Why Do Better Loss Functions Lead to Less Transferable Features?

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep learning researchers have proposed many new loss functions and regularizers that improve test accuracy on image classification tasks. However, it is not clear whether these loss functions learn better representations for downstream tasks. This paper studies how the choice of training objective affects the transferability of the hidden representations of convolutional neural networks trained on ImageNet. We show that many objectives lead to statistically significant improvements in ImageNet accuracy over vanilla softmax cross-entropy, but the resulting fixed feature extractors transfer substantially worse to downstream tasks, and the choice of loss has little effect when networks are fully fine-tuned on the new tasks. Using centered kernel alignment to measure similarity between hidden representations of networks, we find that differences among loss functions are apparent only in the last few layers of the network. We delve deeper into representations of the penultimate layer, finding that different objectives and hyperparameter combinations lead to dramatically different levels of class separation. Representations with higher class separation obtain higher accuracy on the original task, but their features are less useful for downstream tasks. Our results suggest there exists a trade-off between learning invariant features for the original task and features relevant to transfer tasks.

1 Introduction

Features learned by deep neural networks on ImageNet transfer effectively to a wide range of computer vision tasks [21, 54, 30]. These networks are often pretrained using vanilla softmax cross-entropy [9, 10], but recent work reports that other loss functions such as label smoothing [60] and sigmoid cross-entropy [7] outperform softmax cross-entropy on ImageNet. Although a previous investigation suggested that label smoothing and dropout can hurt transfer learning [30], the vast majority of work studying alternatives to softmax cross-entropy has restricted its empirical investigation to the accuracy of the trained networks on the original dataset. While it is valuable to study the impact of objective functions on classification accuracy, often improving transfer accuracy of learned representations has a more significant practical utility.

This paper takes a comparative approach to understand the effects of different training objectives for image classification through the lens of neural network representations and their transferability. We carefully tune hyperparameters of each loss function and confirm that several loss functions outperform softmax cross-entropy by a statistically significant margin on ImageNet. However, we find that these improvements do not transfer to other tasks (see Table 1 and Figure 1). We delve deeper and explain these empirical findings in terms of effects of different objectives on neural network representations. Our key findings are as follows:

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

- We analyze the performance of 9 different objectives on ImageNet and in transfer settings. Although many loss functions and regularizers lead to statistically significant improvements over vanilla softmax cross-entropy on ImageNet, these gains do not transfer. These alternative loss functions produce fixed feature extractors that transfer substantially worse to other tasks, in terms of both linear and k-nearest neighbors classification accuracy, and provide no benefit when representations are fully fine-tuned.
- The choice of objective primarily affects representations in network layers close to the output. Centered kernel alignment (CKA) reveals large differences in representations of the last few layers of the network, whereas earlier layers are similar regardless of which training objective is used. This helps explain why the choice of objective has little impact on fine-tuning transfer accuracy.
- All objectives that improve accuracy over softmax cross-entropy also lead to greater separation between representations of different classes in the penultimate layer features. Thus, these alternative objectives appear to collapse within-class variability in representations, which accounts for both the improvement in accuracy on the original task and the reduction in the quality of the features on downstream tasks.

2 Loss Functions and Output Layer Regularizers

We investigate 9 loss functions and output layer regularizers. Let $\boldsymbol{z} \in \mathbb{R}^K$ denote the network's output ("logit") vector, and let $\boldsymbol{t} \in \{0, 1\}^K$ denote a one-hot vector of targets, where $\|\boldsymbol{t}\|_1 = 1$. Let $\boldsymbol{x} \in \mathbb{R}^M$ denote the vector of penultimate layer activations, which gives rise to the output vector as $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, where $\boldsymbol{W} \in \mathbb{R}^{K \times M}$ is the matrix of final layer weights, and $\boldsymbol{b}$ is a vector of biases.

All investigated loss functions include a term that encourages $\boldsymbol{z}$ to have a high dot product with $\boldsymbol{t}$. To avoid solutions that make this dot product large simply by increasing the scale of $\boldsymbol{z}$, these loss functions must also include one or more contractive terms and/or normalize $\boldsymbol{z}$. Many "regularizers" correspond to additional contractive terms added to the loss, so we do not draw a firm distinction between loss functions and regularizers. We describe each objective in detail below. Hyperparameters are provided in Appendix A.1.

Softmax cross-entropy [9, 10] is the de facto loss function for multi-class classification in deep learning:

$$\mathcal{L}_{\text{softmax}}(\boldsymbol{z}, \boldsymbol{t}) = -\sum_{k=1}^{K} t_k \log\left(\frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}\right) = -\sum_{k=1}^{K} t_k z_k + \log \sum_{k=1}^{K} e^{z_k}. \tag{1}$$

The loss consists of a term that maximizes the dot product between the logits and targets, as well as a contractive term that minimizes the $\mathrm{LogSumExp}$ of the logits.

Label smoothing [60] "smooths" the targets for softmax cross-entropy. The new targets are given by mixing the original targets with a uniform distribution over all labels, $t' = t \times (1 - \alpha) + \alpha/K$, where α determines the weighting of the original and uniform targets. In order to maintain the same scale for the gradient with respect to the positive logit, in our experiments, we scale the label smoothing loss by $1/(1 - \alpha)$. The resulting loss is:

$$\mathcal{L}_{\text{smooth}}(\boldsymbol{z}, \boldsymbol{t}; \alpha) = -\frac{1}{1-\alpha} \sum_{k=1}^{K} \left((1-\alpha)t_k + \frac{\alpha}{K}\right) \log\left(\frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}\right) \tag{2}$$

Müller et al. [38] previously showed that label smoothing improves calibration and encourages class centroids to lie at the vertices of a regular simplex.

