Self-Supervised Contrastive Learning for Efficient Models

Cooper Ang [*1]

Abstract

This paper explores self-supervised contrastive learning using the SimCLR learning algorithm for smaller, efficient model architectures. The goal is to show that contrastive learning is effective for training efficient model architectures leading to faster inference capabilities at the edge. Specifically MobileNetV3 architectures were explored to determine their feasibility to classify CIFAR-10 images. I show through the results and analysis that MobileNetV3-Large with a fifth of the parameters of ResNet50 can still effectively train using contrastive learning resulting in a 13% degradation of top-1 accuracy. Trends in model size do persist with larger models resulting in better results. As for batch size, trends in the results show that the optimal batch size for smaller models are smaller than for larger models. Longer training does seem to help contrastive learning even for smaller architectures with no overfitting even after 1000 epochs.

1. Introduction

Advancements in deep learning have taken over most other approaches in pattern recognition tasks. The deep learning paradigm has led to huge advancements in natural language processing (NLP) and computer vision (CV). The advancement of machine learning (ML) systems is attributed to more data, better compute, and better algorithms. There are a few areas which should be explored more.

As ML inference at the edge increases in popularity, there becomes a need for more efficient, smaller neural network architectures to perform pattern recognition tasks with less compute overhead. This is especially the case in the rising field of Internet of Things (IoT), and on device machine learning for mobile phones. Inference at the edge provides advantages such as being able to inference without being connected to a network, and increased privacy for sensitive inferencing with user data.

Currently in the pattern recognition field the most popular paradigm is supervised learning. In this approach data and its corresponding labels are provided to "train" the network. This presents some challenges because labeled data is expensive to annotate and usually requires a human to manually label swaths of data. At the other end of the spectrum is unsupervised learning, which are classes of pattern recognition algorithms which attempt to learn patterns from unlabeled data. In between these approaches are self-supervised approaches which take unlabeled data but models the problem more closely to a supervised approach using automated self-defined pseudo labels to act as supervision. This approach has become extremely popular in NLP with famous language models such as BERT (Devlin et al., 2018) and GPT (Brown et al., 2020) variants all using the self-supervised paradigm to pre-train using self-supervised learning for state of the art results in their respective areas.

Outside of NLP, there has also been work in the computer vision area with the most popular approach to self-supervised CV learning being contrastive learning. In this project I hope to explore contrastive learning and the result of approaching the problem with the constraint of using efficient architectures which can be run on the edge on lower end hardware.

2. Method

2.1. Contrastive Learning

This study replicates the work from "A Simple Framework for Contrastive Learning of Visual Representations" by (Chen et al., 2020). The method applies the contrastive learning framework called SimCLR which aims to learn representations in an embedding space by maximizing similarity among a pair of augmented data points, while minimizing similarity for the augmented data points compared to all other data points in the batch.

As outlined in the SimCLR paper (Chen et al., 2020), the approach relies on four components. One is a stochastic data augmentation component which transforms an input into an augmented version to generate positive pairs which should

[1]Department of Systems Design Engineering, University of Waterloo, ON, Canada. Correspondence to: Cooper Ang <ck2ang@uwaterloo.ca>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

be embedded closely together. In the Data Augmentation section of the paper, I will go into a bit more depth about why this is a case and the specifics of the implementation. The second component is the base encoder, which is a neural network that creates a feature representation space. As mentioned later in the Architecture section of the paper, here I use variants of MobileNetV3-Large and MobileNetV3-Small (Howard et al., 2019). In addition to the base architecture a projection head is used to map the representation to an embedding space to compare augmented data points in. The last component is the contrastive loss function which adjusts the weights and biases to pull together known same examples and pull apart other examples.

