arXiv:2004.11362v1 [cs.LG] 23 Apr 2020

Supervised Contrastive Learning

Prannay Khosla*
Google Research
prannayk@google.com

Piotr Teterwak*
Google Research
pteterwak@google.com

Chen Wang
Google Research
wangch@google.com

Aaron Sarna
Google Research
sarna@google.com

Yonglong Tian
MIT
yonglong@mit.edu

Phillip Isola
MIT
phillipi@mit.edu

Aaron Maschinot
Google Research
amaschinot@google.com

Ce Liu
Google Research
celiu@google.com

Dilip Krishnan
Google Research
dilipkay@google.com

Abstract

Cross entropy is the most widely used loss function for supervised training of image classification models. In this paper, we propose a novel training methodology that consistently outperforms cross entropy on supervised learning tasks across different architectures and data augmentations. We modify the batch contrastive loss, which has recently been shown to be very effective at learning powerful representations in the self-supervised setting. We are thus able to leverage label information more effectively than cross entropy. Clusters of points belonging to the same class are pulled together in embedding space, while simultaneously pushing apart clusters of samples from different classes. In addition to this, we leverage key ingredients such as large batch sizes and normalized embeddings, which have been shown to benefit self-supervised learning. On both ResNet-50 and ResNet-200, we outperform cross entropy by over 1%, setting a new state of the art number of 78.8% among methods that use AutoAugment data augmentation. The loss also shows clear benefits for robustness to natural corruptions on standard benchmarks on both calibration and accuracy. Compared to cross entropy, our supervised contrastive loss is more stable to hyperparameter settings such as optimizers or data augmentations.

1. Introduction

The cross-entropy loss is the most widely used loss function for supervised learning. It is naturally defined as the KL-divergence between two discrete distributions: the label distribution (a discrete distribution of 1-hot vectors) and the empirical distribution of the logits. A number of works have explored shortcomings with this loss, such as lack of robustness to noisy labels [59, 44] and the possibility of poor margins [14, 30], leading to reduced generalization performance. In practice, however, most proposed alternatives do not seem to have worked better for large-scale datasets, such as ImageNet [11], as evidenced by the continued use of cross-entropy to achieve state of the art results [9, 10, 51, 26].

Many proposed improvements to regular cross-entropy in fact involve a loosening of the definition of the loss, specifically that the reference distribution is axis-aligned. These improvements are often motivated in dif-

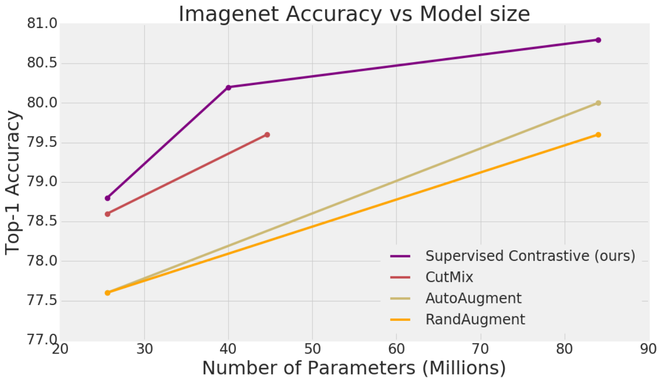


Figure 1: Our supervised contrastive loss outperforms the cross entropy loss with standard data augmentations such as AutoAugment[9] and RandAugment[10]; we also compare to CutMix [55]). We show results on ResNet-50, ResNet-101 and ResNet-200, and compare against the same ResNet architectures for other techniques (except CutMix models for which we compare against ResNeXt-101).

[*] Work done as part of Google AI Residency

- Proposes to replace cross-entropy loss function with a modified batch contrastive loss that is generally used in SSL.
- It is well known that cross-entropy has a lot of shortcomings like poor robustness, reduced generalisation performance and poor calibration.
- The performance improvements here in terms of accuracy may not be that much (though it is over 1%), but if this method improves robustness compared to other methods like label smoothing, etc., that would be a big deal.

We can leverage labels information when using contrastive methods in a supervised setting. Same class -> positive examples

In SSL, we don't have labels. So, we generate the positive examples for an image by doing multiple augmentations

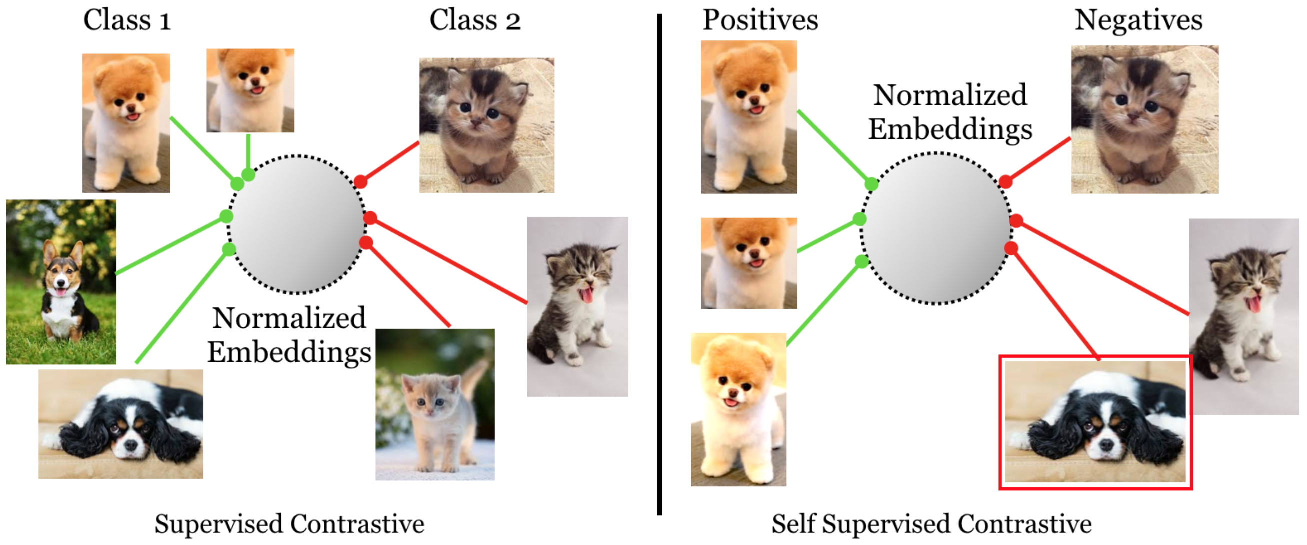


Figure 2: Supervised vs. self-supervised contrastive losses: In the supervised contrastive loss considered in this paper (left), positives from one class are contrasted with negatives from other classes (since labels are provided); images from the same class are mapped to nearby points in a low-dimensional hypersphere. In self-supervised contrastive loss (right), labels are not provided. Hence positives are generated as data augmentations of a given sample (crops, flips, color changes etc.), and negatives are randomly sampled from the mini-batch. This can result in false negatives (shown in bottom right), which may not be mapped correctly, resulting in a worse representation.

ferent ways. Label smoothing [45] makes a fuzzy distinction between correct and incorrect labels by moving off-axis, which provides a small but significant boost in many applications [33]. In self-distillation [24], multiple rounds of cross-entropy training are performed by using the "soft" labels from previous rounds as reference class distributions. Mixup [56] and related data augmentation strategies create explicit new training examples, often by linear interpolation, and then apply the same linear interpolation to the target label distribution, akin to a softening of the original cross entropy loss. Models trained with these modifications show improved generalization, robustness, and calibration.

In this work, we propose a new loss for supervised training which completely does away with a reference distribution; instead we simply impose that normalized embeddings from the same class are closer together than embeddings from different classes. Our loss is directly inspired by the family of *contrastive* objective functions, which have achieved excellent performance in self-supervised learning in recent years in the image and video domains [50, 25, 21, 19, 46, 6, 43] and have connections to the large literature on metric learning [48, 5].

As the name suggests, contrastive losses consist of two "opposing forces": for a given *anchor* point, the first force pulls the anchor closer in representation space to other points, and the second force pushes the anchor farther away from other points. The former set is known as *positives*, and the latter as *negatives*. Our key technical novelty in this work is to consider *many positives* per anchor in addition to many negatives (as opposed to the convention in self-supervised contrastive learning which uses only a single positive). We use provided labels to select the positives and negatives. Fig. 2 and Fig. 3 provide a visual explanation of our proposed loss.

