CLUSTER-BASED WARM-START NETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Theories in cognitive psychology postulate that humans use similarity as a basis for object categorization. However, work in image classification generally assumes disjoint and equally dissimilar classes to achieve super-human levels of performance on certain datasets. In our work, we adapt notions of similarity using weak labels over multiple hierarchical levels to boost classification performance. Instead of pitting clustering directly against classification, we use a warm-start based evaluation to explicitly provide value to a clustering representation by its ability to aid classification. We evaluate on CIFAR10 and a fine-grained classification dataset to show improvements in performance with the procedural addition of intermediate losses and weak labels based on multiple hierarchy levels. Furthermore, we show that pretraining AlexNet on hierarchical weak labels in conjunction with intermediate losses outperforms a classification baseline by over 17% on a subset of Birdsnap dataset. Finally, we show improvement over AlexNet trained using ImageNet pre-trained weights as initializations which further supports our claim of the importance of similarity.

1 INTRODUCTION

Similarity is one of the bases of object categorization in humans (Rosch, 1999). Theories of perceptual categorization in cognitive psychology postulate that humans construct categories by grouping similar stimuli to construct one or several prototypes. New instances are then labeled as the category that they are most similar to. For instance, when one categorizes a specific animal as a dog, they are saying that it is more similar to previously observed dogs than it is to all other objects. This view of categorization better explains the often fuzzy boundaries between real-life classes, where a new object may be equally similar to multiple prototypes. For example, while a dolphin is technically a mammal, its visual appearance is similar to fish, resulting in a lot of people misclassifying it.

While research in cognitive psychology on object categorization might indicate that similarity should play a central role in object classification in humans, image classification in computer vision seems to do pretty well without using it. The task is commonly interpreted as one of classifying an image into one of multiple classes that are assumed to be disjoint and equally dissimilar. The use of softmax-based loss functions assumes that the classes are disjoint, while the use of one-hot labels assumes that classes are equally dissimilar. Despite those strong assumptions, which seem to violate human notions of categories, image classification has shown remarkable progress, achieving super-human performance on multiple datasets (Ioffe & Szegedy, 2015; He et al., 2015). Far from being an anomaly, state-of-the-art models across image classification benchmark datasets use losses, such as cross-entropy loss, that make the explicit assumption of disjoint classes. However, Hinton et al. (2015) notes that the predictions of ensembles of image classification models produces soft labels that "define a rich similarity structure over the data" despite the predictions of individual models not capturing this structure.

Can similarity-based metrics improve classification performance in convolutional neural networks? Previous work has tried to answer this question in different problems such as metric learning (Bellet et al., 2013), clustering (Seldin et al., 2003), and hierarchical classification (Lee & Crawford, 2005). We focus on applications of similarity in the context of convolutional neural networks such as Hsu & Kira (2015) who use a contrastive loss to perform end-to-end clustering using weak labels. While they show impressive performance on MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky & Hinton, 2009), their method does not scale well to more complex datasets. Rippel et al. (2016) propose a new loss function, *magnet loss*, to learn a representational space where distance corresponds

to similarity. They show that clustering in this space allows them to achieve state-of-the-art performance on multiple fine-grained classification datasets.[1] However, they initialize their model with pre-trained weights on ImageNet(Russakovsky et al., 2015), so it is not clear whether their model is learning a good representation, or if it is finding a good mapping between already learned representations and the labels. Wu et al. (2017) pose the problem as one of hierarchical classification and propose a loss based on an ultra-metric tree representation of a hierarchy over the labels. While their loss has interesting properties, it only outperforms classification losses for datasets with a small number of instances per class. In this work, we use a contrastive loss to improve the classification performance of randomly initialized convolutional neural networks on a fine-grained classification task.

