Tuned Contrastive Learning — Supplementary

Chaitanya Animesh [1,2*] **Manmohan Chandraker**[1]

[1]UC San Diego [2]Otter.ai, Inc.

canimesh@ucsd.edu mkchandraker@ucsd.edu

1. Proofs for Theoretical Results

Proof for Lemma 1: Section 2 of the supplementary material of SupCon paper [7] gives a clear proof for Lemma 1 (refer to the derivation of L_{out}^{sup} in that section).

Lemma 2 *The gradient of the TCL loss per sample —* L_i^{tcl} *with respect to the normalized projection network embedding* z_i *is given by:*

$$\frac{\partial L_i^{tcl}}{\partial z_i} = \frac{1}{\tau}(\underbrace{\sum_{p \in P(i)} z_p(P_{ip}^t - X_{ip} - Y_{ip}^t)}_{\textit{Gradient from positives}} + \underbrace{\sum_{n \in N(i)} z_n P_{in}^t}_{\textit{Gradient from negatives}}) \tag{1}$$

where

$$X_{ip} = \frac{1}{|P(i)|} \tag{2}$$

$$P_{ip}^t = \frac{exp(z_i.z_p/\tau)}{D(z_i)} \tag{3}$$

$$Y_{ip}^t = \frac{\tau k_1 exp(-z_i.z_p)}{D(z_i)} \tag{4}$$

$$P_{in}^t = \frac{k_2 exp(z_i.z_n/\tau)}{D(z_i)} \tag{5}$$

Proof

$$L_i^{tcl} = \frac{-1}{|P(i)|}\sum_{p \in P(i)} \log(\frac{\exp(z_i.z_p/\tau)}{D(z_i)}) \tag{6}$$

$$\implies L_i^{tcl} = \frac{-1}{|P(i)|}\sum_{p \in P(i)} (\frac{z_i.z_p}{\tau} - \log(D(z_i)) \tag{7}$$

$$\begin{aligned}\implies \frac{\partial L_i^{tcl}}{\partial z_i} = \frac{-1}{\tau|P(i)|}\sum_{p \in P(i)}\Bigg(z_p - &\\ \frac{(\sum_{p' \in P(i)} z_{p'} \exp(z_i.z_{p'}/\tau))}{D(z_i)} &\\ + \frac{\tau k_1(\sum_{p' \in P(i)} z_{p'} \exp(-z_i.z_{p'}))}{D(z_i)} - &\\ \frac{k_2(\sum_{n \in N(i)} z_n \exp(z_i.z_n/\tau))}{D(z_i)}\Bigg)&\end{aligned} \tag{8}$$

$$\begin{aligned}\implies \frac{\partial L_i^{tcl}}{\partial z_i} = \frac{-1}{\tau|P(i)|}\Bigg[\sum_{p \in P(i)} z_p - &\\ \sum_{p \in P(i)} \frac{(\sum_{p' \in P(i)} z_{p'} \exp(z_i.z_{p'}/\tau))}{D(z_i)} &\\ + \sum_{p \in P(i)} \frac{\tau k_1(\sum_{p' \in P(i)} z_{p'} \exp(-z_i.z_{p'}))}{D(z_i)} &\\ - \sum_{p \in P(i)} \frac{k_2(\sum_{n \in N(i)} z_n \exp(z_i.z_n/\tau))}{D(z_i)}\Bigg]&\end{aligned} \tag{9}$$

$$\begin{aligned}\implies \frac{\partial L_i^{tcl}}{\partial z_i} = \frac{-1}{\tau|P(i)|}\Bigg[\sum_{p \in P(i)} z_p - &\\ \sum_{p' \in P(i)} \frac{(\sum_{p \in P(i)} z_{p'} \exp(z_i.z_{p'}/\tau))}{D(z_i)} &\\ + \sum_{p' \in P(i)} \frac{\tau k_1(\sum_{p \in P(i)} z_{p'} \exp(-z_i.z_{p'}))}{D(z_i)} - &\\ \sum_{p \in P(i)} \frac{k_2(\sum_{n \in N(i)} z_n \exp(z_i.z_n/\tau))}{D(z_i)}\Bigg]&\end{aligned} \tag{10}$$

[*]Work done as part of Master's Thesis at UC San Diego.