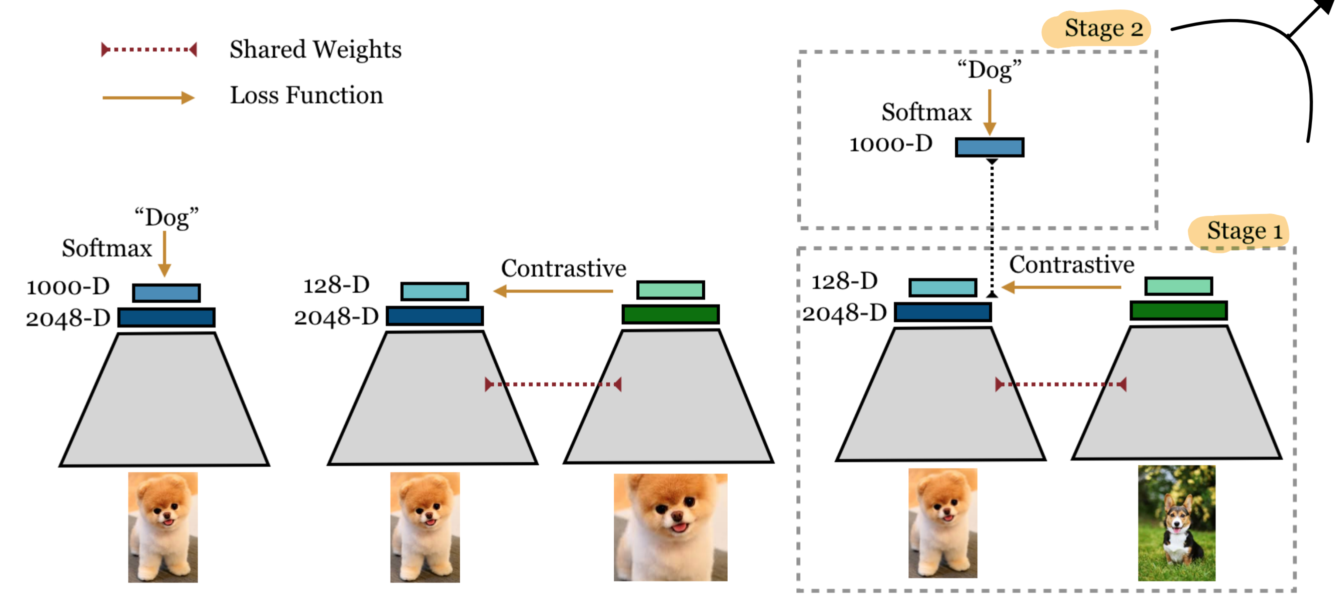
Difference in SSL and this technique

The resulting loss is stable to train, as our empirical results show. It achieves very good top-1 accuracy on the ImageNet dataset on the ResNet-50 and ResNet-200 architectures [20]. On ResNet-50 with Auto-Augment [9], we achieve a top-1 accuracy of 78.8%, which is a 1.6% improvement over the cross-entropy loss with the same data augmentation and architecture (see Fig. 1). The gain in top-1 accuracy is also accompanied by increased robustness as measured on the ImageNet-C dataset [22]. Our main contributions are summarized below:

1. We propose a novel extension to the contrastive loss function that allows for multiple positives per anchor. We thus adapt contrastive learning to the fully supervised setting.

2. We show that this loss allows us to learn state of the art representations compared to cross-entropy, giving significant boosts in top-1 accuracy and robustness.

One of the key differences is that in this work, we are not only considering a lot of negatives for each positive as done in SSL, but we are also considering many positives as well. The selection of positives and negatives is done using labels



2 stage process

(a) Supervised Cross Entropy (b) Self Supervised Contrastive (c) Supervised Contrastive

Figure 3: Cross entropy, self-supervised contrastive loss and supervised contrastive loss: The cross entropy loss (left) uses labels and a softmax loss to train a model; the self-supervised contrastive loss (middle) uses a contrastive loss and data augmentations to learn representations about classes; the supervised contrastive loss (right) proposed in this paper has two stages; in the first stage we use labels to choose the images for a contrastive loss. In the second stage, we freeze the learned representations and then learn a classifier on a linear layer using a softmax loss: thus combining the benefits of using labels and contrastive losses.

3. Our loss is less sensitive to a range of hyperparameters than cross-entropy. This is an important practical consideration. We believe that this is due to the more natural formulation of our loss that pulls representations of samples from the same class to be pulled closer together, rather than forcing them to be pulled towards a specific target as done in cross-entropy.

4. We show analytically that the gradient of our loss function encourages learning from hard positives and hard negatives. We also show that triplet loss [48] is a special case of our loss when only a single positive and negative are used.

2. Related Work

Our work draws on existing literature in self-supervised representation learning, metric learning and supervised learning. Due to the large amount of literature, we focus on the most relevant papers. The cross-entropy loss was introduced as a powerful loss function to train deep networks [38, 1, 28]. The key idea is simple and intuitive: each class is assigned a target (usually 1-hot) vector and the logits at the last layer of the network, after a softmax transformation, are gradually transformed towards the target vector. However, it is unclear why these target labels should be the optimal ones; some work has been done into identifying better target labels vectors e.g. [52].

In addition, a number of papers have studied other drawbacks of the cross-entropy loss, such as sensitivity to noisy labels [59, 44], adversarial examples [14, 34], and poor margins [4]. Alternative losses have been proposed; however, the more popular and effective ideas in practice have been approaches that change the reference label distribution, such as label smoothing [45, 33], data augmentations such as Mixup [56] and CutMix [55], and knowledge distillation [24].

Recent years have seen significantly more powerful self-supervised representation learning approaches based on deep learning models, and exploiting structure in the data. In the language domain, state of the art models learn pre-trained embeddings by predicting masked out tokens in a sentence or paragraph e.g. [12, 53, 31]. Downstream fine-tuning is then used to achieve excellent results on tasks such as sentiment classification and question answering. Due to the very general nature of pre-training, a huge amount of unlabeled data can be utilized, along with very large architectures.

Remember Meta Pseudo labels paper? Read about that more here: https://bit.ly/2Rt9xkw

- The main difference in contrastive and triplet losses is that the triplet uses only one positive and one negative sample per anchor whereas contrastive methods tend to use one positive but many negatives per sample. The negatives are chosen randomly with the assumption that the prob of choosing a FN is very low.
- Apart from using contrastive loss and normalised embedding and inner product for similarity, this paper uses heavy data augmentation, and two stage training: contrastive followed by cross-entropy. This method tries to disentangle classes at the final layer.

In the image domain, predictive approaches have also been used to learn embeddings [13, 57, 58, 35]: the typical setup is that some part of the signal is left out and we try to predict that portion from other parts of the signal. To do this effectively requires the network to learn some semantic information about the signal, especially when passed through a bottleneck of lower dimension. However, the accurate prediction of high dimensional signals such as images is very difficult. A more powerful approach has been to replace a dense per-pixel predictive loss in input space with a loss in lower-dimensional representation space. A powerful family of models for self-supervised representation learning are collected under the umbrella of contrastive learning [50, 21, 25, 46, 41, 6]. In these works, the losses are inspired by noise contrastive estimation [17, 32] or N-pair losses [43]. Typically, the loss is applied at the last layer of a deep network. At test time, the embeddings from a previous layer are utilized for downstream transfer tasks, fine tuning or direct retrieval tasks.

Closely related to contrastive learning are metric learning and triplet losses [7, 48, 40]. These losses have been used to learn powerful representations, often in supervised settings, where labels are used to guide the choice of positive and negative pairs. The key distinction between triplet losses and contrastive losses is the number of positive and negative pairs per data point. The triplet loss uses just one positive and one negative pair. In the supervised metric learning setting, the positive pair is chosen from the same class or category and the negative pair is chosen from other classes, often using hard-negative mining [40]. Self-supervised contrastive losses similarly use just one positive pair, selected using either co-occurence [21, 25, 46] or using data augmentation [6]. The major difference is that many negative pairs are used for each data-point. These are usually chosen uniformly at random using some form of weak knowledge, such as patches from other images, or frames from other randomly chosen videos, relying on the assumption that this approach yields a very low probability of false negatives.

Most similar to our supervised contrastive is the soft-nearest neighbors loss introduced in [39] and used in [49]. Similar to [49], we improve upon [39] by normalizing the embeddings and replacing euclidean distance with inner products. We further improve on [49] by the increased use of data augmentation, a disposable contrastive head and two-stage training (contrastive followed by cross-entropy). These distinctions help us achieve state-of-the-art top-1 accuracies for ImageNet on the ResNet-50 and ResNet-200 architectures [20]. In [49] introduces the approximation of only backpropagating through part of the loss, and also the approximation of using stale representations in the form of memory bank. By only contrasting against samples in the current mini-batch, we are able to remove these approximations. Furthermore, our specific loss formulation makes our learning gradient efficient (see Section 3.2.3 for more details). The work in [15] also uses a similar loss formulation to ours; however it is used to *entangle* classes at intermediate layers by maximizing, instead of *disentangling* classes at the final layer as is done in our work.

3. Method

In this section, we start by reviewing the contrastive learning loss for self-supervised representation learning, as used in recent papers that achieve state of the art results [36, 21, 46, 6]. Then we show how we can modify this loss to be suitable for fully supervised learning, while simultaneously preserving properties important to the self-supervised approach. A natural transition between self-supervision and full supervision is semi-supervision, but we do not consider that paradigm in this paper.

3.1. Representation Learning Framework

Our representation learning framework is structurally similar to that used in [46, 6] for self-supervised contrastive learning and consists of the following components (see Fig. 2 and Fig. 3 for an illustration of the difference between the supervised and self-supervised scenarios).

- A *data augmentation* module, A(·), which transforms an input image, $\boldsymbol{x}$, into a randomly augmented image, $\tilde{\boldsymbol{x}}$. For each input image, we generate two randomly augmented images, each of which represents a different *view* of the data and thus contains some subset of the information in the original input image. The first stage of augmentation is applying a random crop to the image and then resizing that back to the image's native resolution. In light of the findings of [6] that self-supervised contrastive loss requires significantly different data augmentation than cross-entropy loss, for the second stage we evaluate three different options:
  - AutoAugment: [9]
  - RandAugment: [10]
  - SimAugment: A variant of the strategy of [6] to sequentially apply random color distortion and Gaussian blurring, where we probabilistically add an additional sparse image warp to the end of the sequence.

- Data augmentation is very much similar to SSL. For each input, two augmented images are generated. Also, the data augmentation strategy for the two stages is different.

