.

[19] D. Chen, J. Lv, and Y. Zhang, "Unsupervised Multi-Manifold Clustering by Learning Deep Representation," in *AAAI Workshops*, 2017.

[20] S. A. Shah and V. Koltun, "Deep Continuous Clustering," *arXiv:1803.01449 [cs]*, Mar. 2018. arXiv: 1803.01449.

[21] N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders," *arXiv:1611.02648 [cs, stat]*, Jan. 2017. arXiv: 1611.02648.

[22] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering," *arXiv:1611.05148 [cs]*, June 2017. arXiv: 1611.05148.

[23] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets," *arXiv:1606.03657 [cs, stat]*, June 2016. arXiv: 1606.03657.

[24] J. T. Springenberg, "Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks," *arXiv:1511.06390 [cs, stat]*, Apr. 2016. arXiv: 1511.06390.

[25] J. Huang, Q. Dong, S. Gong, and X. Zhu, "Unsupervised Deep Learning by Neighbourhood Discovery," *arXiv:1904.11567 [cs]*, May 2019. arXiv: 1904.11567.

[26] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers, "Clustering with Deep Learning: Taxonomy and New Methods," *arXiv:1801.07648 [cs, stat]*, Sept. 2018. arXiv: 1801.07648.

[27] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture," *IEEE Access*, vol. 6, pp. 39501–39514, 2018. Conference Name: IEEE Access.

[28] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised Visual Representation Learning by Context Prediction," *arXiv:1505.05192 [cs]*, Jan. 2016. arXiv: 1505.05192.

[29] G. Larsson, M. Maire, and G. Shakhnarovich, "Learning Representations for Automatic Colorization," *arXiv:1603.06668 [cs]*, Aug. 2017. arXiv: 1603.06668.

[30] R. Zhang, P. Isola, and A. A. Efros, "Colorful Image Colorization," *arXiv:1603.08511 [cs]*, Oct. 2016. arXiv: 1603.08511.

[31] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context Encoders: Feature Learning by Inpainting," *arXiv:1604.07379 [cs]*, Nov. 2016. arXiv: 1604.07379.

[32] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, "Learning Discrete Representations via Information Maximizing Self-Augmented Training," *arXiv:1702.08720 [cs, stat]*, June 2017. arXiv: 1702.08720.

[33] R. Zhang, P. Isola, and A. A. Efros, "Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Honolulu, HI), pp. 645–654, IEEE, July 2017.

[34] X. Wang, K. He, and A. Gupta, "Transitive Invariance for Self-supervised Visual Representation Learning," *arXiv:1708.02901 [cs]*, Aug. 2017. arXiv: 1708.02901.

[35] M. Noroozi and P. Favaro, "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles," *arXiv:1603.09246 [cs]*, Aug. 2017. arXiv: 1603.09246.

[36] M. Noroozi, H. Pirsiavash, and P. Favaro, "Representation Learning by Learning to Count," *arXiv:1708.06734 [cs]*, Aug. 2017. arXiv: 1708.06734.

[37] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised Representation Learning by Predicting Image Rotations," *arXiv:1803.07728 [cs]*, Mar. 2018. arXiv: 1803.07728.

[38] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-Wise Training of Deep Networks," in *Advances in Neural Information Processing Systems 19* (B. Schölkopf, J. C. Platt, and T. Hoffman, eds.), pp. 153–160, MIT Press, 2007.

[39] Yichuan Tang, R. Salakhutdinov, and G. Hinton, "Robust Boltzmann Machines for recognition and denoising," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, (Providence, RI), pp. 2264–2271, IEEE, June 2012.

[40] S. M. A. Eslami, N. Heess, and J. Winn, "The Shape Boltzmann Machine: a Strong Model of Object Shape," p. 8.

[41] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, Nov. 2016. Google-Books-ID: Np9SDQAAQBAJ.

[42] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," p. 38.

[43] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning - ICML '08*, (Helsinki, Finland), pp. 1096–1103, ACM Press, 2008.

[44] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv:1312.6114 [cs, stat]*, May 2014. arXiv: 1312.6114.

[45] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *arXiv:1406.2661 [cs, stat]*, June 2014. arXiv: 1406.2661.