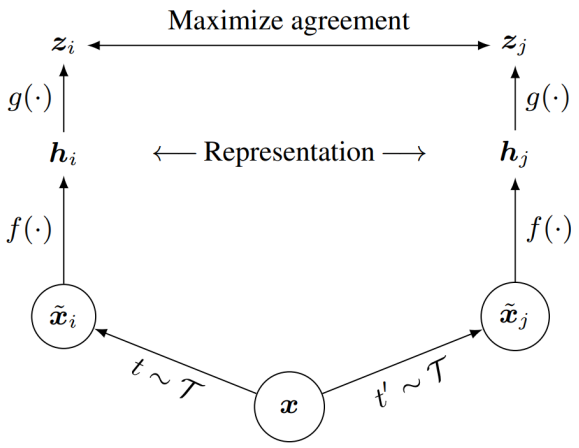

Figure 1. A diagram showing the contrastive learning approach for learning visual representations. From a sample x, two data augmentation operations are applied to t and t' to obtain two representations which are correlated. $f(\cdot)$ is applied yielding h which are representations for downstream tasks, and a further projection head $g(\cdot)$ is used in training to maximize agreement among the embedding z_i and z_j. Figure sourced from (Chen et al., 2020).

A review of the algorithm and loss found in the (Chen et al., 2020) are included below:

The contrastive loss function is not novel and has been used in (Sohn, 2016; Oord et al., 2018; Wu et al., 2018; Chen et al., 2020). The loss is defined by the similarity between two vectors. Let

$$\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\| \tag{1}$$

Then the loss function for a pair of examples (i, j) is:

$$\ell_{i,j} = -\log \frac{\exp\left(\text{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp\left(\text{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_k\right)/\tau\right)} \tag{2}$$

The loss is computed across all positive pairs in a minibatch. τ denotes a temperature parameter. The learning algorithm to train the model is followed exactly from the SimCLR paper (Chen et al., 2020), the training algorithm used from the paper is copied here for convenience of reference. No changes have been made to the training process. Algorithm 1 summarizes the constrastive learning approach.

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of $f, g, \mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

2.2. Research Questions

As there is no change to the learning algorithm compared to (Chen et al., 2020) implementation, the main questions that I hope to answer in this paper are the following research questions:

RQ1: How much of a performance degradation do we see with decreased model size for self-supervised contrastive learning?

RQ2: Do trends in self-supervised contrastive learning hold in the pretext of smaller model architectures, namely that a larger batch size and longer training lead to better performance?

2.3. Large Batch Size

Results from (Chen et al., 2020) show that larger batch sizes usually result in better results. The motivation is to verify to see if this result is consistent with smaller MobileNetV3 architectures that use depth wise separable convolution and squeeze and excitation blocks (Howard et al., 2019). For MobileNetV3-Small batch sizes of 1024 and 512 were trained and evaluated for 1000 epochs. For MobileNetV3-

Large batch sizes of 1024, 512, and 256 were trained and evaluated for 1000 epochs. Resnet-18 and Resnet-50 were trained using batch sizes of 512.

2.4. Data Augmentation

The contrastive learning approach relies on using data augmentation to create pairs from each sample for the model to train with. The self-supervised predictive task allows the model to learn from unlabeled data. It is thus important that the data augmentation step preserves the underlying meaning of the data in order to create a pair of samples which would to a human evaluator be the same object. In (Chen et al., 2020) it was found that contrastive learning works best with a composition of data augmentation operations, and generally strong augmentations help learn good representations. In this implementation random resized crop, random horizontal flip, random color jitter, random grayscale, and normalization were used. Compared to the original paper (Chen et al., 2020) no gaussian blur is used.

2.5. Dataset

The CIFAR-10 Dataset is used for the experimentation. The CIFAR dataset contains 60,000 color images in 10 classes with 6,000 images per class. 50,000 images were used as the training set and 10,000 images were used as the test set. Labels were not provided during training and only given at test time to determine test accuracy. Classes include airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. In order to satisfy the chosen architecture of MobileNetV3 the images had to be resized to 36x36 pixels instead of the original 32x32 size. Images were also standardized centering around the mean and divided by their standard deviation. CIFAR-10 was chosen as ImageNet would have taken too long to train for the limitations of this study.