- For the first stage, random crops resized to original size are used while for second stage, three different augmentation strategies are evaluated namely, AutoAugment, RandAugment and SimAugment

- An encoder projects image data to a N dimensional vector. This vector is then normalised to the unit sphere.
- This normalised representation is projected to a M dimensional space (M << N) using a projection head. The projected vector is normalized again to the unit sphere. Also, the projection head is used only during the training and is replaced by a linear layer.

- An *encoder network*, $\mathrm{E}(\cdot)$, which maps an augmented image $\tilde{\boldsymbol{x}}$ to a representation vector, $\boldsymbol{r} = \mathrm{E}(\tilde{\boldsymbol{x}}) \in \mathcal{R}^{D_E}$. In our framework, both augmented images for each input image are separately input to the same encoder, resulting in a pair of representation vectors. We experiment with two commonly used encoder architectures, ResNet-50 and ResNet-200 [20], where the activations of the final pooling layer ($D_E = 2048$) are used as the representation vector. This representation layer is always normalized to the unit hypersphere in $\mathcal{R}^{D_E}$. We find from experiments that this normalization always improves performance, consistent with other papers that have used metric losses e.g. [40]. We also find that the new supervised loss is able to train both of these architectures to a high accuracy with no special hyperparameter tuning. In fact, as reported in Sec. 4, we found that the supervised contrastive loss was less sensitive to small changes in hyperparameters, such as choice of optimizer or data augmentation.
- A *projection network*, $\mathrm{P}(\cdot)$, which maps the normalized representation vector $\boldsymbol{r}$ into a vector $\boldsymbol{z} = \mathrm{P}(\boldsymbol{r}) \in \mathcal{R}^{D_P}$ suitable for computation of the contrastive loss. For our projection network, we use a multi-layer perceptron [18] with a single hidden layer of size 2048 and output vector of size $D_P = 128$. We again normalize this vector to lie on the unit hypersphere, which enables using an inner product to measure distances in the projection space. The projection network is only used for training the supervised contrastive loss. After the training is completed, we discard this network and replace it with a single linear layer (for more details see Sec. 4). Similar to the results for self-supervised contrastive learning [46, 6], we found representations from the encoder to give improved performance on downstream tasks than those from the projection network. Thus our inference-time models contain exactly the same number of parameters as their cross-entropy equivalents.

3.2. Contrastive Losses: Self-Supervised and Supervised

We seek to develop a contrastive loss function that allows for an impactful incorporation of labeled data while at the same time preserves the beneficial properties of contrastive losses which have been paramount to the success of self-supervised representation learning. Similar to self-supervised contrastive learning, we generate minibatches by randomly sampling the data. For a set of N randomly sampled image/label pairs, $\{\boldsymbol{x}_k, \boldsymbol{y}_k\}_{k=1...N}$, the corresponding minibatch used for training consists of $2N$ pairs, $\{\tilde{\boldsymbol{x}}_k, \tilde{\boldsymbol{y}}_k\}_{k=1...2N}$, where, $\tilde{\boldsymbol{x}}_{2k}$ and $\tilde{\boldsymbol{x}}_{2k-1}$ are two random augmentations of $\boldsymbol{x}_k$ ($k = 1...N$) and $\tilde{\boldsymbol{y}}_{2k-1} = \tilde{\boldsymbol{y}}_{2k} = \boldsymbol{y}_k$.

3.2.1 Self-Supervised Contrastive Loss

Within a minibatch, let $i \in \{1...2N\}$ be the index of an arbitrary augmented image, and let $j(i)$ be the index of the other augmented image originating from the same source image. In *self-supervised* contrastive learning (e.g., [6, 46, 21, 25]), the loss takes the following form.

$$\mathcal{L}^{self} = \sum_{i=1}^{2N} \mathcal{L}_i^{self} \tag{1}$$

$$\mathcal{L}_i^{self} = -\log \frac{\exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_{j(i)}/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{i\neq k} \cdot \exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_k/\tau\right)} \tag{2}$$

Contrastive loss function in self-supervised settings

where $\boldsymbol{z}_\ell = \mathrm{P}(\mathrm{E}(\tilde{\boldsymbol{x}}_\ell))$ (remember that $\mathrm{P}(\cdot)$ and $\mathrm{E}(\cdot)$ refer to the projection and encoder networks), $\mathbb{1}_{\mathrm{B}} \in \{0, 1\}$ is an indicator function that returns 1 iff B evaluates as true, and $\tau > 0$ is a scalar temperature parameter. Within the context of Eq. 2, index i is called the *anchor*, index $j(i)$ is called the *positive*, and the other $2(N-1)$ indices ($k = 1...2N, k \notin \{i, j\}$) are called the *negatives*. $\boldsymbol{z}_i \bullet \boldsymbol{z}_{j(i)}$ computes an inner (dot) product between the normalized vectors $\boldsymbol{z}_i$ and $\boldsymbol{z}_{j(i)}$ in 128-dimensional space. Note that for each anchor i, there is 1 positive pair and $2N-2$ negative pairs. The denominator has a total of $2N-1$ terms (the positive and negatives).

It is insightful to consider the effects on the encoder due to minimizing Eq. 1. During training, for any i, the encoder is tuned to maximize the numerator of the log argument in Eq. 2 while simultaneously minimizing its denominator. The constraint that the term $\exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_{j(i)}\right)$ is present in both the numerator and the denominator ensures that the log argument goes no higher than 1, and since Eq. 1 sums over all pairs of indices ((i, j) and (j, i)), the encoder is restricted from minimizing the denominator or maximizing the numerator without doing the other as well. As a result, the encoder learns to map similar views to neighboring representations while mapping dissimilar ones to non-neighboring ones.

- Pause for a moment and think about how this loss function affects the state of the encoder. Why is it better than CE?
- If you look at the numerator and denominator closely, you will realise that the encoder will tend to maximize the numerator. But the term exp(Zi, Zji) is present both in the numerator as well as the denominator.
- This in turn creates a beautiful constraint for the encoder: it cannot maximize the numerator without minimizing the denominator. The only way to achieve this is to learn how to map similar views of the same image in a close neighbourhood while making sure to push the dissimilar ones farther away.

Shortcoming of the above form of contrastive loss function

3.2.2 Supervised Contrastive Loss

For supervised learning, the contrastive loss in Eq. 2 is incapable of handling the case where more than one sample is known with certainty to belong to the same class. To generalize the loss to handle arbitrary numbers of positives belonging to the same class, we propose the following novel loss function:

$$\mathcal{L}^{sup} = \sum_{i=1}^{2N} \mathcal{L}_i^{sup} \tag{3}$$

$$\boxed{\mathcal{L}_i^{sup} = \frac{-1}{2N_{\tilde{\boldsymbol{y}}_i} - 1} \sum_{j=1}^{2N} \mathbb{1}_{i \neq j} \cdot \mathbb{1}_{\tilde{\boldsymbol{y}}_i = \tilde{\boldsymbol{y}}_j} \cdot \log \frac{\exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_j / \tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{i \neq k} \cdot \exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_k / \tau\right)}} \tag{4}$$

where $N_{\tilde{\boldsymbol{y}}_i}$ is the total number of images in the minibatch that have the same label, $\tilde{\boldsymbol{y}}_i$, as the anchor, i. This loss has important properties well suited for supervised learning:

- **Generalization to an arbitrary number of positives.** The major structural change of Eq. 4 over Eq. 2 is that now, for any anchor, *all* positives in a minibatch (i.e., the augmentation-based one as well as any of the remaining $2(N-1)$ entries that are from the same class) contribute to the numerator. For minibatch sizes that are large with respect to the number of classes, multiple additional terms will be present (on average, $N_{L_i} = N/C$, where C is the number of classes). The loss encourages the encoder to give closely aligned representations to *all* entries from the same class in each instance of Eq. 4, resulting in a more robust clustering of the representation space that that generated from Eq. 2, as will be supported by our experiments in Sec. 4.

- **Contrastive power increases with more negatives.** The general form of the self-supervised contrastive loss (Eq. 4) is largely motivated by noise contrastive estimation and N-pair losses [17, 43], wherein the ability to discriminate between signal and noise (negatives) is improved by adding more examples of negatives. This property has been shown to be important to representation learning via self-supervised contrastive learning, with many studies showing increased performance with increasing number of negatives [21, 19, 46, 6]. The supervised contrastive loss in Eq. 4 preserves this structure: adding larger numbers of negatives to the denominator provides increased contrast for the positives.

Importance of more negative samples for a contrastive or similar kind of loss in any setting

By using many positives and many negatives, we are able to better model both *intra-class* and *inter-class* variability. As we expect intuitively, and supported by our experiments in Sec. 4, this translates to representations that are provide improved generalization, since they better capture the representation for a particular class.

3.2.3 Supervised Contrastive Loss Gradient Properties

We now provide further motivation for the form of the supervised contrastive loss in Eq. 4 by showing that its gradient has a structure that naturally causes learning to focus more on *hard* positives and negatives (i.e., ones against which continuing to contrast the anchor *greatly* benefits the encoder) rather than on *weak* ones (i.e., ones against which continuing to contrast the anchor only *weakly* benefits the encoder). The loss can thus be seen to be efficient in its training. Other contrastive losses, such as triplet loss [48], often use the computationally expensive technique of hard negative mining to increase training efficacy [40]. As a byproduct of this analysis, we motivate the addition of a normalization layer at the end of the projection network, since its presence allows the gradient to have this structure.