What measures of similarity should we teach our model? We represent the relations between our classes in a hierarchy where the labels are all leaf nodes. Therefore, there are two kinds of similarities that we want our model to capture. The first is intra-class similarity—all cats are similar to each other and they are dissimilar to other animals. The second is inter-class similarity—dogs and cats are more similar to each other than they are to non-living objects. For a more complex hierarchy, our model should learn similarity at different levels of the class hierarchy. Wu et al. (2017) use their hierarchical loss function to learn those similarities, however, they observe that reducing the similarity between two classes across all levels of the hierarchy to a single value biases the model towards correct classification at the higher levels of the hierarchy resulting in poor classification performance. Rippel et al. (2016) also observe that applying classification losses to the final layer reduces the entire representation of each class to a single scalar value which destroys intra- and inter-class variation. However, these are the same variations that we want our model to capture. We overcome those limitations by explicitly training the model to capture different grains of similarity at different levels of the network through applying an intermediate pairwise contrastive loss at those levels. In this manner, we can use hierarchical information, such as species of birds having the same coarse-grained category of bird while having different fine-grained labels. Despite being able to encode different levels of similarity, a contrastive loss does not require an explicit hierarchy; we only need weak labels for pairs of instances.

How can we evaluate the representations learned by a clustering algorithm? Previous work has always tried to compete against classification using metrics biased towards the latter task. While some researchers have used hierarchical metrics (Wu et al., 2017) or qualitative measures (Rippel et al., 2016) to evaluate the quality of their representations, their primary evaluation criteria has consistently been the classification accuracy of their model. We propose another way to evaluate the clustering representations by using the clustering model as a "warm start" from which they can train on classification. The intuition is that if the clustering model learned a representations that captures similarity in a space, it would be be at an advantage when it is fine-tuned for classification.

We propose the use of a contrastive loss at different layers of a convolutional neural network to learn a notion of similarity across a set of labels. We also propose the use of the accuracy of a model pre-trained for clustering and fine-tuned for classification as an evaluation metric for clustering algorithms.

2 Intermediate Pairwise Contrastive Loss

Pairwise contrastive losses were first proposed by Hadsell et al. (2006) to learn a function that maps high dimensional inputs to lower dimensional outputs such that distances in the lower dimensional space approximate relationships in the input space. Relationships in the input space are not restricted to simple distance measures. In the context of image classification, the resulting function would have to learn the complex invariances that make two images of the same class similar, such that they are both mapped to points that are close to each other in the output space. This loss can be calculated using the representation of two points in the same embedding space and a binary label for whether or not they are similar. Hence, the loss only requires weak labels in the form of pairwise relationships.

[1] It should be noted that in work following that paper, Krause et al. (2016) achieve a better performance using softmax, which can be seen as their baseline classification models.

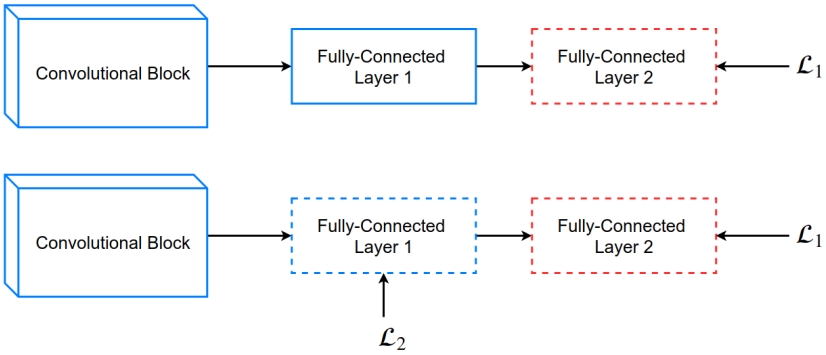


Figure 1: Pairwise losses are applied to intermediate layers of the network as they are calculated through pairwise comparisons of the activations at those layers.