Dropout on penultimate layer: Dropout [58] is among the most prominent regularizers in the deep learning literature. We consider dropout applied to the penultimate layer of the neural network, i.e., when inputs to the final layer are randomly kept with some probability ρ. When employing dropout, we replace the penultimate layer activations $\boldsymbol{x}$ with $\tilde{\boldsymbol{x}} = \boldsymbol{x} \odot \boldsymbol{\xi}/\rho$ where $\xi_i \sim \mathrm{Bernoulli}(\rho)$. Writing the dropped out logits as $\tilde{\boldsymbol{z}} = \boldsymbol{W}\tilde{\boldsymbol{x}} + \boldsymbol{b}$, the dropout loss is:

$$\mathcal{L}_{\text{dropout}}(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}, \boldsymbol{t}; p) = \mathbb{E}_{\boldsymbol{\xi}}\left[\mathcal{L}_{\text{softmax}}(\tilde{\boldsymbol{z}}, \boldsymbol{t})\right] = -\sum_{k=1}^{K} t_k z_k + \mathbb{E}_{\boldsymbol{\xi}}\left[\log \sum_{k=1}^{K} e^{\tilde{z}_k}\right]. \tag{3}$$

Dropout produces both *implicit* regularization, by introducing noise into the optimization process, and *explicit* regularization, by changing the parameters that minimize the loss [62, 68].

Extra final layer L^2 regularization: It is common to place the same L^2 regularization on the final layer as elsewhere in the network. However, we find that applying greater L^2 regularization to the final layer can improve performance. In architectures with batch normalization, adding additional L^2 regularization has no explicit regularizing effect if the learnable scale (γ) parameters that are unregularized, but it still exerts an implicit regularizing effect by altering optimization.

Logit penalty: Whereas label smoothing encourages logits not to be too negative, and dropout imposes a penalty on the logits that depends on the covariance of the weights, an alternative possibility is simply to explicitly constrain logits to be small in L^2 norm:

$$\mathcal{L}_{\text{logit_penalty}}(\boldsymbol{z}, \boldsymbol{t}; \beta) = \mathcal{L}_{\text{softmax}}(\boldsymbol{z}, \boldsymbol{t}) + \beta \|\boldsymbol{z}\|^2. \tag{4}$$

Dauphin and Cubuk [18] showed that this regularizer yields comparable accuracy improvements to dropout.

Logit normalization: We consider the use of L^2 *normalization*, rather than regularization, of the logits. Because the entropy of the output of the softmax function depends on the scale of the logits, which is lost after normalization, we introduce an additional temperature parameter τ that controls the magnitude of the logit vector, and thus, indirectly, the minimum entropy of the output distribution:

$$\mathcal{L}_{\text{logit_norm}}(\boldsymbol{z}, \boldsymbol{t}; \tau) = \mathcal{L}_{\text{softmax}}(\boldsymbol{z}/(\tau\|\boldsymbol{z}\|), \boldsymbol{t}) = -\frac{1}{\tau\|\boldsymbol{z}\|} \sum_{k=1}^{K} t_k z_k + \log \sum_{k=1}^{K} e^{z_k/(\tau\|\boldsymbol{z}\|)}. \tag{5}$$

Cosine softmax: We additionally consider L^2 normalization of both the penultimate layer features and the final layer weights corresponding to each class. This loss is equivalent to softmax cross-entropy loss if the logits are given by cosine similarity $\mathrm{sim}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\intercal \boldsymbol{y}/(\|\boldsymbol{x}\|\|\boldsymbol{y}\|)$ between the weight vector and the penultimate layer plus a per-class bias:

$$\mathcal{L}_{\cos}(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}, \boldsymbol{t}; \tau) = -\sum_{k=1}^{K} t_k \left(\mathrm{sim}(\boldsymbol{W}_{k,:}, \boldsymbol{x})/\tau + b_k\right) + \log \sum_{k=1}^{K} e^{\mathrm{sim}(\boldsymbol{W}_{k,:}, \boldsymbol{x})/\tau + b_k}, \tag{6}$$

where τ is a temperature parameter as above. Similar losses have appeared in previous literature [49, 64, 69, 65, 66, 20, 33, 14], and variants have introduced explicit additive or multiplicative margins to this loss that we do not consider here [33, 65, 66, 20]. We observe that, even without an explicit margin, manipulating the temperature has a large impact on observed class separation.

Sigmoid cross-entropy, also known as binary cross-entropy, is the natural analog to softmax cross-entropy for multi-label classification problems. Although we investigate only single-label multi-class classification tasks, we train networks with sigmoid cross-entropy and evaluate accuracy by ranking the logits of the sigmoids. This approach is related to the one-versus-rest strategy for converting binary classifiers to multi-class classifiers. The sigmoid cross-entropy loss is:

$$\begin{aligned}\mathcal{L}_{\text{sce}}(\boldsymbol{z}, \boldsymbol{t}) &= -\textstyle\sum_{k=1}^{K} \left(t_k \log\left(\frac{e^{z_k}}{e^{z_k}+1}\right) + (1 - t_k) \log\left(1 - \frac{e^{z_k}}{e^{z_k}+1}\right) \right) \\ &= -\textstyle\sum_{k=1}^{K} t_k z_k + \sum_{k=1}^{K} \log(e^{z_k} + 1).\end{aligned} \tag{7}$$

The $\mathrm{LogSumExp}$ term of softmax loss is replaced with the sum of the $\mathrm{softplus}$-transformed logits. We initialize the biases of the logits $\boldsymbol{b}$ to $-\log(K)$ so that the initial output probabilities are approximately $1/K$. Beyer et al. [7] have previously shown that sigmoid cross-entropy loss leads to improved accuracy on ImageNet relative to softmax cross-entropy.