As shown in the supplementary (see Sec. 10), if we let $\boldsymbol{w}$ denote the projection network output immediately *prior* to normalization (i.e., $\boldsymbol{z} = \boldsymbol{w}/\|\boldsymbol{w}\|$), then the gradients of Eq. 4 with respect to $\boldsymbol{w}$ has the form:

$$\frac{\partial \mathcal{L}_i^{sup}}{\partial \boldsymbol{w}_i} = \left.\frac{\partial \mathcal{L}_i^{sup}}{\partial \boldsymbol{w}_i}\right|_{\text{pos}} + \left.\frac{\partial \mathcal{L}_i^{sup}}{\partial \boldsymbol{w}_i}\right|_{\text{neg}} \tag{5}$$

where:

$$\left.\frac{\partial \mathcal{L}_i^{sup}}{\partial \boldsymbol{w}_i}\right|_{\text{pos}} \propto \sum_{j=1}^{2N} \mathbb{1}_{i \neq j} \cdot \mathbb{1}_{\tilde{\boldsymbol{y}}_i = \tilde{\boldsymbol{y}}_j} \cdot \left((\boldsymbol{z}_i \bullet \boldsymbol{z}_j) \cdot \boldsymbol{z}_i - \boldsymbol{z}_j \right) \cdot (1 - P_{ij}) \tag{6}$$

$$\left.\frac{\partial \mathcal{L}_i^{sup}}{\partial \boldsymbol{w}_i}\right|_{\text{neg}} \propto \sum_{j=1}^{2N} \mathbb{1}_{i \neq j} \cdot \mathbb{1}_{\tilde{\boldsymbol{y}}_i = \tilde{\boldsymbol{y}}_j} \cdot \sum_{k=1}^{2N} \mathbb{1}_{k \notin \{i,j\}} \cdot \left(\boldsymbol{z}_k - (\boldsymbol{z}_i \bullet \boldsymbol{z}_k) \cdot \boldsymbol{z}_i \right) \cdot P_{ik} \tag{7}$$

- Check the supplementary material for annotations for this part where:

$$P_{i\ell} = \frac{\exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_\ell/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{i\neq k} \cdot \exp\left(\boldsymbol{z}_k \bullet \boldsymbol{z}_\ell/\tau\right)} \quad , \quad i,\ell \in \{1...2N\}\ , \ i \neq \ell \tag{8}$$

is the ℓ'th component of the temperature-scaled softmax distribution of inner products of representations with respect to anchor i and is thus interpretable as a probability. Eq. 6 generally includes contributions from the positives in the minibatch, while Eq. 7 includes those for negatives. We now show that easy positives and negatives have small gradient contributions while hard positives and negatives have large ones. For an easy positive, $\boldsymbol{z}_i \bullet \boldsymbol{z}_j \approx 1$ and thus P_{ij} is large. Thus (see Eq. 6):

$$\boxed{\|((\boldsymbol{z}_i \bullet \boldsymbol{z}_j) \cdot \boldsymbol{z}_i - \boldsymbol{z}_j)\| \cdot (1 - P_{ij}) = \sqrt{1 - (\boldsymbol{z}_i \bullet \boldsymbol{z}_j)^2} \cdot (1 - P_{ij}) \approx 0} \tag{9}$$

However, for a hard positive, $\boldsymbol{z}_i \bullet \boldsymbol{z}_j \approx 0$ and P_{ij} is moderate, so:

$$\boxed{\|((\boldsymbol{z}_i \bullet \boldsymbol{z}_j) \cdot \boldsymbol{z}_i - \boldsymbol{z}_j)\| \cdot (1 - P_{ij}) = \sqrt{1 - (\boldsymbol{z}_i \bullet \boldsymbol{z}_j)^2} \cdot (1 - P_{ij}) > 0} \tag{10}$$

Thus, for weak positives, where further contrastive efforts are of diminishing returns, the contribution to $\left\|\nabla_{\boldsymbol{z}_i} \mathcal{L}_{i,\text{pos}}^{sup}\right\|$ is small, while for hard positives, where further contrastive efforts are still needed, the contribution is large. For a weak negative ($\boldsymbol{z}_i \bullet \boldsymbol{z}_k \approx -1$) and a hard negative ($\boldsymbol{z}_i \bullet \boldsymbol{z}_k \approx 0$), analogous calculations of $\|(\boldsymbol{z}_k - (\boldsymbol{z}_i \bullet \boldsymbol{z}_k) \cdot \boldsymbol{z}_i)\| \cdot P_{ik}$ from Eq. 7 give similar conclusions: the gradient contribution is large for hard negatives and small for weak ones. As shown in the supplementary, the general $((\boldsymbol{z}_i \bullet \boldsymbol{z}_\ell) \cdot \boldsymbol{z}_\ell - \boldsymbol{z}_\ell)$ structure, which plays a key role in ensuring the gradients are large for hard positives and negatives, appears only if a normalization layer is added to the end of the projection network, thereby justifying the use of a normalization in the network.

3.3. Connections to Triplet Loss

Contrastive learning is closely related to the triplet loss [48], which is one of the widely-used alternatives to cross-entropy for supervised representation learning. As discussed in Sec 2, the triplet loss has been used to generate robust representations via supervised settings where hard negative mining leads to efficient contrastive learning [40]. The triplet loss, which can only handle one positive and negative at a time, can be shown to be a special case of the contrastive loss when the number of positives and negatives are each one. Assuming the representation of the anchor and the positive are more aligned than that of the anchor and negative ($\boldsymbol{z}_a \bullet \boldsymbol{z}_p \gg \boldsymbol{z}_a \bullet \boldsymbol{z}_n$), we have:

$$\begin{aligned}
\mathcal{L}_{con} &= -\log \frac{\exp\left(\boldsymbol{z}_a \bullet \boldsymbol{z}_p/\tau\right)}{\exp\left(\boldsymbol{z}_a \bullet \boldsymbol{z}_p/\tau\right) + \exp\left(\boldsymbol{z}_a \bullet \boldsymbol{z}_n/\tau\right)} \\
&= \log\left(1 + \exp\left(\left(\boldsymbol{z}_a \bullet \boldsymbol{z}_n - \boldsymbol{z}_a \bullet \boldsymbol{z}_p\right)/\tau\right)\right) \\
&\approx \exp\left(\left(\boldsymbol{z}_a \bullet \boldsymbol{z}_n - \boldsymbol{z}_a \bullet \boldsymbol{z}_p\right)/\tau\right) \qquad \text{(Taylor expansion of log)} \\
&\approx 1 + \frac{1}{\tau} \cdot \left(\boldsymbol{z}_a \bullet \boldsymbol{z}_n - \boldsymbol{z}_a \bullet \boldsymbol{z}_p\right) \\
&= 1 - \frac{1}{2\tau} \cdot \left(\left\|\boldsymbol{z}_a - \boldsymbol{z}_n\right\|^2 - \left\|\boldsymbol{z}_a - \boldsymbol{z}_p\right\|^2\right) \\
&\propto \left\|\boldsymbol{z}_a - \boldsymbol{z}_p\right\|^2 - \left\|\boldsymbol{z}_a - \boldsymbol{z}_n\right\|^2 + 2\tau
\end{aligned}$$

which has the same form as a triplet loss with margin $\alpha = 2\tau$. This result is consistent with empirical results [6] which show that contrastive loss performs better in general than triplet loss on representation tasks. Additionally, whereas triplet loss in practice requires computationally expensive hard negative mining (e.g., [40]), the discussion in the previous section shows that the gradients of the supervised contrastive loss naturally impose a measure of hard negative reinforcement during training. This of course comes at the cost of requiring large batch sizes to allow for the inclusion of many positives and negatives, some of which will be hard in expectation as training proceeds. → Pain doesn't go away! 😂😂

Real pain

4. Experiments

We evaluate our supervised contrastive loss by measuring classification accuracy on ImageNet and robustness to common image corruptions [22]. After training the embedding network with supervised contrastive loss on ImageNet [11], we replace the projection head of the network with a a new randomly initialized linear dense (fully connected) layer. This linear layer is trained with standard cross entropy while the parameters of the embedding network are kept unchanged.

Loss	Architecture	Top-1	Top-5
Cross Entropy (baselines)	AlexNet [27]	56.5	84.6
	VGG-19+BN [42]	74.5	92.0
	ResNet-18 [20]	72.1	90.6
	MixUp ResNet-50 [56]	77.4	93.6
	CutMix ResNet-50 [55]	78.6	94.1
	Fast AA ResNet-50 [9]	77.6	95.3
	Fast AA ResNet-200 [9]	80.6	95.3
Cross Entropy (our implementation)	ResNet-50	77.0	92.9
	ResNet-200	78.0	93.3
Supervised Contrastive	ResNet-50	**78.8**	**93.9**
	ResNet-200	**80.8**	**95.6**

Table 1: Top-1/Top-5 accuracy results on ImageNet on ResNet-50 and ResNet-200 with AutoAugment [9] being used as the augmentation for Supervised Contrastive learning. Achieving 78.8% on ResNet-50, we outperform all of the top methods whose performance is shown above. Baseline numbers are taken from the referenced papers and we also additionally re-implement cross-entropy ourselves for fair comparison.

Loss	Architecture	rel. mCE	mCE
Cross Entropy (baselines)	AlexNet [27]	100.0	100.0
	VGG-19+BN [42]	122.9	81.6
	ResNet-18 [20]	103.9	84.7
Cross Entropy (our implementation)	ResNet-50	103.7	68.4
	ResNet-200	96.6	69.4
Supervised Contrastive	ResNet-50	**87.5**	**64.4**
	ResNet-200	**77.1**	**57.2**

Table 2: Training with Supervised Contrastive Loss makes models more robust to corruptions in images, as measured by Mean Corruption Error (mCE) and relative mCE over the ImageNet-C dataset [22] (lower is better).