2.1 Loss Formulation

Let (X_1,X_2) be vectors that are calculated as mappings of two input instances in the same embedding space. $\mathbb{1}_{X_1,X_2}$ is an indicator variable that takes a value of one if (X_1, X_2) are similar and zero otherwise. $D(X, Y)$ can be any distance function between two vectors in the same space. $L_S(\cdot)$ is a partial loss function for a similar pair, and $L_D(\cdot)$ is a partial loss function for dissimilar pairs. Then the general formulation of the pairwise contrastive loss function is as shown in Eq. (1):

$$\mathcal{L}(\mathbf{X_1}, \mathbf{X_2}) = \mathbb{1}_{X_1,X_2}\, L_S(D(\mathbf{X_1}, \mathbf{X_2})) + (1 - \mathbb{1}_{X_1,X_2}) L_D(D(\mathbf{X_1}, \mathbf{X_2})). \tag{1}$$

The first half of Eq. (1) penalizes any divergence between the two vectors for similar instances while the second half of the loss rewards divergence up to a margin. Following the formulation proposed by (Hsu & Kira, 2015), we use Kullback-Leibler (KL) divergence as our distance metric and a hinge loss for dissimilar cases. The final formulation of the loss we use is shown in Eq. (2):

$$\mathcal{L}(\mathbf{P}, \mathbf{Q}) = loss(\mathbf{P}, \mathbf{Q}) + loss(\mathbf{Q}, \mathbf{P}), \tag{2}$$

$$loss(\mathbf{P}, \mathbf{Q}) = \mathbb{1}_{P,Q}\, KL(\mathbf{P}||\mathbf{Q}) + (1 - \mathbb{1}_{P,Q}) \max\{margin - KL(\mathbf{P}||\mathbf{Q}), 0\}, \tag{3}$$

$$KL(\mathbf{P} \parallel \mathbf{Q}) = \sum_{i=1}^{k} P_i \log\left(\frac{P_i}{Q_i}\right). \tag{4}$$

We have found that a value of two for the $margin$ works well. Since we apply the loss to activations at intermediate layers of the network, we first apply a softmax operation on the activations to obtain the two vectors, P and Q, used in Eq. (2).

2.2 Hierarchical Similarity

Given that one of our objectives is to encode different levels of similarity into the representations learned by the network, we explore different ways of using the intermediate loss to achieve that goal as shown in Fig. 2. We establish a baseline network where the losses are all applied to the output of the last layer. We then use three different variants by applying intermediate losses at different levels of the network. In variant B, we apply the intermediate loss at the penultimate layer. This allows us to apply the contrastive-loss to a higher dimensional space where it could potentially learn a richer representation. In variants C and D, we provide the network with two different kinds of weak labels based on fine-grained and coarse-grained. Since we expect that a weaker form of similarity would be more beneficial earlier on in the network pipeline, we apply the coarse-grained loss to the second-to-last fully-connected layer in Network C. It should be noted that many instance pairs will have contradicting losses since they may match at the coarse-grained level but conflict at the fine-grained