2.6. Architecture

The architecture chosen was a slightly augmented version of the MobileNetV3-Small and MobileNetV3-Large architectures from (Howard et al., 2019). The MobileNetV3 architecture is a convolutional neural network focused on CV tasks that aims to combine techniques and novel architecture design to tune towards fast inference time on mobile phone CPUs (Howard et al., 2019). MobileNetV3 builds off the previous work of MobileNetV1 which introduced the novel at the time depthwise separable convolution, which "separates spatial filtering from the feature generation mechanism" (Howard et al., 2019). MobileNetV3 also was tuned with hardware-aware network architecture search (NAS) complemented by the NetAdapt algorithm.

Some highlights of the architecture are the use of swish non-linearities (Howard et al., 2019) instead of the normally used ReLU operations. Second is the use of squeeze and excitation blocks which adaptively recalibrates channel-wise feature responses by explicitly modeling interdependencies between channels (Hu et al., 2017). The MobileNetV3 architecture showed state of the art results for the model size and latency on supervised learning CV tasks such as ImageNet, or Cityscapes segmentation.

Input	Operator	exp size	$\#out$	SE	NL	s
$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$1^2 \times 576$	conv2d 1x1, NBN	-	1024	-	HS	1
$1^2 \times 1024$	conv2d 1x1, NBN	-	k	-	-	1

Figure 2. Specification for MobileNetV3-Small from (Howard et al., 2019). SE denotes whether there is a Squeeze-And-Excite in that block. NL denotes the type of nonlinearity used. Here, HS denotes h-swish and RE denotes ReLU. NBN denotes no batch normalization. s denotes stride.

Compared to previous work done in the area which used the Resnet-50 architecture (He et al., 2015) the MobileNetV3 architecture has much fewer parameters and is optimized for faster inference on mobile devices. In order to change the architecture for the use of contrastive learning, the classification head had to be changed from the original classification layer to a nonlinear classification head. This head is a simply defined as: $g(h) = W^{(2)}\sigma\left(W^{(1)}h\right)$. Compared to the original paper which used a ReLU activation with batch normalization, I changed the head to be consistent with the MobileNetV3 architecture to instead use the hard swish activation and 0.2 Dropout with no batch normalization. If time allowed I would have liked to explore the differences in the projection head architectures and how impacts the end performance on the smaller architectures.

The model was trained using augmented versions of MobileNetV3-Small and MobileNetV3-Large for multiple batch sizes. The Adam optimizer (Kingma & Ba, 2014) was used with a learning rate of 0.001. The feature dimension of the embedding space for the contrastive learning approach is 128D. Training was done on a single Nvidia 3060 Ti. Training took around 36 hours per model. As a baseline ResNet-50 and ResNet-18, with a batch size of 512 were also trained and evaluated. To keep things consistent,

Table 1. CIFAR-10 Contrastive Learning Results. KNN denotes the KNN evaluation protocol used during training, the other accuracy follows the method's linear evaluation protocol freezing weights in the model and using the output embedding from $f(\cdot)$ to train a linear classifier. All models were trained with 1000 epochs and used an embedding feature head size of 128.

BASE MODEL	BATCH SIZE	PARAM	TOP-5 (KNN)	TOP-1 (KNN)	TOP-5	TOP-1
MOBILENETV3 SMALL	1024	2.4 M	97.1	67.4	97.5	68.8
MOBILENETV3 LARGE	1024	4.1 M	98.3	76.4	98.5	78.2
MOBILENETV3 LARGE	512	4.1 M	98.3	78.3	98.8	79.6
MOBILENETV3 LARGE	256	4.1 M	98.4	77.4	98.8	79.2
RESNET-18	512	11.4 M	99.5	87.0	99.7	89.2
RESNET-50	512	24 M	99.6	89.1	99.9	92.0

the projection heads shared across the models were kept the same, only alternating the number of input neurons in the first layer to ensure that the matrix multiplies would be valid.