Squared error: Finally, we investigate squared error loss, as formulated by Hui and Belkin [28]:

$$\mathcal{L}_{\text{mse}}(\boldsymbol{z}, \boldsymbol{t}; \kappa, M) = \frac{1}{K} \sum_{k=1}^{K} \left(\kappa t_k (z_k - M)^2 + (1 - t_k) z_k^2 \right), \tag{8}$$

where κ and M are hyperparameters. κ sets the strength of the loss for the correct class relative to incorrect classes, whereas M controls the magnitude of the correct class target. When $\kappa = M = 1$, the loss is simply the mean squared error between $\boldsymbol{z}$ and $\boldsymbol{t}$. Like Hui and Belkin [28], we find that placing greater weight on the correct class slightly improves ImageNet accuracy.

3 Results

For each loss, we trained 8 ResNet-50 [27, 25] models on the ImageNet ILSVRC 2012 dataset [19, 51]. To tune loss hyperparameters and the epoch for early stopping, we performed 3 training runs per hyperparameter configuration where we held out a validation set of 50,046 ImageNet training example. We provide further details regarding training in Appendix A.1.

3.1 Better objectives improve accuracy, but do not transfer better

Table 1: **Objectives that produce higher ImageNet accuracy lead to less transferable fixed features.** "ImageNet" columns reflect accuracy of each model on the ImageNet validation set. "Transfer" columns reflect accuracy of L^2-regularized multinomial logistic regression or k-nearest neighbors classifiers trained to classify different transfer datasets using the fixed penultimate layer features of the ImageNet-trained networks. Numbers are averaged over 8 different pretraining initializations, and all values not significantly different than the best are bold-faced ($p < 0.05$, t-test). The strength of L^2 regularization is selected on a validation set that is independent of the test set, and k is selected using leave-one-out cross-validation on the training set. See Appendix A.2 for training details.

	ImageNet		Transfer							
Loss	Top-1	Top-5	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Pets	Flowers
Linear transfer:										
Softmax	77.0	93.40	**74.6**	**92.4**	**76.9**	**55.4**	**62.0**	**60.3**	92.0	**94.0**
Label smoothing	77.6	93.78	72.7	91.6	75.2	53.6	61.6	54.8	**92.9**	91.9
Sigmoid	**77.9**	93.50	73.4	91.7	75.7	52.3	**62.0**	56.1	92.5	92.9
More final layer L^2	77.7	93.79	70.6	91.0	73.7	51.5	60.1	50.3	92.4	89.8
Dropout	77.5	93.62	72.6	91.4	75.0	53.6	61.2	54.7	**92.6**	92.1
Logit penalty	77.7	**93.83**	68.1	90.2	72.3	48.1	59.0	48.3	92.3	86.6
Logit normalization	**77.8**	93.71	66.3	90.5	72.9	50.7	58.1	45.4	92.0	82.9
Cosine softmax	**77.9**	**93.86**	62.0	89.9	71.3	45.4	55.0	36.7	91.1	75.3
Squared error	77.2	92.79	39.8	82.2	56.3	21.8	39.9	15.3	84.7	46.7
K-nearest neighbors:										
Softmax	77.0	93.40	**60.9**	**88.8**	**67.4**	**38.4**	53.0	**28.9**	88.8	**83.6**
Label smoothing	77.6	93.78	59.2	88.3	66.3	**39.2**	**53.5**	27.3	**91.5**	80.3
Sigmoid	**77.9**	93.50	58.5	88.2	66.6	34.4	52.8	26.7	90.8	81.5
More final layer L^2	77.7	93.79	55.0	87.7	65.0	36.2	51.1	24.4	90.9	75.6
Dropout	77.5	93.62	58.4	88.2	66.4	38.1	52.4	27.8	91.1	80.1
Logit penalty	77.7	**93.83**	52.5	86.5	62.9	31.0	49.3	22.4	90.8	68.1
Logit normalization	**77.8**	93.71	50.9	86.6	63.1	35.1	45.8	24.1	88.2	63.1
Cosine softmax	**77.9**	**93.86**	45.5	86.0	61.5	28.8	41.8	19.0	87.0	52.7
Squared error	77.2	92.79	27.7	74.5	44.3	13.8	28.6	9.1	82.6	28.2

We found that, when properly tuned, all investigated objectives except squared error provide a statistically significant improvement over softmax cross-entropy, as shown in the left two columns of Table 1. Although the gains are small, they are potentially meaningful, with sigmoid cross-entropy and cosine softmax both leading to an improvement of 0.9% in top-1 accuracy over the baseline. For further discussion of the characteristics of models trained with different losses, see Appendix B.

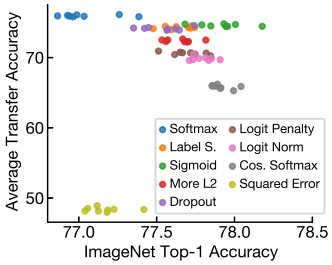

Figure 1: **Higher ImageNet accuracy is not associated with higher linear transfer accuracy.** Points represent the accuracies of individual training runs.

Although networks trained with softmax cross-entropy attain lower ImageNet top-1 accuracy than any other loss function, they nonetheless provide the most transferable features. We evaluated the transferability of the fixed features of our ImageNet-pretrained models by training linear or k-nearest neighbors (kNN) classifiers to classify eight different datasets of natural images: Food-101 [8], CIFAR-10 and CIFAR-100 [32], Birdsnap [5], SUN397 [70], Stanford Cars [31], Oxford-IIIT Pets [46], and Oxford Flowers [43]. The results of these experiments are shown in Table 1 and summarized in Figure 1. In both linear and kNN settings, vanilla softmax cross-entropy is the best-performing loss function on the majority of tasks. The one task where other

Table 2: **The training objective has little impact on the performance of fine-tuned networks.** Accuracy of fine-tuning pretrained networks on transfer datasets, averaged over 3 different pretraining initializations. We tune hyperparameters separately for each objective and dataset. Numbers not significantly different than the best are bold-faced ($p < 0.05$, t-test for individual datasets, two-way ANOVA for average). See Appendix A.3 for training details.