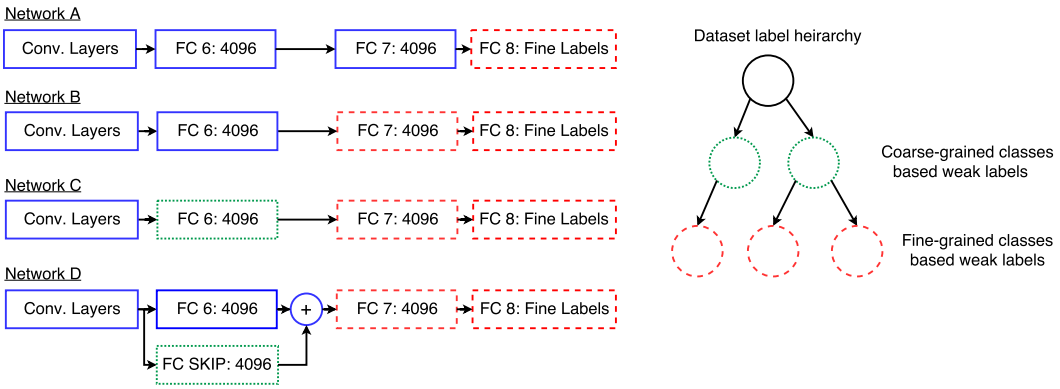


Figure 2: Different network architecture and losses. Network A is the baseline network. Network B applies the fine-grained intermediate loss to the penultimate layer as well. Network C extends Network B by applying the coarse-grained intermediate loss at the second-to-last layer. Network D extends Network B by adding a branch around the second-to-last layer, and applying the coarse-grained intermediate loss there.

level. To tackle this, we augment the network with a skip layer around the second-to-last layer and apply the coarse-grained loss to the skip layer only. Hence, the coarse-grained loss will only provide additional information to the network without directly detracting from the representations passed to subsequent layers. While we show the network variants on AlexNet, the same ideas are applied to a variant of LeNet that has three fully-connected layers.

We follow the efficient implementation regime proposed by (Hsu & Kira, 2015) in order to train a single network based on pairwise constraints. The original network formulation to handle pairwise constraints is a Siamese network. For every pairwise constraint, at least two data passes need to be performed which results in redundancies. Instead, we feed the softmax outcomes from a single network and the pairwise similarity constraints to the contrastive loss function which externally handles all possible pairwise constraints that are available within a mini-batch. We further extend the application of contrastive loss functions to multiple intermediate layers with a minimal overhead of extra processing and pairwise constraints, if alternate hierarchical labels are used. Details regarding the exact parameters and setup used to perform experiments are furnished in the Appendices. PyTorch[2] is used for all implementations discussed within this paper.

3 Experiments

We evaluate our approach on two different datasets using two different convolutional neural networks: LeNet on CIFAR10 and AlexNet on Birdsnap34–a 34-class subset of the Birdsnap dataset (Berg et al., 2014). We generate coarse-grained labels for CIFAR10 by separating the ten classes into six animal classes and four vehicle classes. Below is a discussion of how we constructed Birdsnap34.

Birdsnap34 Birdsnap is a popular fine-grained classification dataset that has 500 different classes and over 40,000 images. We annotated Birdsnap using the scientific classification of each bird to create a six-level hierarchy over all the labels. In this work, we use the birds' subfamily and family as our fine-grained and coarse-grained labels, respectively. Using a pairwise loss with a large number of classes poses a significant implementation challenge. For a uniform dataset with n classes, the ratio of similar pairs to dissimilar pairs is roughly $1 : (n-1)$. Hence, for a large number of classes, most of the pairs will be dissimilar resulting in a very weak signal to the model for clustering instances together. This challenge is analogous to that of imbalanced datasets (Van Horn & Perona, 2017), which is a major problem that lies beyond the scope of our paper. Instead of abandoning Birdsnap, we curate a 34-class subset, which we will refer to as Birdsnap34, and apply our approach to it. The classes are chosen such that each class comes from a unique biological subfamily of birds. Through

[2]http://pytorch.org/

sampling form unique subfamilies, we increase the interclass variance between our classes. The resulting dataset has 2982 training images and 158 test images that are divided into 34 fine-grained and 18 coarse-grained classes. We plan on extending our model to the entire Birdsnap dataset in future work.

All the network architectures used have randomly initialized weights unless specified otherwise. The models are evaluated using accuracy, purity, and Normalized Mutual Information (NMI) (Strehl & Ghosh, 2002). We train each model for a fixed number of epochs, and we report the top performing results out of 5 runs. Details regarding the exact parameters and setups used to perform the experiments are available in Appendix B.