$$\implies \frac{\partial L_i^{tcl}}{\partial z_i} = \frac{-1}{\tau |P(i)|}\Bigg[\sum_{p \in P(i)} z_p - \sum_{p' \in P(i)} \frac{(|P(i)| z_{p'} \exp(z_i.z_{p'}/\tau))}{D(z_i)} + \sum_{p' \in P(i)} \frac{\tau k_1 (|P(i)| z_{p'} \exp(-z_i.z_{p'}))}{D(z_i)} - \frac{|P(i)| k_2 (\sum_{n \in N(i)} z_n \exp(z_i.z_n/\tau))}{D(z_i)}\Bigg] \tag{11}$$

$$\implies \frac{\partial L_i^{tcl}}{\partial z_i} = \frac{-1}{\tau |P(i)|}\Bigg[\sum_{p \in P(i)} z_p - \sum_{p \in P(i)} \frac{(|P(i)| z_p \exp(z_i.z_p/\tau))}{D(z_i)} + \sum_{p \in P(i)} \frac{\tau k_1 (|P(i)| z_p \exp(-z_i.z_p))}{D(z_i)} - \frac{|P(i)| k_2 (\sum_{n \in N(i)} z_n \exp(z_i.z_n/\tau))}{D(z_i)}\Bigg] \tag{12}$$

$$\implies \frac{\partial L_i^{tcl}}{\partial z_i} = \frac{-1}{\tau}\Bigg[\sum_{p \in P(i)} \frac{z_p}{|P(i)|} - \sum_{p \in P(i)} \frac{(z_p \exp(z_i.z_p/\tau))}{D(z_i)} + \sum_{p \in P(i)} \frac{\tau k_1 (z_p \exp(-z_i.z_p))}{D(z_i)} - \frac{k_2 (\sum_{n \in N(i)} z_n \exp(z_i.z_n/\tau))}{D(z_i)}\Bigg] \tag{13}$$

$$\implies \frac{\partial L_i^{tcl}}{\partial z_i} = \frac{1}{\tau}\Bigg[\sum_{p \in P(i)} z_p \Bigg(\frac{\exp(z_i.z_p/\tau)}{D(z_i)} - \frac{1}{|P(i)|} - \frac{\tau k_1 \exp(-z_i.z_p)}{D(z_i)}\Bigg) + \sum_{n \in N(i)} z_n \frac{k_2 \exp(z_i.z_n/\tau)}{D(z_i)}\Bigg] \tag{14}$$

This completes the proof.

Theorem 1 *For $k_1, k_2 \geq 1$, the magnitude of the gradient from a hard positive for TCL loss is strictly greater than the magnitude of the gradient from a hard positive for SupCon and hence, the following result follows:*

$$\underbrace{|X_{ip} - P_{ip}^t + Y_{ip}^t|}_{\text{(TCL's hard positive gradient)}} > \underbrace{|X_{ip} - P_{ip}^s|}_{\text{(Supcon's hard positive gradient)}} \tag{15}$$

Proof As the authors of [7] show in Section 3 of their supplementary (we also mention the same in our main paper in Section 3.1) that the gradient from a positive while flowing back through the projector into the encoder reduces to almost zero for easy positives and $|P_{ip}^s - X_{ip}|$ for a hard positive because of the normalization consideration in the projection network combined with the assumption that $z_i.z_p \approx 1$ for easy positives and $z_i.z_p \approx 0$ for hard positives. Proceeding in a similar manner, it is straightforward to see that the gradient response from a hard positive in case of TCL is $|P_{ip}^t - X_{ip} - Y_{ip}^t|$. We don't prove this explicitly again since the derivation will be identical to what authors [7] have already shown. One can refer section 3 of the supplementary of [7] for details.

Now, because $k_1, k_2 \geq 1$, it is easy to observe from equations 6 and 14 of our main paper that,

$$P_{ip}^t < P_{ip}^s \tag{16}$$

And from equation 15 of our main paper:

$$Y_{ip}^t > 0 \tag{17}$$

Hence, the result follows. This completes the proof.

Theorem 2 *For fixed k_1, the magnitude of the gradient response from a hard negative for TCL loss — P_{in}^t increases strictly with k_2.*

Proof

$$P_{in}^t = \frac{k_2 \exp(z_i.z_n/\tau)}{D(z_i)} \tag{18}$$

$$= \frac{k_2 \exp(z_i.z_n/\tau)}{D_1(z_i) + k_1 D_2(z_i) + k_2 D_3(z_i)} \tag{19}$$

where

$$D_1(z_i) = \sum_{p' \in P(i)} \exp(z_i.z_{p'}/\tau) \tag{20}$$

and

$$D_2(z_i) = (\sum_{p' \in P(i)} \exp(-z_i.z_{p'})) \tag{21}$$

and

$$D_3(z_i) = (\sum_{n \in N(i)} \exp(z_i.z_n/\tau)) \tag{22}$$

$$= \frac{\exp(z_i.z_n/\tau)}{(D_1(z_i) + k_1 D_2(z_i))/k_2 + D_3(z_i)} \tag{23}$$

It is now easy to observe that for a fixed k_1, P_{in}^t increases strictly with k_2. This completes the proof.