Pretraining loss	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Pets	Flowers	Avg.
Softmax	**88.2**	**96.9**	**84.1**	**76.2**	63.4	91.3	93.1	**96.7**	**86.2**
Label smoothing	**88.3**	**96.7**	84.0	**76.3**	63.6	91.2	**93.7**	**96.3**	**86.3**
Sigmoid	**88.3**	**96.9**	83.6	**76.2**	63.5	**91.8**	**93.7**	**96.3**	**86.3**
More final layer L^2	88.1	**96.9**	**84.4**	75.9	**64.5**	**91.5**	**93.8**	**96.2**	**86.4**
Dropout	**88.3**	**96.7**	**84.2**	**76.5**	63.9	91.2	**93.9**	**96.3**	**86.4**
Logit penalty	**88.4**	**96.9**	83.9	**76.4**	63.3	91.2	**93.4**	**96.0**	86.2
Logit normalization	87.9	**96.9**	82.9	**76.0**	58.3	91.2	92.7	95.9	85.2
Cosine softmax	**88.3**	**96.9**	83.2	**75.6**	56.9	91.3	92.5	95.9	85.1
Squared error	87.8	**96.9**	**84.0**	75.7	61.0	91.4	93.0	95.2	85.6

objectives consistently perform better than vanilla softmax cross-entropy is Oxford Pets, which has significant class-level overlap with ImageNet.

As shown in Table 2, when networks are fully fine-tuned on downstream tasks, the pretraining objective has little effect on the resulting accuracy. When averaging across all tasks, the best loss provides only a 0.2% improvement over softmax cross-entropy, which does not reach statistical significance and is much smaller than the 0.9% difference in ImageNet top-1 accuracy. Thus, using a different loss function for pretraining can improve accuracy on the pretraining task, but this improvement does not appear to transfer to downstream tasks.

3.2 The choice of objective primarily affects hidden representations close to the output

Our observation that "improved" objectives improve only on ImageNet and not for transfer tasks raises questions about which representations, exactly, they affect. We use two tools to investigate differences in the hidden representations of networks trained with different loss functions. First, we use centered kernel alignment [29, 16, 17] to directly measure the similarity of hidden representations across networks trained with different loss functions. Second, we measure the sparsity of the ReLU activations in each layer. Both analyses suggest that all loss functions investigated lead to similar representations throughout the majority of the network, and differences are apparently only in the last few ResNet blocks.

Linear centered kernel alignment (CKA) provides a way to measure similarity of neural network representations that is invariant to rotation and isotropic scaling in representation space [29, 16, 17]. Unlike other ways of measuring representational similarity between neural networks, linear CKA can identify architectural correspondences between layers of networks trained from different initializations [29], a prerequisite for comparing networks trained with different objectives. Given two matrices $\boldsymbol{X} \in \mathbb{R}^{n \times p_1}$ and $\boldsymbol{Y} \in \mathbb{R}^{n \times p_2}$ containing activations to the same n examples, linear CKA computes the cosine similarity between the reshaped $n \times n$ covariance matrices between examples:

$$\mathrm{CKA}_{\text{linear}}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{\mathrm{vec}(\mathrm{cov}(\boldsymbol{X})) \cdot \mathrm{vec}(\mathrm{cov}(\boldsymbol{Y}))}{\|\mathrm{cov}(\boldsymbol{X})\|_{\mathrm{F}} \|\mathrm{cov}(\boldsymbol{Y})\|_{\mathrm{F}}}. \tag{9}$$

We measured CKA between all possible pairings of ResNet blocks from 18 networks (2 different initializations for each objective). To reduce memory requirements, we used minibatch CKA [42] with minibatches of size 1500 and processed the ImageNet validation set for 10 epochs.

As shown in Figure 2, representations of early network layers are highly similar regardless of loss function, whereas representations of later network layers differ substantially. These results indicate that different loss functions lead networks to learn similar features throughout most of the network, and it is only the features in the last few ResNet blocks that are clearly different.

The sparsity of activations reveals a similar pattern of layer-wise differences between networks trained with different loss functions. Figure 3 shows the proportion of non-zero activations in different layers. In all networks, the percentage of non-zero ReLU activations decreases with depth, attaining its minimum at the last convolutional layer. In the first three ResNet stages, activation sparsity is broadly similar regardless of the loss. However, in the final stage and penultimate average pooling layer, the