4.1. ImageNet Classification Accuracy

Using the linear evaluation protocol as described above, we find that networks trained using our supervised contrastive loss give state-of-the-art results on ImageNet. Table 1 shows results for ResNet-50 and ResNet-200 (we use ResNet-v1 [20]). The supervised contrastive loss performs better than cross entropy for both architectures that we considered by over 1%. We achieve a new state of the art accuracy of 78.8% on ResNet-50 with AutoAugment (for comparison, a number of the other top-performing methods are shown in Table 1). Note that we also achieve a slight improvement over CutMix [55], which is considered to be a state of the art data augmentation strategy. Incorporating data augmentation strategies such as CutMix [55] and MixUp [56] into supervised contrastive learning could potentially improve results further. However, mixing labels blurs the interpretation of our loss, so we leave such experiments for future work.

4.2. Robustness to Image Corruptions and Calibration

Deep neural networks often lack robustness to out of distribution data or natural corruptions. This has been shown not only with adversarially constructed examples [16], but also with naturally occurring variations such as noise, blur and JPEG compression [22]. To this end, [22] made a benchmark dataset, ImageNet-C, which applies common naturally occuring perturbations such as noise, blur and contrast changes to the ImageNet dataset. In Table 2, we compare the supervised contrastive models to cross entropy using the mean Corruption Error (mCE) and relative mean Corruption Error (rel. mCE) metrics [22].

We see that the supervised contrastive models have lower mCE values across different corruptions, thus showing their increased robustness. We believe this increased robustness reflects the more powerful representations that are learnt by the contrastive loss function. In Fig. 5, it is seen that the supervised contrastive methods retain high accuracy at high corruption severities while having low expected calibration errors. More details are provided in the supplementary material.

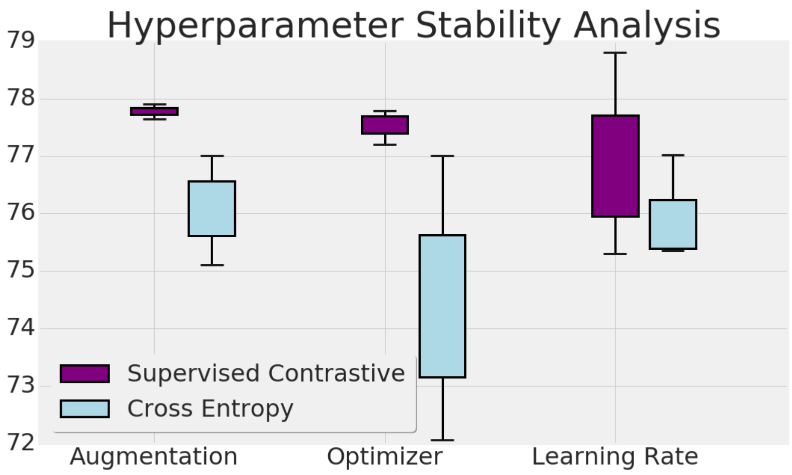


Figure 4: Comparison of top-1 accuracy variability of cross entropy and supervised contrastive loss to changes in hyperparameters. We compare three augmentations (RandAugment [10], AutoAugment [9] and SimAugment) (left plot); three optimizers (LARS, SGD with Momentum and RMSProp); and 3 learning rates that vary from the optimal rate by a factor of 10 smaller or larger. The supervised contrastive loss is more stable to changes in hyperparameters.

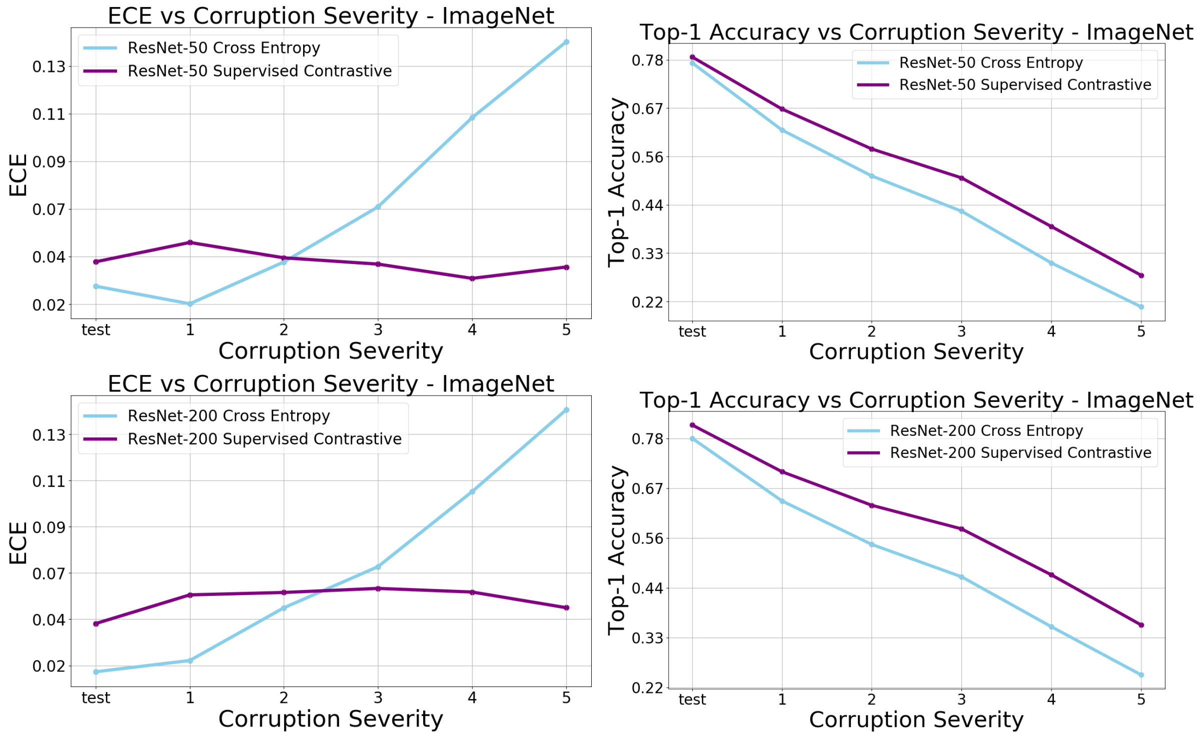


Figure 5: Expected Calibration Error and mean top-1 accuracy at different corruption severities on ImageNet-C, on the ResNet-50 architecture (top) and ResNet-200 architecture (bottom). The contrastive loss maintains a higher accuracy over the range of corruption severities, and does not suffer from increasing calibration error, unlike the cross entropy loss.

4.3. Hyperparameter Stability

Deep network training is well known to be sensitive to hyper-parameters and a large body of literature is devoted to finding efficient ways to perform hyperparameter tuning [2, 8, 3]. We find that the contrastive supervised loss is more stable to changes in hyperparameters. In Figure 4, we compare the top-1 accuracy of our loss against cross-entropy for different

BEST THING IMHO! 😂😂

optimizers, data augmentations, and learning rates. We see significantly lower variance in the output of the contrastive loss when changing optimizer and data augmentation. While we do not have a theoretical explanation of this behaviour, we conjecture that this is due to the smoother geometry of the hypersphere compared to labels which are the endpoints of the n-dimensional simplex (as cross-entropy requires). Note that the minibatch sizes for cross entropy and supervised contrastive are the same, thus ruling out effects related to batch size. We experiment with hyperparameter stability by changing augmentations, optimizers and learning rates one at a time from the best combination for each of the methodologies.

4.4. Effect of Number of Positives

We run ablations to test the effect of the number of positives in Eq. 4. Table 3 shows the steady benefit of adding more positives for a ResNet-50 model trained on ImageNet with supervised contrastive loss. The trade-off is that more positives corresponds to a higher computational cost at training time. However, this is a highly parallelizable computation. Note that for each experiment, the number of positives always contains one positive which is the same sample but with a different data augmentation; and the remainder of the positives are different samples from the same class. Under this definition, self-supervised learning is considered as having 1 positive.

Number of positives	1 [6]	2	3	5
Top-1 Accuracy	69.3	78.1	78.2	78.8

Table 3: Comparison of Top-1 accuracy variability as a function of the number of positives $N_{\tilde{\boldsymbol{y}}_i}$ in Eq. 4 varies from 1 to 5. Adding more positives benefits the final Top-1 accuracy. We compare against previous state of the art self-supervised work [6] which has used **one** positive which is another data augmentation of the *same sample*; see text for details.

4.5. Training Details

The supervised contrastive loss was trained for up to 700 epochs during the pretraining stage. We found that using half the number of epochs (350) only dropped the top-1 accuracy by a small amount. Each training step is about 50% slower than cross-entropy. This is due to the need to compute cross-products between every element of the minibatch and every other element (Eq. 4). The supervised contrastive loss needs an (optional) additional step of training a final linear classifier to compute top-1 accuracy. However, this is not needed if the purpose is to use representations for transfer learning tasks or retrieval.