2.7. Evaluation Protocol

During training the models were tested using the weighted KNN algorithm with a K-value of 200. In this procedure the test set is put into the embedding space, before the projection head, and weighted KNN is applied to determine the corresponding model predictions. The results of this process are found in Table 1. Following the paper and the current standard, the linear evaluation protocol was used, which freezes the weights of the model, throws out the projection head, and adds a linear layer which is trained using supervised learning. The intuition behind this approach is that this protocol will determine how linear separable the representation space the model has learned. A standard 100 epochs was used to train the linear layer for evaluation. Batch size for training the linear layer was kept at 512 across each model candidate.

3. Results

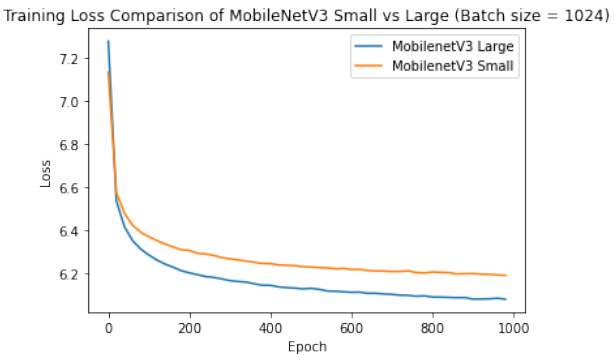

Figure 3. Training curves of MobileNetV3-Small vs MobilenetV3-Large with 1024 batch size trained using SimCLR

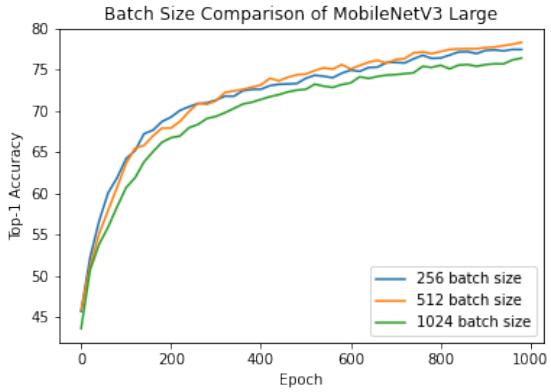

Figure 4. Training curves of MobileNetV3-Small vs MobilenetV3-Large with 1024 batch size trained using SimCLR

3.1. Analysis

As expected, It seems like a smaller model does degrade performance by a large amount resulting in a decrease from baseline 89.1% top-1 accuracy of the ResNet50 architecture (Chen et al., 2020) compared to the best MobileNetV3 variant only reaching 78.3% top-1 accuracy. Although the results show that contrastive learning is able to reach 78.3% top-1 accuracy without any labels, which is impressive nonetheless. A around 10% degradation in performance would be acceptable in some circumstances using a fifth of the parameters. MobileNetV3 architectures are also optimized to perform well on mobile hardware leading to much faster inference time. Depthwise separable convolutions, which MobileNetV3 uses result in less computations compared to traditional convolutions leading to further inference time savings.

In terms of the overall trends as seen in 1, the increase from 2.4 M parameters with MobileNetV3 Small to 4.1 M parameters with MobileNetV3 Large does see a sizable increase in model performance, confirming the results in (Chen et al., 2020), and showing that a deeper/larger model with more parameters generally performs better than a shallower network on CV tasks, especially the case with contrastive learning.

This is in line with results in supervised learning CV problems which generally show that larger models achieve better results when compared with smaller models. Comparing the loss curves of of MobileNetV3-Small and MobileNetV3-Large in Figure 3 you see that the larger model reaches a plateau later than the smaller model, demonstrating and solidifying the result that the larger model with more parameters is able to better learn representations with the contrastive learning algorithm. Further testing with ResNet18 which has around 11.4 M parameters showed slightly lower performance than ResNet50 which has around double the number of parameters (24.0 M). This might show that there is a limit to how much improvement we can see with larger model sizes and there may be an optimal approach depending on the constraints and tradeoffs in model size, accuracy, and latency. Overall the trend holds that a smaller model decreases performance by a considerable amount, thus answering RQ1.