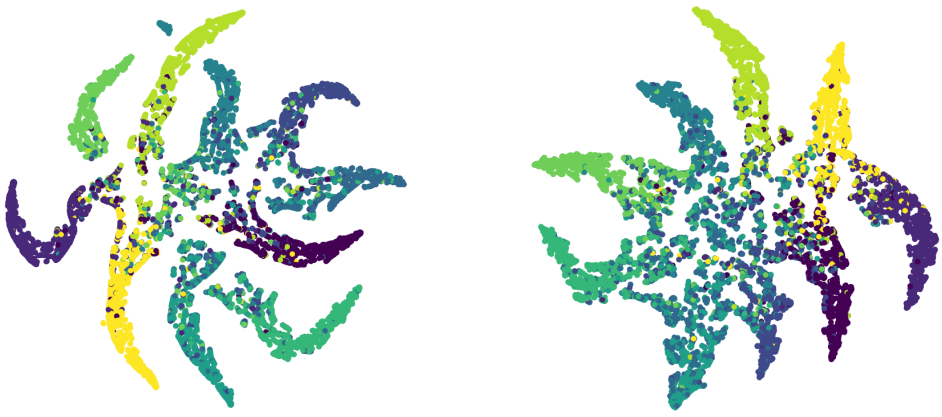
3.1 Quality of Warm Start as an Evaluation Metric

In our first experiment, we show the value of evaluating the representations learned by a clustering loss through using it as a warm start from training on classification. We replicate the setup used by Hsu & Kira (2015) by using LeNet on CIFAR10, and extend their framework to AlexNet on Birdsnap34. We use the Hungarian algorithm (Kuhn, 1955) to evaluate clustering accuracy. This method of evaluation assumes that the clusters will be centered across the different dimensions of the softmax output of the final layer. The Hungarian algorithm is then used to find the assignment of dimensions to classes that would achieve the highest accuracy. We use the Hungarian algorithm to evaluate clustering performance in all of our experiments.

Table 1: Baseline classification performance of LeNet and AlexNet

	CIFAR-10	Birdsnap-34
Classification	75.48	**53.50**
Clustering	74.21	17.20
Warm Start Classification	**75.75**	50.32

From Table 1 we observe that for CIFAR10, the application of the loss to the final layer results in clustering performance almost matching classification performance. However, in more complex dataset such as Birdsnap34, this pattern disappears. This supports the pattern shown in (Hsu & Kira, 2015) where the gap between clustering and classification performance increases when they move from MNIST to CIFAR10. The Cross-Entropy loss formulation, as mentioned in (Hsu & Kira, 2015), is extremely harsh and seeks to segment each and every sample into one absolute class. The tSNE (van der Maaten & Hinton, 2008) outcomes from the last fully connected layer of LeNet in Fig. 3 match our observations.



(a) Classification (b) Clustering

Figure 3: t-SNE Visualization of the outputs of LeNet on CIFAR10

3.2 INTERMEDIATE LOSS

The previous experiment indicates that the representations learned from end-to-end clustering do not extend beyond simple datasets. We posit that this is a result of how the clustering loss is applied. Most machine learning models only apply losses to their last layer. This is generally a reasonable thing to do since we only have labels for the output of the last layer. On the other hand, pairwise losses can be applied at any level of the network since they only require a pair of vectors in the same embedding space. In this experiment we explore the potential of applying losses at intermediate layers on the learned representations.

Table 2: Using intermediate losses

		Clustering Acc.	NMI	Purity	Classification Acc.
CIFAR10	Cross-Entropy	–	–	–	75.48
	LeNet A	**74.21**	**0.55**	**0.74**	**75.75**
	LeNet B	72.66	0.54	0.73	75.42
Birdsnap34	Cross-Entropy	–	–	–	**53.50**
	AlexNet A	17.20	0.39	0.17	50.32
	AlexNet B	**20.38**	**0.45**	**0.20**	45.86

We train two variants of LeNet and AlexNet using the pairwise contrastive loss. Variant A follows the setup used by Hsu & Kira (2015), while Variant B applies the same loss at both the final and penultimate layer of the convolutional neural network. We also train the same network using Cross-Entropy loss to establish baseline performance. As shown in Table 2, performance on CIFAR10 is almost the same across all three settings. Meanwhile, we see clear differences on Birdsnap34. There is a significant increase across all metrics for the clustering models for the intermediate loss version, however, this improvement does not carry over to the classification accuracy.