[46] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *arXiv:1511.06434 [cs]*, Jan. 2016. arXiv: 1511.06434.

[47] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial Feature Learning," *arXiv:1605.09782 [cs, stat]*, Apr. 2017. arXiv: 1605.09782.

[48] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks," *arXiv:1406.6909 [cs]*, June 2015. arXiv: 1406.6909.

[49] P. Bojanowski and A. Joulin, "Unsupervised Learning by Predicting Noise," *arXiv:1704.05310 [cs, stat]*, Apr. 2017. arXiv: 1704.05310.

[50] Z. Wu, Y. Xiong, S. Yu, and D. Lin, "Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination," *arXiv:1805.01978 [cs]*, May 2018. arXiv: 1805.01978.

[51] Y. M. Asano, C. Rupprecht, and A. Vedaldi, "A critical analysis of self-supervision, or what we can learn from a single image," *arXiv:1904.13132 [cs]*, Feb. 2020. arXiv: 1904.13132.

[52] M. Caron, P. Bojanowski, J. Mairal, and A. Joulin, "Unsupervised Pre-Training of Image Features on Non-Curated Data," p. 15.

[53] A. Kolesnikov, X. Zhai, and L. Beyer, "Revisiting Self-Supervised Visual Representation Learning," *arXiv:1901.09005 [cs]*, Jan. 2019. arXiv: 1901.09005.

[54] Y. Bansal, G. Kaplun, and B. Barak, "For self-supervised learning, Rationality implies generalization, provably," *arXiv:2010.08508 [cs, stat]*, Oct. 2020. arXiv: 2010.08508.

[55] D. Erhan, Y. Bengio, A. C. Courville, and P. Vincent, "Visualizing Higher-Layer Features of a Deep Network," 2009.

[56] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," *arXiv:1312.6034 [cs]*, Apr. 2014. arXiv: 1312.6034.

[57] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), Lecture Notes in Computer Science, (Cham), pp. 818–833, Springer International Publishing, 2014.

[58] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization," pp. 618–626, 2017.

[59] S. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," *arXiv:1705.07874 [cs, stat]*, Nov. 2017. arXiv: 1705.07874.

[60] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic Attribution for Deep Networks," in *International Conference on Machine Learning*, pp. 3319–3328, PMLR, July 2017. ISSN: 2640-3498.

[61] Q.-s. Zhang and S.-c. Zhu, "Visual interpretability for deep learning: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, pp. 27–39, Jan. 2018.

[62] C. Olah, A. Mordvintsev, and L. Schubert, "Feature Visualization," *Distill*, vol. 2, p. e7, Nov. 2017.

[63] C. Chen, O. Li, A. Barnett, J. Su, and C. Rudin, "This looks like that: deep learning for interpretable image recognition," in *NeurIPS*, 2019.

[64] J. Bien and R. Tibshirani, "Prototype selection for interpretable classification," *The Annals of Applied Statistics*, vol. 5, Dec. 2011. arXiv: 1202.5933.

[65] L. A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell, "Generating Visual Explanations," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), Lecture Notes in Computer Science, (Cham), pp. 3–19, Springer International Publishing, 2016.

[66] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 664–676, Apr. 2017. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[67] F.-L. Fan, J. Xiong, M. Li, and G. Wang, "On Interpretability of Artificial Neural Networks: A Survey," *IEEE Transactions on Radiation and Plasma Medical Sciences*, pp. 1–1, 2021.

[68] A. Ke, W. Ellsworth, O. Banerjee, A. Y. Ng, and P. Rajpurkar, "CheXtransfer: Performance and Parameter Efficiency of ImageNet Models for Chest X-Ray Interpretation," *Proceedings of the Conference on Health, Inference, and Learning*, pp. 116–124, Apr. 2021. arXiv: 2101.06871.

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015. arXiv: 1512.03385.

[70] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified Perceptual Parsing for Scene Understanding," pp. 418–434, 2018.

[71] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum Contrast for Unsupervised Visual Representation Learning," *arXiv:1911.05722 [cs]*, Mar. 2020. arXiv: 1911.05722.