We trained our models with batch sizes of up to 😤😤 8192, although batch sizes of 2048 suffice for most purposes for both supervised contrastive and cross entropy losses. We report metrics for experiments with batch size 8192 for ResNet-50 and batch size 2048 for ResNet-200 (due to the larger network size, a smaller batch size is necessary). We observed that for a fixed batch size it was possible to train with supervised contrastive loss using larger learning rates and for a smaller number of epochs than what was required by cross entropy to achieve similar performance. Additionally, we observe that a small number of steps suffice to train the dense layers on top of the frozen embedding network, and we see minimal degradation in performance by training for as few as 10 epochs. All our results used a temperature of $\tau = 0.07$ and note that smaller temperature benefit training more than higher ones. But, lower temperatures can be sometimes harder to train due to numerical stability issues. We also find that AutoAugment [9] gives the best results for both Supervised Contrastive and Cross Entropy and we report ablations with other augmentations in the supplementary. We experimented with standard optimizers such as LARS [54], RMSProp [23] and SGD with momentum [37] in different permutations for the initial pre-training step and training of the dense layer. While the momentum optimizer works best for training ResNets with cross entropy, we get the best performance for supervised contrastive loss by using LARS for pre-training and RMSProp for training the dense layer on the top of the frozen network. We give detailed results for combination of optimizers in the supplementary section of the paper.

5. Discussion

We have presented a novel loss, inspired by contrastive learning, that outperforms cross entropy on classification accuracy and robustness benchmarks. Furthermore, our experiments show that this loss is less sensitive to hyperparameter changes, which could be a useful practical consideration. The loss function provides a natural connection between fully unsupervised training on the one end, and fully supervised training on the other. This opens the possibility of applications in semi-supervised learning which can leverage the benefits of a single loss that can smoothly shift behavior based on the availability of labeled data.

References

[1] Eric B Baum and Frank Wilczek. Supervised learning of probability distributions by neural networks. In *Neural information processing systems*, pages 52–61, 1988. 3

[2] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012. 9

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*, 2017. 9

[4] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems*, pages 1565–1576, 2019. 3

[5] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(Mar):1109–1135, 2010. 2

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 2, 4, 5, 6, 7, 10, 14, 15, 16

[7] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005. 4

[8] Marc Claesen and Bart De Moor. Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*, 2015. 9

[9] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019. 1, 2, 4, 8, 9, 10

[10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019. 1, 4, 9, 16

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 2009. 1, 7

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 3

[13] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015. 4

[14] Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. In *Advances in neural information processing systems*, pages 842–852, 2018. 1, 3

[15] Nicholas Frosst, Nicolas Papernot, and Geoffrey E. Hinton. Analyzing and improving representations with the soft nearest neighbor loss. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2012–2020. PMLR, 2019. 4

[16] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 8

[17] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010. 4, 6

[18] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001. 5

[19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019. 2, 6

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 4, 5, 8

[21] Olivier J Hénaff, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019. 2, 4, 5, 6

[22] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019. 2, 7, 8, 14, 15

[23] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8), 2012. 10

[24] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2, 3

[25] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019. 2, 4, 5

[26] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Large scale learning of general visual representations for transfer. *arXiv preprint arXiv:1912.11370*, 2019. 1

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 8

[28] Esther Levin and Michael Fleisher. Accelerated learning in layered neural networks. *Complex systems*, 2:625–640, 1988. 3

[29] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *arXiv preprint arXiv:1905.00397*, 2019. 15

[30] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, volume 2, page 7, 2016. 1

[31] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 3

[32] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273, 2013. 4

[33] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pages 4696–4705, 2019. 2, 3

[34] Kamil Nar, Orhan Ocal, S Shankar Sastry, and Kannan Ramchandran. Cross-entropy loss and low-rank features have responsibility for adversarial examples. *arXiv preprint arXiv:1901.08360*, 2019. 3

[35] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 4

[36] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 4

[37] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 10

[38] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. 3

[39] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, pages 412–419, 2007. 4

[40] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 4, 5, 6, 7, 14

[41] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018. 4

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 8

[43] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems*, pages 1857–1865, 2016. 2, 4, 6

[44] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014. 1, 3

[45] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 2, 3

[46] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 2, 4, 5, 6, 14

[47] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012. 15

[48] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009. 2, 3, 4, 6, 7

[49] Zhirong Wu, Alexei A Efros, and Stella Yu. Improving generalization via scalable neighborhood component analysis. In *European Conference on Computer Vision (ECCV) 2018*, 2018. 4

[50] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018. 2, 4

[51] Qizhe Xie, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Self-training with noisy student improves imagenet classification. *arXiv preprint arXiv:1911.04252*, 2019. 1

[52] Shuo Yang, Ping Luo, Chen Change Loy, Kenneth W Shum, and Xiaoou Tang. Deep representation learning with target coding. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. 3

[53] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019. 3

[54] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 10, 15

[55] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019. 1, 3, 8, 16

[56] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2, 3, 8, 16

[57] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 4

[58] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1058–1067, 2017. 4

[59] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, pages 8778–8788, 2018. 1, 3

- The gradient of the loss function includes the temperature parameter T and the norm is inversely proportional to it.
- This means the norm would become smaller and smaller as we increase the temperature but after one point the gains would diminish as expected.
- As the magnitude of the gradients become smaller, it makes it easy to use larger learning rates.
- If you compare contrastive loss with the triplet loss by taking only one positive and one negative example per anchor, then the temperature parameter in contrastive loss is proportional to the margin used in the triplet loss

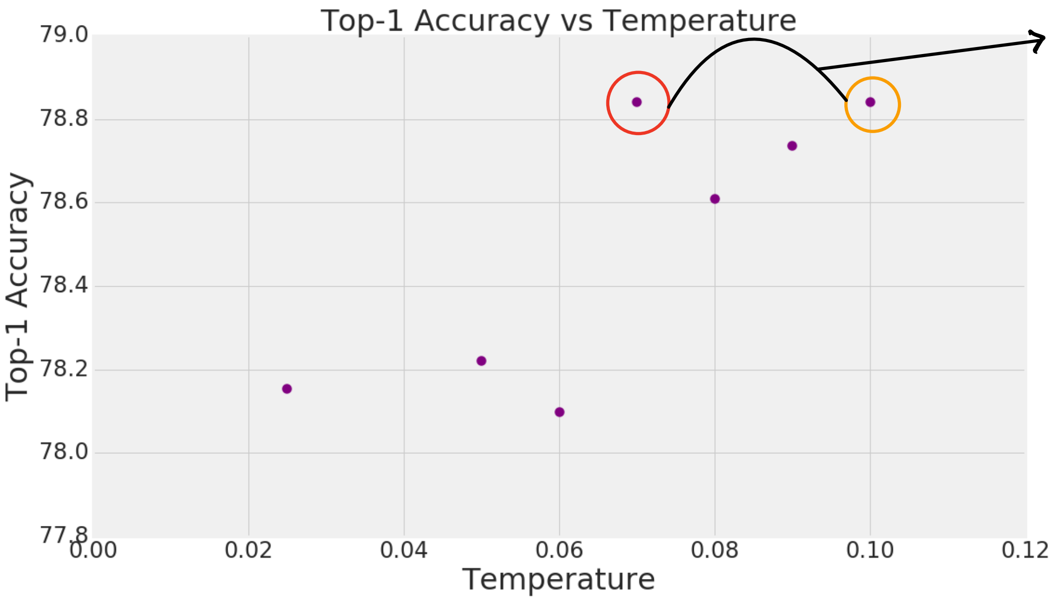
Supplementary

6. Effect of Temperature in Loss Function

Similar to previous work [6, 46], we find that the temperature used in the loss function (for the softmax) has an important role to play in supervised contrastive learning and that the model trained with the optimal temperature can outperform ablations up to 3%. Two competing effects that changing the temperature has on training the model are:

1. **Smoothness:** The distances in the representation space used for training the model have gradients with smaller norm ($||\nabla\mathcal{L}|| \propto \frac{1}{\tau}$); see Section 10. Smaller magnitude gradients make the optimization problem simpler by allowing for larger learning rates. In Section 3.3 of the paper, it is shown that in the case of a single positive and negative, the contrastive loss is equivalent to a triplet loss with margin $\propto \tau$. Therefore, in these cases, a larger temperature makes the optimization easier, and classes more separated.
2. **Hard negatives:** On the other hand, hard negatives have shown to improve classification accuracy when models are trained with the triplet loss [40]. Low temperatures are equivalent to optimizing for hard negatives: for a given batch of samples and a specific anchor, lowering the temperature increases the value of P_{ik} (see Eq. 8) for samples which have larger inner product with the anchor, and reduces it for samples which have smaller inner product. Further the magnitude of gradient coming from a given sample k belonging to a different class than the anchor is proportional to the probability P_{ik}. Therefore the model derives a large amount of training signal from samples which belong to a different class but it finds hard to separate from the given anchor, which is by definition a *hard negative*.

Based on numerical experiments, we found a temperature of 0.07 to be optimal for top-1 accuracy on ResNet-50; results on various temperatures are shown in Fig. 6. We use the same temperature for all experiments on ResNet-200, as well.



As we keep increasing the temp, accuracy improves up to a point and then the gains start to diminish. Temperature is the true hyper parameter to tune here.

Figure 6: Low temperatures benefit representation learning with contrastive losses since they put more weight on harder negatives. Alternatively extremely low temperatures shape the geometry in the representation space to be considerably less smooth, making the optimization problem more difficult. → Shouldn't increasing T=1.0 provide the best results then? Think about it 😛😛

7. Robustness

Along with measuring the mean Corruption Error (mCE) and mean relative Corruption Error [22] on the ImageNet-C dataset (see paper, Section 4.2 and Table 2), we also measure the Expected Calibration Error and the mean accuracy of our models on different severities of the corruption. The aim is to understand how performance and calibration degrades as the data shifts farther from the training distribution and becomes harder to classify. Table 4 shows the results. We clearly observe that models trained with contrastive learning do not see degradation of calibration and show a lower degradation of top-1 accuracy as the corruption severity increases. This shows that the model learns better representations which are robust to corruptions.