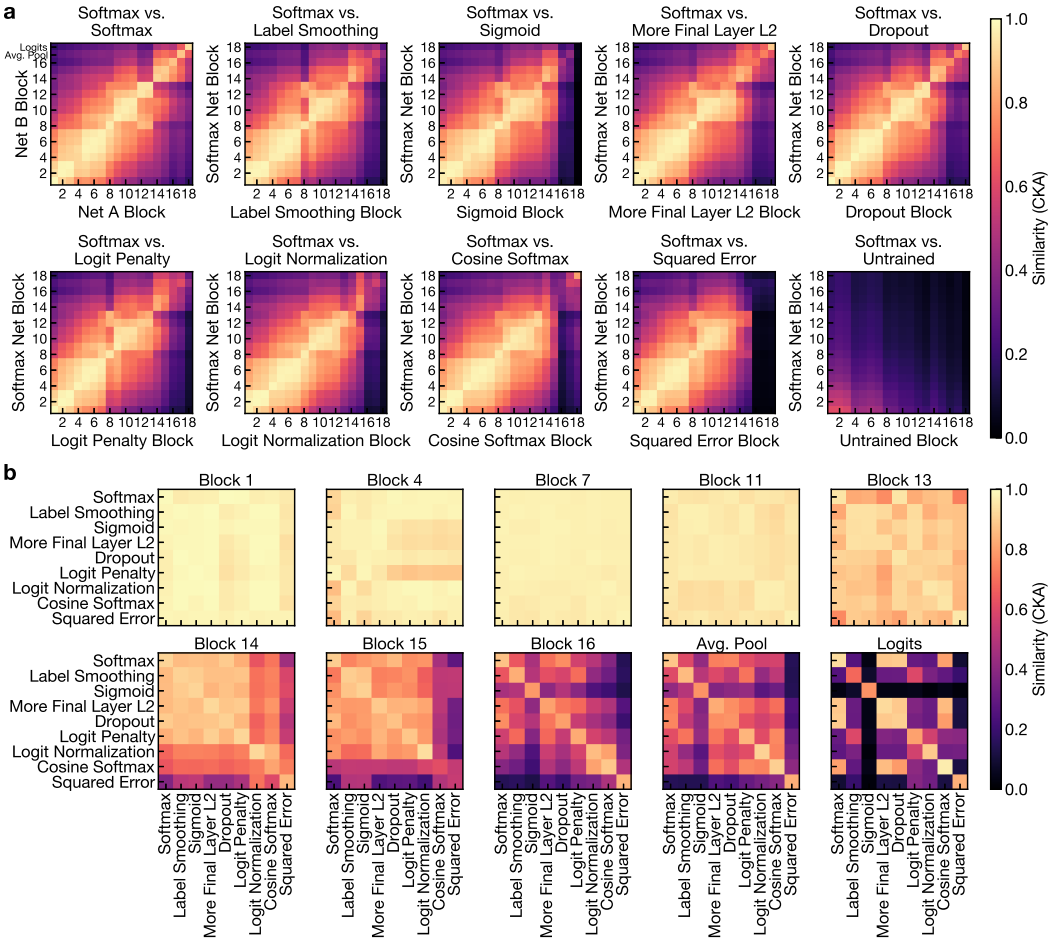

Figure 2: **The loss function has little impact on representations in early network layers.** All plots show linear centered kernel alignment (CKA) between representations computed on the ImageNet validation set. **a**: CKA between network layers, for pairs of ResNet-50 models trained from different initializations with the same or different losses. As a control, the bottom-right plot shows CKA between a model trained with softmax loss and a model at initialization where batch norm moments have been computed on the training set. **b**: CKA between representations extracted from corresponding layers of networks trained with different loss functions. Diagonal reflects similarity of networks with the same loss function trained from different initalizations.

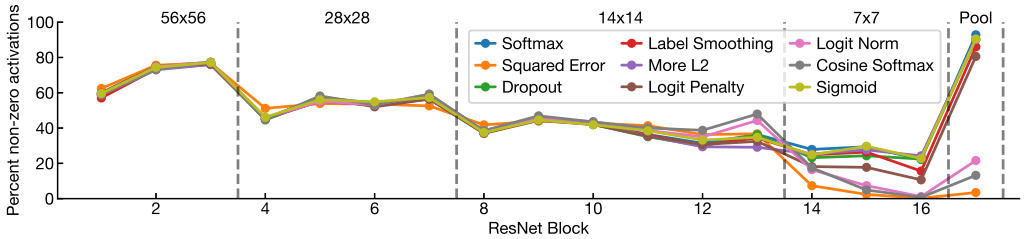

Figure 3: **Loss functions affect sparsity of later layer representations.** Plot shows the average % non-zero activations for each ResNet-50 block, after the residual connection and subsequent nonlinearity, on the ImageNet validation set. Dashed lines indicate boundaries between stages.

degree of sparsity depends greatly on the loss. Penultimate layer representations of vanilla softmax networks are the least sparse, with 92.8% non-zero activations. Logit normalization, cosine softmax, and squared error all result in much sparser (<25% non-zero) activations.

These results immediately suggest an explanation for the limited effect of the training objective on networks' fine-tuning performance. We observe differences among objectives only in later network layers, but previous work has found that these layers change substantially during fine-

Table 3: **Regularization and alternative losses improve class separation in the penultimate layer.** Results averaged over 8 ResNet-50 models per loss on the ImageNet training set.

Loss/regularizer	ImageNet top-1	Class separation (R^2)
Softmax	77.0	0.345
Label smoothing	77.6	0.420
Sigmoid	77.9	0.427
Extra L^2	77.7	0.572
Dropout	77.5	0.461
Logit penalty	77.7	0.601
Logit norm	77.8	0.517
Cosine softmax	77.9	0.641
Squared error	77.2	0.845

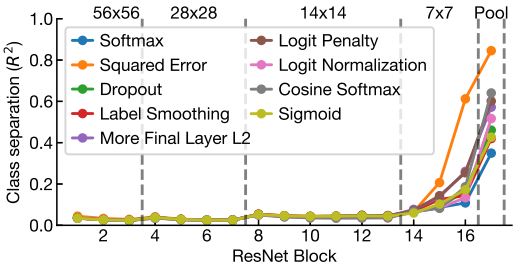

Figure 4: Class separation in different layers of ResNet-50, averaged over 8 models per loss on the ImageNet training set. For convolutional layers, we compute cosine distances by flattening the representations of examples across spatial dimensions.

tuning [72, 48, 41], an observation we replicate in Appendix C. Thus, the choice of training objective appears to affect parts of the network that are specific to the pretraining task, and do not transfer when the network is fine-tuned on other tasks.

3.3 Regularization and alternative losses increase class separation

Is there a property of the investigated regularizers that can potentially explain their beneficial effect on accuracy as well as their harmful effect on fixed features? As shown in Appendix D, combining different losses and penultimate layer regularizers provides no accuracy improvements over training with only one, suggesting that all operate via a similar mechanism. In this section, we demonstrate that, compared to vanilla softmax cross-entropy, all investigated losses cause the network to shrink within-class variance in the representations.