3.3 HIERARCHICAL CLUSTERING

Previous work on embedding spaces has shown that adding structure to the embedding space or learning it using a hierarchical structure often validates its integrity as well as improves performance (Mikolov et al., 2013). In an effort to understand the impact of adding a hierarchical-label-based contrastive loss to our baseline networks, we apply it on variants C and D of LeNet and AlexNet. Since the restriction on dimensionality is removed by using a contrastive loss with KL divergence, we can directly apply the same loss to an alternate set of weak labels and representations. Our ultimate expectation is that adding hierarchical structure should aid in improving classification, given their potency to improve embedding spaces as shown in previous works.

Table 3: Impact of adding heirarchy based weak labels

		Clustering Acc.	NMI	Purity	Classification Acc.
CIFAR10	Cross-Entropy	–	–	–	75.48
	LeNet A	**74.21**	0.55	0.74	**75.75**
	LeNet B	72.66	0.54	0.73	75.42
	LeNet C	72.84	0.55	0.73	75.11
	LeNet D	74.17	**0.56**	**0.74**	75.53
Birdsnap34	Cross-Entropy	–	–	–	53.50
	AlexNet A	17.20	0.39	0.17	50.32
	AlexNet B	20.38	0.45	0.20	45.86
	AlexNet C	**24.84**	**0.51**	**0.25**	**70.70**
	AlexNet D	22.29	0.49	0.22	65.61

Table 3 shows that the experimental results align with our expectations in terms of overall improvement in classification. As seen in other experiments, the performance on CIFAR10 is almost the same across all metrics. Meanwhile, there is a spike in performance for Birdsnap34 for both AlexNet C and D with the metrics far exceeding classification performance. Given that Birdsnap34

has a more well-defined hierarchy, it would be expected that using the hierarchical labels to learn similarity would result in learning good representations. Another thing worth noting is that the dataset is curated by having subfamilies as fine-grained labels and biological families as coarse-grained labels. This structure results in similarity along this tree corresponding to distinct visual features. This might be indicating that choice of hierarchy can have an impact on performance. A surprising finding is that AlexNet C outperforms AlexNet D. One would expect AlexNet D to be able to extract features for both the coarse and fine levels of the hierarchy. We plan on investigating this in future work.

3.4 Sensitivity to Network Size and Weight Initialization

Given the ability of standard end-to-end clustering networks to perform fine-tuned classification reasonably well, we test their sensitivity towards network size in order to understand if they would be able to achieve or outperform their simpler baseline classification counterparts. In order to do so, we double the number of convolutional filters and the sizes of fully-connected layers within AlexNet. We use their doubled networks to perform clustering and fine-tune them for classification. We denote these modified networks with a suffix of "Double."

Table 4: Sensitivity to network capacity

	Clustering Acc.	NMI	Purity	Classification Acc.
Cross-Entropy	–	–	–	53.50
AlexNet A (Double)	22.93	0.49	0.23	59.24
AlexNet C	**24.84**	**0.51**	**0.25**	**70.70**

Table 5: Comparison against ImageNet pre-trained weights

	Clustering Acc.	NMI	Purity	Classification Acc.
Cross-Entropy	–	–	–	81.53
AlexNet C	24.20	0.51	0.24	**82.80**

From Table 4, we clearly observe that clustering is highly sensitive to network capacity. Compared to AlexNet A, its doubled counterpart allows clustering to learn better representations which leads to classificaiton performance being higher than the baseline classification model. However, our AlexNet C variant outperforms the double capacity network. This points to the efficacy of the combination of weak labels over multiple levels of heirarchy and the use of intermediate losses. Furthermore, Table 5 shows that even we use ImageNet pre-trained weights, variant C outperforms the ImageNet pre-trained classification baseline. This points to a tractable and viable approach to extending performance over more complex datasets without the need to increase the capacity of neural networks.