Model		Test	1	2	3	4	5
Loss	Architecture	ECE					
Cross Entropy	ResNet-50	0.03	0.02	0.04	0.07	0.11	0.15
	ResNet-200	0.02	0.03	0.05	0.07	0.11	0.15
Supervised Contrastive	ResNet-50	0.04	0.05	0.04	0.04	0.03	0.04
	ResNet-200	0.04	0.06	0.06	0.06	0.06	0.05
		Top-1 Accuracy					
Cross Entropy	ResNet-50	77.1	61.7	51.1	43.0	31.1	21.0
	ResNet-200	77.8	63.8	54.1	46.9	35.8	25.0
Supervised Contrastive	ResNet-50	78.4	66.5	57.3	50.6	39.5	28.2
	ResNet-200	80.8	70.3	62.8	57.6	47.3	36.1

cool!

Table 4: **Top**: Average Expected Calibration Error (ECE) over all the corruptions in ImageNet-C [22] for a given level of severity (lower is better); **Bottom**: Average Top-1 Accuracy over all the corruptions for a given level of severity (higher is better).

8. Comparison with Cross Entropy

We also compared against using models trained with cross-entropy loss for representation learning. We do this by first training the model with cross entropy and then re-initializing the final layer of the network. In this second stage of training we again train with cross entropy but keep the weights of the network fixed. Table 5 shows that the representations learnt by cross-entropy for a ResNet-50 network are not robust and just the re-initialization of the last layer leads to large drop in accuracy and a mixed result on robustness compared to a single-stage cross-entropy training. Both methods of training cross-entropy are inferior to supervised contrastive loss.

Proves that representations learnt using CE aren't robust enough

	Accuracy	mCE	rel. mCE
Supervised Contrastive	78.8	64.4	87.5
Cross Entropy (1 stage)	77.1	68.4	103.7
Cross Entropy (2 stage)	73.7	73.3	92.9

Using CE instead of contrastive loss

Table 5: Comparison between representations learnt using SupCon and representations learnt using Cross Entropy loss with either 1 stage of training or 2 stages (representation learning followed by linear classifier).

9. Training Details

In this section we provide the details of the experiments we ran to find the best set of optimizers and data augmentations to train supervised contrastive models.

9.1. Optimizer

We experiment with various optimizers for the contrastive learning and training the linear classifier in various combinations. We present our results in Table 6. The LARS optimizer [54] gives us the best results to train the embedding network, confirming what has been reported by previous work [6]. With LARS we use a cosine learning rate decay. On the other hand we find that the RMSProp optimizer [47] works best for training the linear classifier. For RMSProp we use an exponential decay for the learning rate.

9.2. Data Augmentation

We experimented with various data augmentations and found that AutoAugment [29] gave us the best results in terms of downstream performance. We also note that AutoAugment is faster to implement than other augmentation schemes such

Contrastive Optimizer	Linear Optimizer	Top-1 Accuracy
LARS	RMSProp	78.6
LARS	LARS	77.9
RMSProp	RMSProp	77.8
LARS	Momentum	77.6
Momentum	RMSProp	73.4

Table 6: Results of training the ResNet-50 architecture with AutoAugment data augmentation policy for 350 epochs and then training the linear classifier for another 350 epochs. Learning rates were optimized for every optimizer while all other hyper-parameters were kept the same.

as RandAugment [10] or the data augmentations proposed in [6], which we denote SimAugment. As we show in Table 7, using the same data augmentation (AutoAugment) for both pre-training and training the linear classifier is optimal. We leave experimenting with MixUp [56] or CutMix [55] as future work.

Contrastive Augmentation	Linear classifier Augmentation	Accuracy
AutoAugment	AutoAugment	78.6
AutoAugment	RandAugment	78.1
AutoAugment	SimAugment	75.4
SimAugment	AutoAugment	76.1
SimAugment	RandAugment	75.9
SimAugment	SimAugment	77.9
RandAugment	AutoAugment	78.3
RandAugment	RandAugment	78.4
RandAugment	SimAugment	76.3

Table 7: Combinations of different data augmentations for ResNet-50 trained with optimal set of hyper-parameters and optimizers. We observe the best performance when the same data augmentation is used for both pre-training and training the linear classifier on top of the frozen embedding network.

Further we experiment with varying levels of augmentation magnitude for RandAugment since that has shown to affect performance when training models with cross entropy loss [10]. Fig. 7 show that supervised contrastive methods consistently outperform cross entropy training independent of augmentation magnitude.

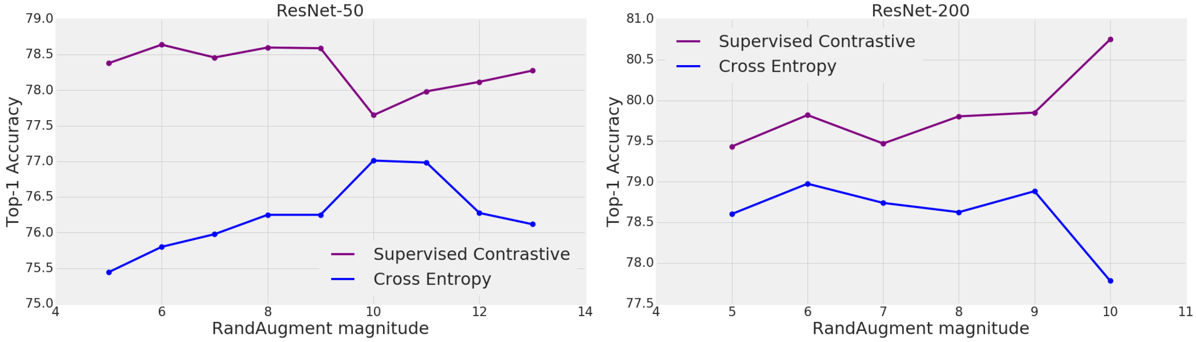


Figure 7: Top-1 Accuracy vs RandAugment magnitude for ResNet-50 (left) and ResNet-200 (right). We see that supervised contrastive methods consistently outperform cross entropy for varying strength of augmentation.

10. Derivation of Supervised Contrastive Learning Gradient

In Sec 2 in the main paper, we presented motivation based on the functional form of the gradient of the supervised contrastive loss, $\mathcal{L}_i^{sup}(\boldsymbol{z}_i)$ (Eq. 4 in the paper), that the supervised contrastive loss intrinsically causes learning to focus on *hard* positives and negatives, where the encoder can greatly benefit, instead of easy ones, where the encoder can only minimally benefit. In this section, we derive the mathematical expression for the gradient:

$$\boxed{\frac{\partial \mathcal{L}_i^{sup}(\boldsymbol{z}_i)}{\partial \boldsymbol{w}_i} = \frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{w}_i} \cdot \frac{\partial \mathcal{L}_i^{sup}(\boldsymbol{z}_i)}{\partial \boldsymbol{z}_i}} \tag{11}$$

where $\boldsymbol{w}_i$ is the projection network output *prior* to normalization, i.e., $\boldsymbol{z}_i = \boldsymbol{w}_i / \|\boldsymbol{w}_i\|$. As we will show, normalizing the representations provides structure to the gradient that causes learning to focus on hard positives and negatives instead of easy ones. *

The supervised contrastive loss can be rewritten as:

$$\begin{aligned}
\mathcal{L}_i^{sup}(\boldsymbol{z}_i) &= \frac{-1}{2N_{\tilde{\boldsymbol{y}}_i} - 1} \sum_{j=1}^{2N} \mathbb{1}_{i \neq j} \cdot \mathbb{1}_{\tilde{\boldsymbol{y}}_i = \tilde{\boldsymbol{y}}_j} \cdot \log \frac{\exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_j / \tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{i \neq k} \cdot \exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_k / \tau\right)} \\
&= \frac{-1}{2N_{\tilde{\boldsymbol{y}}_i} - 1} \sum_{j=1}^{2N} \mathbb{1}_{i \neq j} \cdot \mathbb{1}_{\tilde{\boldsymbol{y}}_i = \tilde{\boldsymbol{y}}_j} \cdot \left(\frac{\boldsymbol{z}_i \bullet \boldsymbol{z}_j}{\tau} - \log \sum_{k=1}^{2N} \mathbb{1}_{i \neq k} \cdot \exp \frac{\boldsymbol{z}_i \bullet \boldsymbol{z}_k}{\tau} \right) \rightarrow \text{Log expansion nothing else}
\end{aligned}$$

Thus:

$$\begin{aligned}
\frac{\partial \mathcal{L}_i^{sup}(\boldsymbol{z}_i)}{\partial \boldsymbol{z}_i} &= \frac{-1}{(2N_{\tilde{\boldsymbol{y}}_i} - 1) \cdot \tau} \cdot \sum_{j=1}^{2N} \mathbb{1}_{i \neq j} \cdot \mathbb{1}_{\tilde{\boldsymbol{y}}_i = \tilde{\boldsymbol{y}}_j} \cdot \left(\boldsymbol{z}_j - \sum_{k=1}^{2N} \frac{\mathbb{1}_{i \neq k} \cdot \boldsymbol{z}_k \cdot \exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_k / \tau\right)}{\sum_{m=1}^{2N} \mathbb{1}_{i \neq m} \cdot \exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_m / \tau\right)} \right) \\
&= \frac{-1}{(2N_{\tilde{\boldsymbol{y}}_i} - 1) \cdot \tau} \cdot \sum_{j=1}^{2N} \mathbb{1}_{i \neq j} \cdot \mathbb{1}_{\tilde{\boldsymbol{y}}_i = \tilde{\boldsymbol{y}}_j} \cdot \left(\boldsymbol{z}_j - \sum_{k=1}^{2N} \mathbb{1}_{i \neq k} \cdot \boldsymbol{z}_k \cdot P_{ik} \right) \\
&= \boxed{\frac{-1}{(2N_{\tilde{\boldsymbol{y}}_i} - 1) \cdot \tau} \cdot \sum_{j=1}^{2N} \mathbb{1}_{i \neq j} \cdot \mathbb{1}_{\tilde{\boldsymbol{y}}_i = \tilde{\boldsymbol{y}}_j} \cdot \left((1 - P_{ij}) \cdot \boldsymbol{z}_j - \sum_{k=1}^{2N} \mathbb{1}_{k \notin \{i,j\}} \cdot \boldsymbol{z}_k \cdot P_{ik} \right)}
\end{aligned} \tag{12}$$

where we have defined $P_{i\ell}$ as follows:

$$\boxed{P_{i\ell} = \frac{\exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_\ell / \tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{i \neq k} \cdot \exp\left(\boldsymbol{z}_i \bullet \boldsymbol{z}_\ell / \tau\right)} \quad , \quad i, \ell \in \{1...2N\} \; , \; i \neq \ell}$$