The ratio of the average within-class cosine distance to the overall average cosine distance measures the dispersion of representations of examples belonging to the same class relative to the overall dispersion of embeddings. We take one minus this quantity to get a closed-form index of class separation that is between 0 and 1:

$$R^2 = 1 - \bar{d}_{\text{within}}/\bar{d}_{\text{total}} \tag{10}$$

$$\bar{d}_{\text{within}} = \sum_{k=1}^{K} \sum_{m=1}^{N_k} \sum_{n=1}^{N_k} \frac{1 - \text{sim}(\boldsymbol{x}_{k,m}, \boldsymbol{x}_{k,n})}{K N_k^2}, \quad \bar{d}_{\text{total}} = \sum_{j=1}^{K} \sum_{k=1}^{K} \sum_{m=1}^{N_j} \sum_{n=1}^{N_k} \frac{1 - \text{sim}(\boldsymbol{x}_{j,m}, \boldsymbol{x}_{k,n})}{K^2 N_j N_k}$$

where $\boldsymbol{x}_{k,m}$ is the embedding of example m in class $k \in \{1, \ldots, K\}$, N_k is the number of examples in class k, and $\text{sim}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}/(\|\boldsymbol{x}\|\|\boldsymbol{y}\|)$ is cosine similarity between vectors. As we show in Appendix E, if the embeddings are first L^2 normalized, then $1 - R^2$ is the ratio of the average within-class variance to the weighted total variance, where the weights are inversely proportional to the number of examples in each class. For a balanced dataset, R^2 is equivalent to centered kernel alignment [16, 17] between the embeddings and the one-hot label matrix, with a cosine kernel. Appendix H shows results with other distance metrics.

As shown in Table 3 and Figure 4, all regularizers and alternative loss functions result in greater class separation in penultimate (average pooling) layer representations as compared to vanilla softmax loss. We also found that the training dynamics of class separation differed between softmax cross-entropy and other objectives. Appendix F shows that, for softmax cross-entropy, class separation peaks early in training and then falls, whereas for other objectives, class separation either saturates or continues to rise as training progresses.

3.4 Greater class separation is associated with less transferable features

Although losses that improve class separation lead to higher accuracy on the ImageNet validation set, the feature extractors they learn transfer worse to other tasks. Figure 5a plots mean linear transfer accuracy versus class separation for each of the losses we investigate. We observe a significant negative correlation (Spearman's $\rho = -0.93$, $p = 0.002$). Notably, vanilla softmax cross-entropy

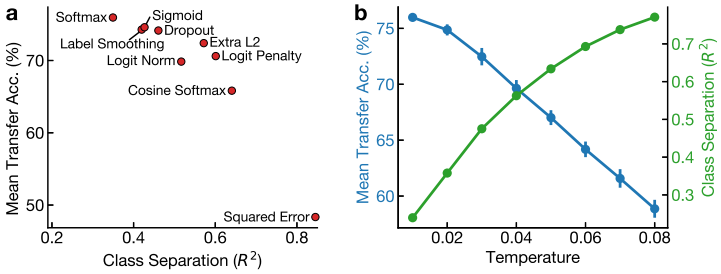

Figure 5: **Class separation negatively correlates with linear transfer accuracy. a**: Mean linear transfer accuracy across all tasks vs. class separation on ImageNet, for different loss functions. **b**: For cosine softmax loss, training at higher temperature produces greater class separation but worse linear transfer accuracy.

Table 4: **Temperature of cosine softmax loss controls ImageNet accuracy, class separation (R^2), and linear transfer accuracy.**

	ImageNet		Transfer							
Temp.	Top-1	R^2	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Pets	Flowers
0.01	74.9	0.236	**73.4**	**91.9**	**76.5**	**57.2**	**60.5**	**62.9**	91.7	**93.6**
0.02	77.0	0.358	72.1	91.8	76.2	56.5	60.4	58.5	92.2	91.2
0.03	77.5	0.475	69.1	91.5	74.9	53.7	59.1	51.8	**92.3**	87.4
0.04	**77.6**	0.562	66.0	90.7	73.8	50.3	57.4	45.1	91.7	82.2
0.05	**77.6**	0.634	62.8	90.4	72.2	47.6	55.4	38.6	91.0	78.3
0.06	77.5	0.693	60.3	89.3	69.8	43.3	53.8	33.3	91.0	72.7
0.07	77.5	0.738	57.1	88.7	68.6	39.6	51.4	29.1	90.2	67.9
0.08	**77.6**	**0.770**	53.7	87.7	66.5	35.5	49.4	25.7	89.3	63.2

produces the least class separation and the most transferable features, whereas squared error results in much greater class separation than other losses and leads to much lower transfer performance.

To confirm the relationship between class separation, ImageNet accuracy, and transfer, we trained models with cosine softmax with varying values of the temperature parameter τ.[1] As shown in Table 4, lower temperatures yield lower top-1 accuracies and worse class separation. However, even though the lowest temperature of $\tau = 0.01$ achieved 2.7% lower accuracy on ImageNet than the best temperature, this lowest temperature gave the best features for nearly all transfer datasets. Thus, τ controls a tradeoff between the generalizability of penultimate-layer features and the accuracy on the target dataset. For $\tau \geq 0.04$, ImageNet accuracy saturates, but, as shown in Figure 5b, class separation continues to increase and transfer accuracy continues to decrease.

Is there any situation where features with greater class separation could be beneficial for a downstream task? In Appendix G, we use L^2-regularized logistic regression to relearn the original 1000-way ImageNet classifier from 40,000 examples, and find that features from networks trained with vanilla softmax loss perform *worst*, whereas features from networks with greater class separation perform substantially better. Thus, representations with greater class separation are "overfit," not to the pretraining datapoints, but to the pretraining *classes*—they perform better for classifying these classes, but worse when the downstream task requires classifying different classes.