4 Discussion and Future Work

In this work, we argue that similarity-based losses can be applied as a warm start to boost the performance of image classification models. We show that applying a pairwise contrastive loss to intermediate layers of the network results in better clustering performance, as well as better fine-tuned classification performance. Furthermore, we demonstrate how the pairwise loss can be used to train a model using weak hierarchical labels.

We find that training with hierarchical labels results in the highest performance beating models with more parameters and approaches the performance of models pre-trained with ImageNet weights. This is a very significant finding since ImageNet contains multiple bird classes, so a model pre-trained with ImageNet has seen many more birds than a model pre-trained with pairwise constraints on Birdsnap34. Nevertheless, it only outperforms a cluster-based warm-start model by just 10%. Regardless, we also show that applying our approach to ImageNet weights still results in a boost

of 1.27%. This supports our claim that similarity-based metrics can improve classification performance, but it suggests that the gains we expect decrease if we start of with a good representational space.

We hope to expand this in future work in multiple ways. We plan on extending our approach to the entire Birdsnap dataset, as well as to other fine-grained classification datasets. We also hope to perform more extensive analysis of the quality of the embedding spaces learned as it is obvious that the use of the Hungarian algorithm does not accurately reflect the performance of the clustering model.

REFERENCES

Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L Alexander, David W Jacobs, and Peter N Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2011–2018, 2014.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pp. 1735–1742. IEEE, 2006.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Yen-Chang Hsu and Zsolt Kira. Neural network-based clustering using pairwise constraints. *arXiv preprint arXiv:1511.06321*, 2015.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *European Conference on Computer Vision*, pp. 301–320. Springer, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Sanghoon Lee and Melba M Crawford. Unsupervised multistage image classification using hierarchical clustering with a bayesian similarity measure. *IEEE Transactions on Image Processing*, 14 (3):312–320, 2005.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *hlt-Naacl*, volume 13, pp. 746–751, 2013.

Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. Metric learning with adaptive density discrimination. *International Conference on Learning Representations*, 2016.

Eleanor Rosch. Principles of categorization. *Concepts: core readings*, 189, 1999.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Yevgeny Seldin, Sonia Starik, and Michael Werman. Unsupervised clustering of images using their joint segmentation. In *Proceedings of the 3rd International Workshop on Statistical and Computational Theories of Vision (SCTV 2003)*, 2003.

Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

Grant Van Horn and Pietro Perona. The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*, 2017.

Cinna Wu, Mark Tygert, and Yann LeCun. Hierarchical loss for classification. *arXiv preprint arXiv:1709.01062*, 2017.

A PREPROCESSING STEPS

We use two primary datasets, CIFAR10 and Birdsnap34, in experiments that substantiate our claims of improving on the classification task by means of using clustering representations as a warm-start. For the CIFAR10 dataset, we follow the preprocessing steps used by (Hsu & Kira, 2015). They are listed below,

- Convert the images to YUV format.
- Calculate the mean and standard deviation of U and V channels over the entire training set.
- For each image, normalize by its Y channel mean and standard deviation while normalizing over the remaining channels using the aggregate mean and standard deviation over the entire training set.

NOTE: The mean and standard deviation for U and V channel as per our calculations are (0.001, 0.003) and (0.227, 0.105).

For the Birdsnap34 dataset, we perform the standard normalization using mean and standard deviations for R, G and B channels as (0.485, 0.229), (0.456, 0.224) and (0.406, 0.225).