2. Training Details

2.1. Supervised Setting

We first present the common training details used for each dataset experiment in the supervised setting for SupCon [7] and TCL. Except for the contrastive training learning rate, every other detail presented is common for SupCon and TCL. As mentioned in our main paper, we train for a total of 150 epochs which involves 100 epochs of contrastive training for the encoder and the projector, and 50 epochs of cross-entropy training for the linear layer for both the losses. AutoAugment [3] is the common data augmentation method used except for FMNIST [9] for which we used a simple augmentation strategy consisting of random cropping and horizontal flip. We use cosine annealing based learning rate scheduler and SGD optimizer with momentum=0.9 and weight decay=$1e-4$ for both contrastive and linear layer training. Temperature τ is set to 0.1. For linear layer training, the starting learning rate is $5e-1$. ResNet-50 [6] is the common encoder architecture used. We use NVIDIA-GeForce-RTX-2080-Ti, NVIDIA-TITAN-RTX and NVIDIA-A100-SXM4-80GB GPUs for our experiments.

CIFAR-10 [8] Image size is resized to 32×32 in the data augmentation pipeline. We use a batch size of 128. For both SupCon and TCL we use a starting learning rate of $1e-1$ for contrastive training. We set $k_1 = 5000$ and $k_2 = 1$ for TCL.

CIFAR-100 [8] Image size is resized to 32×32 in the data augmentation pipeline. We use a batch size of 256. For both SupCon and TCL we use a starting learning rate of $2e-1$ for contrastive training. We set $k_1 = 4000$ and $k_2 = 1$ for TCL.

FMNIST [9] Image size is resized to 28×28 in the data augmentation pipeline. We use a batch size of 128. For both SupCon and TCL we use a starting learning rate of $9e-2$ for contrastive training. We set $k_1 = 5000$ and $k_2 = 1$ for TCL.

ImageNet-100 [5] Images are resized to 224×224 in the data-augmentation pipeline and batch size of 256 is used. For SupCon we use a starting learning rate of $2e-1$ for contrastive training while $3e-1$ for TCL. We set $k_1 = 4000$ and $k_2 = 1$ for TCL. **Note that we didin't run experiments on full ImageNet because we simply didn't have the resources to do so. As section 4.5 in the SupCon paper [7] mentions that for ResNet-50 evaluations on ImageNet, a batch size of 6144 (before augmentation) is used which means the batch size is effectively 6144X2=12,288. For such a large batch size with each image being 224X224, we will require easily around 50 large sized (high memory) co-located GPUs/cloud TPUs or even more which was just not possible for us and beyond our scope.** In our experiments, we avoided using momentum queue [2] or any kind of memory bank to ensure a fair and direct comparison between the TCL and SupCon loss functions. Including them would have obscured our ability to clearly assess how the two loss functions perform against each other.

We also ran experiments with different seeds to calculate 95% confidence intervals for top-1 accuracies of SupCon and TCL. For CIFAR-100 we repeated experiment 30 times with a different seed each time. For CIFAR-10 and FMNIST we repeated experiment 5 times with different seeds while for ImageNet-100, we repeated experiment 3 times with unique seeds. The experiment settings for calculating the confidence intervals are same as used in Section 4.1 of our main paper. Tab. 1 shows the confidence intervals obtained from the experiments and clearly suggests that TCL performs consistently better than the Supervised Contrastive Learning.

Dataset	SupCon	TCL
CIFAR-10	96.16 ± 0.11	96.30 ± 0.10
CIFAR-100	78.45 ± 0.64	79.30 ± 0.45
FashionMNIST	95.42 ± 0.08	95.58 ± 0.11
ImageNet-100	85.73 ± 0.15	86.53 ± 0.15

Table 1. 95% confidence intervals for top-1 accuracies of SupCon and TCL

We also present results for 250 epochs of training constituted by 200 epochs of contrastive training and 50 epochs of linear layer training in Tab. 2. As we see, TCL performs consistently better than Supervised Contrastive Learning [7]. Note that we didn't see any performance improvement for FMNIST dataset for either SupCon loss or TCL loss by running them for 250 epochs.

Dataset	SupCon	TCL
CIFAR-10	96.7	96.8
CIFAR-100	81.0	81.6
FashionMNIST	95.5	95.7
ImageNet-100	86.5	87.1

Table 2. Comparisons of top-1 accuracies of TCL with SupCon in supervised setting for 250 epochs of training.

2.2. Hyper-parameter Stability

For the hyper-parameter stability experiments we have presented most of the details in the main paper. We present the learning rates and values of k_1 and k_2 used for TCL. Remaining details are the same as the supervised setting experiments.

2.2.1 Encoder Architecture

The starting learning rate for contrastive training is $1e-1$ for all the encoders except ResNet-101 for which we used a value of $9e-2$. $k_1 = 5000$ and $k_2 = 1$ are the common values used for all the encoders.