4 Related work

Understanding training objectives. There is a great deal of previous work that investigates why some objectives perform better than others, using both theoretical and empirical approaches. Theoretical analysis of loss functions is challenging; in most cases, solutions cannot be expressed in closed form even when the predictor is linear. However, Soudry et al. [57] have previously shown that, on linearly separable data, gradient descent on the unregularized logistic or multinomial logistic regression objectives (i.e., linear models with sigmoid or softmax cross-entropy loss) eventually converges to the minimum norm solution. These results can be extended to neural networks in certain restricted settings [57, 26, 67]. Theoretical analysis of dropout has bounded the excess risk of single-layer [63] and two-layer [37] networks. Empirical studies have investigated regularization in more realistic

[1] Training at $\tau < 0.05$ was unstable; we scale the loss by the τ to reduce the effect of τ on the size of the gradient WRT the correct class logit. Relationships for $\tau \geq 0.05$ remain consistent without loss scaling.

settings. Label smoothing has been explained in terms of mitigation of label noise [34, 12], entropy regularization [47, 36], and accelerated convergence [71].

Training objectives and transferability. Our study of the transferability of networks trained with different objectives extends the previous investigation of Kornblith et al. [30], which showed that two of the regularizers considered here (label smoothing and dropout) lead to features that transfer worse. Similar results have been reported for self-supervised learning, where the loss parameters that maximize accuracy on the contrastive task do not provide the best features for linear classification [13]. We elaborate on these previous observations of dissociations between accuracy and transferability, demonstrating that it is a common feature of many loss functions and penultimate layer regularization techniques, and connecting it to class separation. Other related work has attempted to devise loss functions to learn more transferable embeddings for few-shot learning [56, 59, 6, 14, 44], but Tian et al. [61] show that embeddings of models trained with softmax cross-entropy often perform on par with these more sophisticated techniques. Interestingly, Goldblum et al. [23] find that regularization to *increase* class separation improves few-shot classification performance on mini-ImageNet and CIFAR-FS. We hypothesize that increased class separation helps on these datasets because superclasses in the training set are also present in the test set [50], whereas the transfer datasets we examine are substantially different from ImageNet.

Class separation. Prior work has investigated class separation in neural networks in a variety of different ways. Theoretical work connects various concepts of the normalized margin between classes to the generalization properties of neural networks [3, 39, 4, 40]. Empirically, Chen et al. [11] measure class separation via "angular visual hardness," the arccosine-transformed cosine similarity between the weight vectors and examples, and suggested an association between this metric and accuracy. However, as shown in Appendix Figure H.1, we find that this metric is unable to differentiate between networks trained with softmax and sigmoid cross-entropy. Papyan et al. [45] describe a phenomenon they call "neural collapse," which occurs after training error vanishes and is associated with the collapse of the class means, classifier, and activations to the verticies of an equiangular tight frame, implying maximal class separation. We relate our observations to theirs in Appendix Table H.2. Other work has investigated how class information evolves through the hidden layers of neural networks using linear classifiers [1], binning estimators of mutual information [55, 52, 24], Euclidean distances [53], the soft nearest neighbor loss [22], and manifold geometry [2, 15]. In the context of a study of representational consistency across networks trained from different initializations, Mehrer et al. [35] previously reported increased class separation in CIFAR-10 networks trained with dropout.

5 Limitations

Our results establish that a wide variety of losses that improve over vanilla softmax cross-entropy lead to greater class separation, but this observation does not lead to a recipe for higher pretraining accuracy. The relationship between class separation and pretraining accuracy is non-monotonic; squared error achieves the highest class separation but does not significantly outperform vanilla softmax cross-entropy. In addition, although ResNet-50 on ImageNet is a realistic transfer scenario where our analysis is computationally tractable, bigger models trained on bigger datasets with more classes achieve better performance, and their richer label spaces could potentially help to mitigate the trade-off between pretraining accuracy and feature quality.

6 Conclusion

In this study, we establish that the properties of a loss that yields good performance on the pretraining task are different from the properties of a loss that works well for transfer. Training objectives that lead to better performance on the pretraining task learn more invariant representations with greater class separation. However, these same properties are detrimental when fixed features are transferred to other tasks. Moreover, because different loss functions produce different representations only in later network layers, which change substantially during fine-tuning, gains in pretraining accuracy do not lead to gains when models are fine-tuned on downstream tasks.

Our work suggests opportunities for improving fixed feature representations in deep neural networks. We see no inherent reason that features learned by softmax cross-entropy should be optimal for transfer, but previous work has optimized for test set accuracy, rather than transferable features. With the increasing importance of transfer learning in deep learning, we believe that future research into loss functions should explicitly target and evaluate performance under transfer settings.

References

[1] G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[2] A. Ansuini, A. Laio, J. H. Macke, and D. Zoccolan. Intrinsic dimension of data representations in deep neural networks. In *Advances in Neural Information Processing Systems*, 2019.

[3] P. L. Bartlett. For valid generalization, the size of the weights is more important than the size of the network. *Advances in Neural Information Processing Systems*, pages 134–140, 1997.

[4] P. L. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. *Advances in Neural Information Processing Systems*, 2017:6241–6250, 2017.

[5] T. Berg, J. Liu, S. W. Lee, M. L. Alexander, D. W. Jacobs, and P. N. Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2019–2026. IEEE, 2014.

[6] L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

[7] L. Beyer, O. J. Hénaff, A. Kolesnikov, X. Zhai, and A. v. d. Oord. Are we done with ImageNet? *arXiv preprint arXiv:2006.07159*, 2020.

[8] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101 — mining discriminative components with random forests. In *European Conference on Computer Vision*, pages 446–461. Springer, 2014.

[9] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990.

[10] J. S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Advances in Neural Information Processing Systems*, pages 211–217, 1990.

[11] B. Chen, W. L. A. Garg, Z. Yu, A. Shrivastava, J. Kautz, and A. Anandkumar. Angular visual hardness. In *International Conference on Machine Learning*, 2020.

[12] B. Chen, L. Ziyin, Z. Wang, and P. P. Liang. An investigation of how label smoothing affects generalization. *arXiv preprint arXiv:2010.12648*, 2020.