B EXPERIMENTAL SETUP

Table 6: Parameters used for Classification in Network A

	Parameters	Values
CIFAR10	Total Epochs	95
	Batch Size	16
	Learning Rate	0.01
	Learning Rate Scale Factor	0.5
	Learning Rate Schedule	25, 50, 75, 100
	Momentum	0.9
	Optimizer	SGD
	$margin$	2.0
	Weight Decay	0.0005
	Weights of Losses	N/A
Birdsnap34	Total Epochs	195
	Batch Size	16
	Learning Rate	0.01
	Learning Rate Scale Factor	0.5
	Learning Rate Schedule	50, 100, 150, 200
	Momentum	0.9
	Optimizer	SGD
	$margin$	2.0
	Weight Decay	0.001
	Weights of Losses	N/A

Table 7: Parameters used for Clustering in Network A

	Parameters	Values
CIFAR10	Total Epochs	95
	Batch Size	16
	Learning Rate	0.01
	Learning Rate Scale Factor	0.5
	Learning Rate Schedule	25, 50, 75, 100
	Momentum	0.9
	Optimizer	SGD
	$margin$	2.0
	Weight Decay	0.0005
	Weights of Losses	N/A
Birdsnap34	Total Epochs	195
	Batch Size	1024
	Learning Rate	0.01
	Learning Rate Scale Factor	0.5
	Learning Rate Schedule	50, 100, 150, 200
	Momentum	0.9
	Optimizer	SGD
	$margin$	2.0
	Weight Decay	0.001
	Weights of Losses	N/A

Table 8: Parameters used for Clustering in Network B

	Parameters	Values
CIFAR10	Total Epochs	95
	Batch Size	16
	Learning Rate	0.01
	Learning Rate Scale Factor	0.5
	Learning Rate Schedule	25, 50, 75, 100
	Momentum	0.9
	Optimizer	SGD
	$margin$	2.0
	Weight Decay	0.0005
	Weights of Losses	0.2, 0.8
Birdsnap34	Total Epochs	195
	Batch Size	128
	Learning Rate	0.01
	Learning Rate Scale Factor	0.5
	Learning Rate Schedule	50, 100, 150, 200
	Momentum	0.9
	Optimizer	SGD
	$margin$	2.0
	Weight Decay	0.0005
	Weights of Losses	0.2, 0.8

Table 9: Parameters used for Clustering in Network C

	Parameters	Values
CIFAR10	Total Epochs	95
	Batch Size	16
	Learning Rate	0.01
	Learning Rate Scale Factor	0.5
	Learning Rate Schedule	25, 50, 75, 100
	Momentum	0.9
	Optimizer	SGD
	$margin$	2.0
	Weight Decay	0.0005
	Weights of Losses	0.4, 0.2, 0.4
Birdsnap34	Total Epochs	195
	Batch Size	128
	Learning Rate	0.01
	Learning Rate Scale Factor	0.5
	Learning Rate Schedule	50, 100, 150, 200
	Momentum	0.9
	Optimizer	SGD
	$margin$	2.0
	Weight Decay	0.0005
	Weights of Losses	0.4, 0.2, 0.4

Table 10: Parameters used for Clustering in Network D

	Parameters	Values
CIFAR10	Total Epochs	95
	Batch Size	16
	Learning Rate	0.01
	Learning Rate Scale Factor	0.5
	Learning Rate Schedule	25, 50, 75, 100
	Momentum	0.9
	Optimizer	SGD
	$margin$	2.0
	Weight Decay	0.0005
	Weights of Losses	0.4, 0.2, 0.4
Birdsnap34	Total Epochs	195
	Batch Size	128
	Learning Rate	0.01
	Learning Rate Scale Factor	0.5
	Learning Rate Schedule	50, 100, 150, 200
	Momentum	0.9
	Optimizer	SGD
	$margin$	2.0
	Weight Decay	0.0005
	Weights of Losses	0.4, 0.2, 0.4