$P_{i\ell}$ is the ℓ'th component of the temperature-scaled softmax distribution of inner products of representations with respect to anchor i and is thus interpretable as a probability. Note that, were we *not* to normalize the projection network output representations, then Eq. 12 would effectively be the gradient used for learning (simply let $\boldsymbol{z}_i$ denote a non-normalized vector). It will be insightful to contrast Eq. 12 with the form for the projection network gradient (derived below) in which output representations *are* normalized.

The fact that we are using *normalized* projection network output representations introduces an additional term in Eq. 11, namely $\partial \boldsymbol{z}_i / \partial \boldsymbol{w}_i$, which has the following form:

$$\begin{aligned}
\frac{\partial \boldsymbol{z}_i}{\partial \boldsymbol{w}_i} &= \frac{\partial}{\partial \boldsymbol{w}_i} \left(\frac{\boldsymbol{w}_i}{\|\boldsymbol{w}_i\|} \right) \\
&= \frac{1}{\|\boldsymbol{w}_i\|} \cdot \mathrm{I} - \boldsymbol{w}_i \cdot \left(\frac{\partial \left(1 / \|\boldsymbol{w}_i\|\right)}{\partial \boldsymbol{w}_i} \right)^T \\
&= \frac{1}{\|\boldsymbol{w}_i\|} \left(\mathrm{I} - \frac{\boldsymbol{w}_i \cdot \boldsymbol{w}_i^T}{\|\boldsymbol{w}_i\|^2} \right) \\
&= \frac{1}{\|\boldsymbol{w}_i\|} \left(\mathrm{I} - \boldsymbol{z}_i \cdot \boldsymbol{z}_i^T \right) \rightarrow \text{Identity matrix}
\end{aligned} \tag{13}$$

where I is the identity matrix. Substituting Eqs. 12 and 13 into Eq. 11 gives the following expression for the gradient wrt the *normalized* projection network output representations:

$$\begin{aligned}
\frac{\partial \mathcal{L}_i^{sup}}{\partial \boldsymbol{w}_i} &= \frac{-1}{(2N_{\tilde{\boldsymbol{y}}_i}-1)\cdot\tau}\cdot\frac{1}{\|\boldsymbol{w}_i\|}\left(\mathrm{I}-\boldsymbol{z}_i\cdot\boldsymbol{z}_i^T\right) \\
&\quad\cdot\sum_{j=1}^{2N}\mathbb{1}_{i\neq j}\cdot\mathbb{1}_{\tilde{\boldsymbol{y}}_i=\tilde{\boldsymbol{y}}_j}\cdot\left((1-P_{ij})\cdot\boldsymbol{z}_j-\sum_{k=1}^{2N}\mathbb{1}_{k\notin\{i,j\}}\cdot\boldsymbol{z}_k\cdot P_{ik}\right) \\
&= \frac{1}{(2N_{\tilde{\boldsymbol{y}}_i}-1)\cdot\tau\cdot\|\boldsymbol{w}_i\|}\cdot\sum_{j=1}^{2N}\mathbb{1}_{i\neq j}\cdot\mathbb{1}_{\tilde{\boldsymbol{y}}_i=\tilde{\boldsymbol{y}}_j}\cdot\left((\boldsymbol{z}_i\bullet\boldsymbol{z}_j)\cdot\boldsymbol{z}_j-\boldsymbol{z}_j\right)(1-P_{ij}) \\
&\quad+\frac{1}{(2N_{\tilde{\boldsymbol{y}}_i}-1)\cdot\tau\cdot\|\boldsymbol{w}_i\|}\cdot\sum_{j=1}^{2N}\mathbb{1}_{i\neq j}\cdot\mathbb{1}_{\tilde{\boldsymbol{y}}_i=\tilde{\boldsymbol{y}}_j}\cdot\sum_{k=1}^{2N}\mathbb{1}_{k\notin\{i,j\}}\cdot\left(\boldsymbol{z}_k-(\boldsymbol{z}_i\bullet\boldsymbol{z}_k)\cdot\boldsymbol{z}_k\right)\cdot P_{ik} \\
&= \left.\frac{\partial \mathcal{L}_i^{sup}}{\partial \boldsymbol{w}_i}\right|_{\text{pos}}+\left.\frac{\partial \mathcal{L}_i^{sup}}{\partial \boldsymbol{w}_i}\right|_{\text{neg}}
\end{aligned}$$

where:

$$\begin{aligned}
\left.\frac{\partial \mathcal{L}_i^{sup}}{\partial \boldsymbol{w}_i}\right|_{\text{pos}} &= \frac{1}{(2N_{\tilde{\boldsymbol{y}}_i}-1)\cdot\tau\cdot\|\boldsymbol{w}_i\|}\cdot\sum_{j=1}^{2N}\mathbb{1}_{i\neq j}\cdot\mathbb{1}_{\tilde{\boldsymbol{y}}_i=\tilde{\boldsymbol{y}}_j}\cdot\left((\boldsymbol{z}_i\bullet\boldsymbol{z}_j)\cdot\boldsymbol{z}_j-\boldsymbol{z}_j\right)(1-P_{ij}) \\
\left.\frac{\partial \mathcal{L}_i^{sup}}{\partial \boldsymbol{w}_i}\right|_{\text{neg}} &= \frac{1}{(2N_{\tilde{\boldsymbol{y}}_i}-1)\cdot\tau\cdot\|\boldsymbol{w}_i\|}\cdot\sum_{j=1}^{2N}\mathbb{1}_{i\neq j}\cdot\mathbb{1}_{\tilde{\boldsymbol{y}}_i=\tilde{\boldsymbol{y}}_j}\cdot\sum_{k=1}^{2N}\mathbb{1}_{k\notin\{i,j\}}\cdot\left(\boldsymbol{z}_k-(\boldsymbol{z}_i\bullet\boldsymbol{z}_k)\cdot\boldsymbol{z}_k\right)\cdot P_{ik}
\end{aligned}$$

As discussed in detail in the main paper, for hard positives (i.e., $\boldsymbol{z}_i\bullet\boldsymbol{z}_j\approx 0$), the $((\boldsymbol{z}_i\bullet\boldsymbol{z}_j)\cdot\boldsymbol{z}_j-\boldsymbol{z}_j)(1-P_{ij})$ structure present in $\partial\mathcal{L}_i^{sup}/\partial\boldsymbol{w}_i|_{\text{pos}}$ results in large gradients while for easy positives ($\boldsymbol{z}_i\bullet\boldsymbol{z}_j\approx 1$), it results in small gradients. Analogously, for hard negatives ($\boldsymbol{z}_i\bullet\boldsymbol{z}_j\approx 0$), the $(\boldsymbol{z}_k-(\boldsymbol{z}_i\bullet\boldsymbol{z}_k)\cdot\boldsymbol{z}_k)\cdot P_{ik}$ structure present in $\partial\mathcal{L}_i^{sup}/\partial\boldsymbol{w}_i|_{\text{neg}}$ results in large gradients while for easy negatives ($\boldsymbol{z}_i\bullet\boldsymbol{z}_j\approx -1$), it results in small gradients. Comparing $\partial\mathcal{L}_i^{sup}/\partial\boldsymbol{w}_i|_{\text{pos}}$ and $\partial\mathcal{L}_i^{sup}/\partial\boldsymbol{w}_i|_{\text{neg}}$ to that of Eq. 12, one can see that normalizing the projection network output representations served to introduce the general $((\boldsymbol{z}_i\bullet\boldsymbol{z}_\ell)\cdot\boldsymbol{z}_\ell-\boldsymbol{z}_\ell)$ structure, which plays a key role in ensuring the gradients are large for hard positives and negatives.

- Gradients can be calculated using both normalized as well as non-normalized projections. Eqn. 12 represents the gradients when we use a non-normalized projection vector.
- But if we normalize the projections, it adds another term to the gradient of the loss function calculated in Eqn 12.
- The two terms combined introduces a general structure ((Zi * Zm) * Zm - Zm) which plays a key role in ensuring the gradients are large for hard positives and negatives. How?
- For a hard positive example, Zi Zm is almost zero, the above term makes sure that the gradient is large. Similarly for an easy positive, ZiZm would almost be equal to a value of 1. The above term ensures that the gradient is small in that case. Similar analogy holds for the hard negatives and easy negatives.
- This forces the network to focus more on hard positives and negatives which in turn forces the network to learn better representations. 🤩🤩👌🏼