[13] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

[14] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

[15] U. Cohen, S. Chung, D. D. Lee, and H. Sompolinsky. Separability and geometry of object manifolds in deep neural networks. *Nature Communications*, 11(1):1–13, 2020.

[16] C. Cortes, M. Mohri, and A. Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13(1):795–828, 2012.

[17] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems*, pages 367–373, 2002.

[18] Y. Dauphin and E. D. Cubuk. Deconstructing the regularization of batchnorm. In *International Conference on Learning Representations*, 2021.

[19] 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*, pages 248–255. Ieee, 2009.

[20] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.

[21] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, 2014.

[22] N. Frosst, N. Papernot, and G. Hinton. Analyzing and improving representations with the soft nearest neighbor loss. In *International Conference on Machine Learning*, pages 2012–2020. PMLR, 2019.

[23] M. Goldblum, S. Reich, L. Fowl, R. Ni, V. Cherepanova, and T. Goldstein. Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *International Conference on Machine Learning*, pages 3607–3616. PMLR, 2020.

[24] Z. Goldfeld, E. v. d. Berg, K. Greenewald, I. Melnyk, N. Nguyen, B. Kingsbury, and Y. Polyanskiy. Estimating information flow in deep neural networks. In *International Conference on Machine Learning*, 2018.

[25] S. Gross and M. Wilber. Training and investigating residual nets. In *The Torch Blog*. `http://torch.ch/blog/2016/02/04/resnets.html`, 2016.

[26] S. Gunasekar, J. D. Lee, D. Soudry, and N. Srebro. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems*, pages 9461–9471, 2018.

[27] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[28] L. Hui and M. Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. In *International Conference on Learning Representations*, 2021.

[29] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, 2019.

[30] S. Kornblith, J. Shlens, and Q. V. Le. Do better ImageNet models transfer better? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019.

[31] J. Krause, J. Deng, M. Stark, and L. Fei-Fei. Collecting a large-scale dataset of fine-grained cars. In *Second Workshop on Fine-Grained Visual Categorization*, 2013.

[32] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[33] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 212–220, 2017.

[34] M. Lukasik, S. Bhojanapalli, A. Menon, and S. Kumar. Does label smoothing mitigate label noise? In *International Conference on Machine Learning*, pages 6448–6458. PMLR, 2020.

[35] J. Mehrer, C. J. Spoerer, N. Kriegeskorte, and T. C. Kietzmann. Individual differences among deep neural network models. *Nature Communications*, 11(1):1–12, 2020.

[36] C. Meister, E. Salesky, and R. Cotterell. Generalized entropy regularization or: There's nothing special about label smoothing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6870–6886, 2020.

[37] P. Mianjy and R. Arora. On convergence and generalization of dropout training. *Advances in Neural Information Processing Systems*, 33, 2020.

[38] R. Müller, S. Kornblith, and G. E. Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pages 4696–4705, 2019.

[39] B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401. PMLR, 2015.

[40] B. Neyshabur, S. Bhojanapalli, and N. Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018.

[41] B. Neyshabur, H. Sedghi, and C. Zhang. What is being transferred in transfer learning? In *Advances in Neural Information Processing Systems*, 2020.

[42] T. Nguyen, M. Raghu, and S. Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*, 2021.

[43] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, pages 722–729. IEEE, 2008.

[44] B. N. Oreshkin, P. Rodriguez, and A. Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *arXiv preprint arXiv:1805.10123*, 2018.

[45] V. Papyan, X. Han, and D. L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40): 24652–24663, 2020.

[46] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3498–3505. IEEE, 2012.

[47] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[48] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio. Transfusion: Understanding transfer learning for medical imaging. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[49] R. Ranjan, C. D. Castillo, and R. Chellappa. L2-constrained softmax loss for discriminative face verification. *arXiv preprint arXiv:1703.09507*, 2017.

[50] M. Ren, S. Ravi, E. Triantafillou, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018.

[51] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[52] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

[53] A. Schilling, C. Metzner, J. Rietsch, R. Gerum, H. Schulze, and P. Krauss. How deep is deep enough?–quantifying class separability in the hidden layers of deep neural networks. *arXiv preprint arXiv:1811.01753*, 2018.

[54] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014.

[55] R. Shwartz-Ziv and N. Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[56] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[57] D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[59] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[60] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[61] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*, 2020.

[62] S. Wager, S. Wang, and P. S. Liang. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems*, pages 351–359, 2013.

[63] S. Wager, W. Fithian, S. Wang, and P. S. Liang. Altitude training: Strong bounds for single-layer dropout. In *Advances in Neural Information Processing Systems*, 2014.

[64] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international Conference on Multimedia*, pages 1041–1049, 2017.

[65] F. Wang, J. Cheng, W. Liu, and H. Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018.

[66] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu. Cosface: Large margin cosine loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5265–5274, 2018.

[67] C. Wei, J. D. Lee, Q. Liu, and T. Ma. Regularization matters: Generalization and optimization of neural nets vs their induced kernel. In *Advances in Neural Information Processing Systems*, pages 9712–9724, 2019.

[68] C. Wei, S. Kakade, and T. Ma. The implicit and explicit regularization effects of dropout. In *International Conference on Machine Learning*, 2020.

[69] N. Wojke and A. Bewley. Deep cosine metric learning for person re-identification. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 748–756. IEEE, 2018.

[70] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3485–3492. IEEE, 2010.

[71] Y. Xu, Y. Xu, Q. Qian, H. Li, and R. Jin. Towards understanding label smoothing. *arXiv preprint arXiv:2006.11653*, 2020.

[72] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discuss limitations in Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provide training details in Appendix A.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We do not report the standard deviations across seeds in all tables because it is not possible to do so while keeping the text at a readable size. However, we perform appropriate statistical testing based on the standard deviations across seeds and bold-face all numbers in each table that are not statistically significant from the best.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]