[72] I. Misra and L. van der Maaten, "Self-Supervised Learning of Pretext-Invariant Representations," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Seattle, WA, USA), pp. 6706–6716, IEEE, June 2020.

[73] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," *arXiv:2002.05709 [cs, stat]*, June 2020. arXiv: 2002.05709.

[74] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, "What Makes for Good Views for Contrastive Learning?," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6827–6839, 2020.

[75] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent: A new approach to self-supervised Learning," *arXiv:2006.07733 [cs, stat]*, Sept. 2020. arXiv: 2006.07733.

[76] J. Li, P. Zhou, C. Xiong, R. Socher, and S. C. H. Hoi, "Prototypical Contrastive Learning of Unsupervised Representations," *arXiv:2005.04966 [cs]*, July 2020. arXiv: 2005.04966.

[77] A. Ym, R. C, and V. A, "Self-labelling via simultaneous clustering and representation learning," Sept. 2019.

[78] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments," *arXiv:2006.09882 [cs]*, Oct. 2020. arXiv: 2006.09882.

[79] L. Ericsson, H. Gouk, and T. M. Hospedales, "How Well Do Self-Supervised Models Transfer?," *arXiv:2011.13377 [cs]*, Nov. 2020. arXiv: 2011.13377.

[80] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-Grained Visual Classification of Aircraft," *arXiv:1306.5151 [cs]*, June 2013. arXiv: 1306.5151.

[81] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *2015 IEEE Conference on*

Computer Vision and Pattern Recognition (CVPR), pp. 3973–3981, June 2015. ISSN: 1063-6919.

[82] M. Nilsback and A. Zisserman, "Automated Flower Classification over a Large Number of Classes," in *2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing*, pp. 722–729, Dec. 2008.

[83] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – Mining Discriminative Components with Random Forests," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), Lecture Notes in Computer Science, (Cham), pp. 446–461, Springer International Publishing, 2014.

[84] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3498–3505, June 2012. ISSN: 1063-6919.

[85] Li Fei-Fei, R. Fergus, and P. Perona, "Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories," in *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pp. 178–178, June 2004.

[86] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," *University of Toronto*, May 2012.

[87] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[88] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing Textures in the Wild," *arXiv:1311.3618 [cs]*, Nov. 2013. arXiv: 1311.3618.

[89] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3485–3492, June 2010. ISSN: 1063-6919.

[90] J. Chen, J. Chen, D. Zhang, Y. Sun, and Y. A. Nanehkaran, "Using deep transfer learning for image-based plant disease identification," *Computers and Electronics in Agriculture*, vol. 173, p. 105393, June 2020.

[91] P. Helber, B. Bischke, A. Dengel, and D. Borth, "EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification," *arXiv:1709.00029 [cs]*, Feb. 2019. arXiv: 1709.00029.

[92] P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Scientific Data*, vol. 5, p. 180161, Aug. 2018. Number: 1 Publisher: Nature Publishing Group.

[93] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3462–3471, July 2017. arXiv: 1705.02315.

[94] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv:1506.01497 [cs]*, Jan. 2016. arXiv: 1506.01497.

[95] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor Semantic Segmentation using depth information," *arXiv:1301.3572 [cs]*, Mar. 2013. arXiv: 1301.3572.

[96] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, May 2017.

[97] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu, "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin," *arXiv:1512.02595 [cs]*, Dec. 2015. arXiv: 1512.02595.

[98] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep Learning for Sensor-based Activity Recognition: A Survey," *Pattern Recognition Letters*, vol. 119, pp. 3–11, Mar. 2019. arXiv: 1707.03502.

[99] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN," *arXiv:1912.04958 [cs, eess, stat]*, Mar. 2020. arXiv: 1912.04958.

[100] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *arXiv:1411.1792 [cs]*, Nov. 2014. arXiv: 1411.1792.

[101] D. Sculley, J. Snoek, A. Wiltschko, and A. Rahimi, "Winner's Curse? On Pace, Progress, and Empirical Rigor," Feb. 2018.

[102] A. M. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng, "On Random Weights and Unsupervised Feature Learning," p. 9.

[103] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv:1708.07747 [cs, stat]*, Sept. 2017. arXiv: 1708.07747.

[104] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.