In terms of batch size, as seen in Figure 3, interestingly decreasing the batch size from 1024 to either 256 or 512 resulted in better performance, which counters the results in (Chen et al., 2020). In (Chen et al., 2020) they were able to see performance improvements going all the way up to batch sizes of 4096 and 8192 (Chen et al., 2020). Although it is important to note the confounding factors such as a change of the optimizer from the LARs optimizer (You et al., 2017) used in SimCLR (Chen et al., 2020) to the Adam optimizer could be affecting the performance. There were also some smaller changes like not using a learning rate scheduler, and not using gaussian blur which could be impacting the results compared to (Chen et al., 2020). Comparing the batch sizes of 512 and 256 there is not a large change although it seems like 512 is slightly better. The difference compared to the paper which used a smaller model size could also be a factor in the empirical results although it is difficult to draw any theoretical conclusions. Empirically it seems like the trends in changes in batch size (Chen et al., 2020) do not hold up when dealing with MobileNetV3 variants which are smaller models compared to the ResNet50 model used in (Chen et al., 2020). Overall this leads to the conclusion that for smaller models, batch size choice is more nuanced and likely smaller, and would require a hyperparameter search, as a result answering RQ2. Although, compared to supervised learning approaches the best batch sizes for self-supervised contrastive learning do seem to be overall higher, with supervised learning generally staying around the range of 32, 64, 128.

3.2. Limitations

The limitation of the analysis is that there was limited time and compute budget in order to rigorously test different hyperparameter and architecture options. The slow training (around 36 hours) hindered agility to design better experimentation to do a full analysis. The study would be more complete comparing similar model sizes using the more efficient architecture of MobileNetV3 compared a less efficient but similar number of parameter ResNet models.

4. Related Work

4.1. Self-supervised techniques

Self-supervised learning approaches focus on creating labels from unlabeled data in an automated way (Kolesnikov et al., 2019). Other self-supervised approaches which have shown good results are rotation (Gidaris et al., 2018), where the task is provided with an image that is augmented with a rotation orientation of 0, 90, 180, or 270 degrees the task is to predict the rotation which was applied. Another is the Jigsaw proposed by (Noroozi & Favaro, 2016), where the task is to recover the relative spatial positions of an image which has been separated into 9 patches. All of these self-supervised techniques differ from the contrastive learning approach as its aim is to be able to allow the model to understand that an augmented image from the same class is the same, whereas these other pre-training tasks focus more on an innate understanding of the images.

4.2. Pre-train, fine-tune paradigm

The pre-train fine-tune paradigm has become widely used in the deep learning community for achieving SOTA results for supervised learning tasks. The idea behind the pre-train fine-tune paradigm is to use a large amount of unlabeled data to pre-train a model and then use a smaller amount of labeled data to fine-tune the model. This approach has been found to be useful for a variety of tasks such as image classification, object detection, and natural language processing. This approach has been shown to be especially effective when the labeled data is limited as the model can learn features from the unlabeled data which can be used as a starting point for the fine-tuning stage. This allows the model to make better use of the limited labeled data and achieve better performance. The work with contrastive learning can be extended as a pre-training task for later fine tuning to hopefully achieve better end results. Results have been demonstrated in using contrastive learning for NLP applications as a pre-training task for language modeling applications by (Gunel et al., 2020). (Chen et al., 2020) also briefly explored the capabilities of pre-train, fine-tune for contrastive self supervised learning, leading to some SOTA results for ImageNet provided with fewer labels than its competitors.

4.3. Faster inference methods

Outside of directly training smaller architectures there has been work done in other areas to speed up inference time by

starting with a larger model and then trying to transform it into a smaller model for use in inference workloads. One of the most popular being distillation methods originally proposed by (Hinton et al., 2015) where you can compress the knowledge of a larger model to a smaller one by training the model in a teacher-student relationship. The key to this process is using the "soft labels" usually from the softmax function of the teacher to allow the student to learn more information about the input that is just being provided by the ground truth label alone.