2.2.2 Batch Size

For batch sizes=32, 64, 128, 256, 512 and 1024 we set the starting learning rates for contrastive training to $8e-3$, $9e-3$, $1e-1$, $2e-1$, $5e-1$ and 1 respectively. For batch size of 32 we used $k_1 = 5000$ and $k_2 = 1$. For batch size of 64 we used $k_1 = 7500$ and $k_2 = 1$. For batch size of 128 we used $k_1 = 5000$ and $k_2 = 1$. For batch sizes of 256, 512 and 1024 we used $k_1 = 4000$ and $k_2 = 1$.

2.2.3 Projection Network Embedding (z_i) Size

We used a common starting learning rate of $1e-1$ with $k_1 = 5000$ and $k_2 = 1$ for all the projector output sizes.

2.2.4 Augmentations

For AutoAugment [3] method, we use a learning rate of $1e-1$ with $k_1 = 5000$ and $k_2 = 1$. For SimAugment [1], we use a learning rate of $1e-1$ with $k_1 = 5000$ and $k_2 = 1.2$.

2.3. Self-Supervised Setting

For the self-supervised setting, we reuse the code provided by [4] and we are thankful to them for providing all the required details. The projector used for TCL is exactly the same as SimCLR for fair comparison and consists of one hidden layer of size 2048 and output size of 256. ResNet-18 is the common encoder used for all the methods. We use SGD optimizer with momentum=0.9 wrapped with LARS optimizer [10] and weight deacy of $1e-4$. Augmentation used is SimAugment [1] and is done in the same manner as [4]. Gaussian blur is used for self-supervised setting. We use NVIDIA-GeForce-RTX-2080-Ti, NVIDIA-TITAN-RTX and NVIDIA-A100-SXM4-80GB GPUs for our experiments.

CIFAR-10 [8] All methods do 1000 epochs of contrastive pre-training on CIFAR-10 and images are reshaped to 32×32 in the data augmentation pipeline. We use batch size=256, same as SimCLR. For TCL, we use a starting learning rate of $4e-1$ for contrastive pre-training with $k_1 = 1$ and $k_2 = 1.5$.

CIFAR-100 [8] All methods do 1000 epochs of contrastive pre-training on CIFAR-100 and images are reshaped to 32×32 in the data augmentation pipeline. We use batch size=256, same as SimCLR. For TCL, we use a starting learning rate of $4e-1$ for contrastive pre-training with $k_1 = 1$ and $k_2 = 1.5$.

ImageNet-100 [5] All methods do 400 epochs of contrastive pre-training on ImageNet-100 and images are rescaled to a size of 224×224. We use batch size=256, same as used by SimCLR. For TCL, we use a starting learning rate of $4e-1$ for contrastive pre-training with $k_1 = 1$ and $k_2 = 1.5$.

3. Choosing k_1 and k_2 for TCL

We observe that a value of k_1 in the range of 10^3 to 10^4 works the best with $k_1 = 4 \times 10^3$ or 5×10^3 almost always working on all datasets and configurations we experimented with. We generally start with these two values or otherwise with 2×10^3 and increase it in steps of 2000 till 8×10^3. We also observed during our experiments that choosing any value less than 5×10^3 always gave improvements in performance over SupCon loss. For most of our experiments we set k_1 to 4×10^3 or 5×10^3 and get the desired performance boost in a single run. We found k_2 to be useful to compensate for the reduction in the value of P^t_{in} caused by increasing k_1 and especially in self-supervised settings where hard negative gradient contribution is important. For setting k_2, we fix k_1 (which itself gives boost in performance) and increase k_2 in steps of 0.1 or 0.2 to see if we can get further improvement. As we see, we generally keep $k_2 = 1$ for supervised settings but we do sometimes set it to a value slightly bigger than 1. We set k_2 to a higher value in self-supervised settings as compared to supervised settings to get higher gradient contribution from hard negatives. Increasing k_1 didn't help much in boosting the performance in self-supervised setting (as we only had two positives per anchor) and so we set it to 1. Increasing k_2 also increases the gradient response from positives to some extent by decreasing P^t_{ip} and so, we found it sufficient to increase only k_2 and set k_1 to 1 in self-supervised setting.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020. 4

[2] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning, 2020. 3

[3] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 3, 4

[4] Victor Guilherme Turrisi da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. solo-learn: A library of self-supervised methods for visual representation learning. *Journal of Machine Learning Research*, 23(56):1–6, 2022. 4

[5] 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*, pages 248–255, 2009. 3, 4

[6] 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. 3

[7] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc., 2020. 1, 2, 3

[8] A. Krizhevsky and G Hinton. Learning multiple layers of features from tiny images, 2009. 3, 4

[9] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. 3

[10] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 4