Another approach is the use of quantization methods. Quantization deals with the numerical representation of the weights and biases, and aims to reduce memory requirements and inference time by decreasing the precision of the float or integer representation (Banner et al., 2018). For example an approach can be to train a model using floating point 32 (full precision) and then changing it to int 8 for inference saving on memory and compute. Some issues with the approach of quantization is that there can sometimes be a large dropoff in model performance and tuning processes that may be expensive to undertake.

Another popular approach to faster inference is pruning a model's weights after training. Pruning is the process of systematically removing weights from the neural network. The process of pruning has its own field of study but some approaches analyze the gradients to systematically choose which parameters to delete, and some allow the model to fine-tune again once the weights are deleted to allow the model to compensate for any accuracy losses. As this paper focuses on the computer vision (CV) domain, there have been some results showing that by removing whole filters in a ConvNet you can reduce computational cost without hurting accuracy too much. (Li et al., 2016) was able to reduce inference costs of ResNet-110 while retaining close to the original model's accuracy on CIFAR-10.

5. Conclusion

In this work, I present the analysis of using a smaller MobileNetV3 architecture and training it using the SimCLR contrastive learning algorithm on CIFAR-10. I carefully study the impact of the smaller model architecture on model performance and answer questions about trends found by (Chen et al., 2020) holding in the domain of MobileNetV3 architectures. There is still work to be done in determining the best approach to find compressed models for edge inference trained using self-supervised contrastive learning.

6. Acknowledgements

The base code implementation of training loop and test was provided by Hao Ren's unofficial PyTorch implementation of the SimCLR codebase. The model architectures have been augmented from the originals to use the MobileNetV3 and ResNet-18 architectures. Details can be found at https://github.com/leftthomas/SimCLR. This work is also heavily credited to the work that was completed by the SimCLR by (Chen et al., 2020), this paper is only an extension of their work. I'd also like to thank Professor Ali Ayub and the Hayden Gunraj for the guidance on the project.

References

Banner, R., Nahshan, Y., Hoffer, E., and Soudry, D. Post-training 4-bit quantization of convolution networks for rapid-deployment, 2018. URL https://arxiv.org/abs/1810.05723.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. 2020. doi: 10.48550/ARXIV.2002.05709. URL https://arxiv.org/abs/2002.05709.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL https://arxiv.org/abs/1810.04805.

Gidaris, S., Singh, P., and Komodakis, N. Unsupervised representation learning by predicting image rotations. 2018. doi: 10.48550/ARXIV.1803.07728. URL https://arxiv.org/abs/1803.07728.

Gunel, B., Du, J., Conneau, A., and Stoyanov, V. Supervised contrastive learning for pre-trained language model fine-tuning. 2020. doi: 10.48550/ARXIV.2011.01403. URL https://arxiv.org/abs/2011.01403.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. 2015. doi: 10.48550/ARXIV.1503.02531. URL https://arxiv.org/abs/1503.02531.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V.,

Le, Q. V., and Adam, H. Searching for mobilenetv3. 2019. doi: 10.48550/ARXIV.1905.02244. URL https://arxiv.org/abs/1905.02244.

Hu, J., Shen, L., Albanie, S., Sun, G., and Wu, E. Squeeze-and-excitation networks, 2017. URL https://arxiv.org/abs/1709.01507.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980.

Kolesnikov, A., Zhai, X., and Beyer, L. Revisiting self-supervised visual representation learning. 2019. doi: 10.48550/ARXIV.1901.09005. URL https://arxiv.org/abs/1901.09005.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. 2016. doi: 10.48550/ARXIV.1608.08710. URL https://arxiv.org/abs/1608.08710.

Noroozi, M. and Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles, 2016. URL https://arxiv.org/abs/1603.09246.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding, 2018. URL https://arxiv.org/abs/1807.03748.

Sohn, K. Improved deep metric learning with multi-class n-pair loss objective. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf.

Wu, Z., Xiong, Y., Yu, S., and Lin, D. Unsupervised feature learning via non-parametric instance-level discrimination, 2018. URL https://arxiv.org/abs/1805.01978.

You, Y., Gitman, I., and Ginsburg, B. Large batch training of convolutional networks, 2017. URL https://arxiv.org/abs/1708.03888.