Investigating the Role of the Reduction Ratio in Squeeze-and-Excitation Networks

Henrik Holm
henholm@kth.se

Niklas Lindqvist
nlindqv@kth.se

Isak Persson
isakpe@kth.se

Abstract

Squeeze-and-Excitation Networks (SENets) introduce a building block for convolutional neural networks that improves channel interdependencies at little computational cost. With the addition of Squeeze-and-Excitation (SE) blocks, a new hyperparameter is introduced: the reduction ratio (RR), which determines the capacity and computational cost of an SE block. Previous research in the field has typically employed a constant RR for all SE blocks in a given SENet. The aim of this study is therefore to investigate whether using *different* RRs for different stages of a SENet can result in increases in performance. By performing a greedy search for optimal RRs on a SENet with 52 layers trained using the CIFAR-10 dataset, it is shown that a (8, 2, 16)-scheme for the RRs outperforms all schemes of constant RRs. In our experiments, this gave a relative increase in accuracy of 0.4%. All models and code are available at `https://github.com/nlindqv/RRs-in-SENets`.

***Keywords*—** Squeeze-and-Excitation, Convolutional Neural Networks, Image Representations, Reduction Ratio.

1 Introduction

Deep convolutional neural networks have over the recent years made big breakthroughs in several computer vision tasks, especially in image classification [8, 9, 10, 15]. The convolutional layers use filters to identify features based on spatial and channel-wise information. By stacking multiple convolutional layers, networks naturally integrate both low-, mid- and high-level features of images. In 2015, ResNet [1] was proposed as a solution to the exploding/vanishing gradient problem within deep neural networks and made it possible to successfully make the networks deeper than before, which also became state-of-the-art for several computer vision tasks at the time. Since then, many new network architectures have been proposed, one of them being Squeeze-and-Excitation Networks (SENets) [2] with the Squeeze-and-Excitation (SE) block which can be applied to any existing network architecture.

With the addition of SE blocks, a new hyperparameter is introduced: the reduction ratio (RR). The RR allows us to vary the capacity and computational cost of the SE blocks in an SENet. Smaller RRs increase the number of parameters of the model and vice versa. Increased complexity does not, however, correspond to monotonically increasing performance, as shown by previous studies [2]. It is therefore of interest to investigate further the exact role of the RR, especially as previous research in the area has typically utilized a constant RR throughout the whole network.

In this paper, we investigate the effects on the performance of allowing the RR to vary for different stages of a network as opposed to using the same constant RR for all stages. This is achieved by performing a greedy search, in which different RRs are tested for one stage at a time to find an optimal RR for that particular stage. The experiments are conducted using the CIFAR-10 dataset, explained in more detail in section 4. Parameters considered in our comparisons of different models include achieved test accuracy, loss on test set, and number of parameters. Our findings suggest that tuning the RR individually through the different stages, thus having non-constant RR throughout a SENet can outperform networks employing constant RRs through all stages.

2 Related Work

Architectures such as VGGNets [12] and Inception models [13] have shown that increasing depth can improve performance in image classification tasks. Batch normalization [5] complements these deep networks by improving the gradient propagation through solving the problem of covariate shift which arises when a network grows deeper. Further contributions in the area have allowed the depth of networks to increase further through the addition of skip connections between layers. ResNets [1] make use of identity-based skip connections where a layer's output is fed into the input of a deeper layer, as shown in Fig. 1. On ImageNet, this architecture allowed for an 8 times deeper network than VGGNets while still having lower complexity.

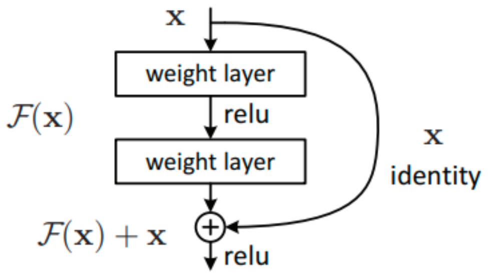

Figure 1: Illustration of a residual module where x is the input and $F(x) + x$ is the output [1].

Before Squeeze-and-Excitation Networks [2], many approaches for improving deep networks were focused on enhancing spatial encoding. SENets instead put focus into *channel relationships* by adaptively rescaling each channel based on the interdependencies between channels. The interdependencies are found through a neural network. Firstly, the spatial information of each channel is *squeezed* into a descriptor through global average pooling. The channels' descriptors are then inputted into a bottleneck network with two fully connected layers where the outputs are used for rescaling the channels. The number of nodes in the hidden layer of the bottleneck network is decided by a constant called the reduction ratio (RR). The number of nodes is equal to the number of input nodes divided by the RR. This module can easily be stacked onto previous state-of-the-art networks and show significant improvements with only slight increases in computational cost.

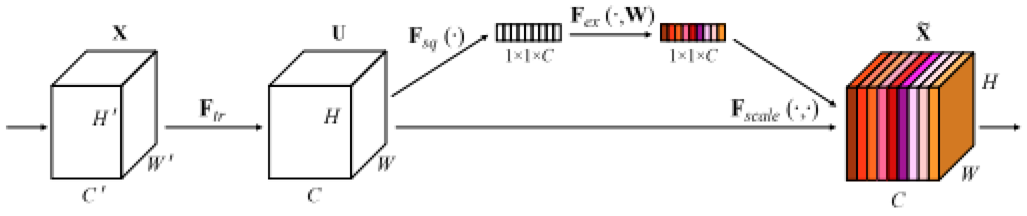

Figure 2: A Squeeze-and-Excitation block [2].

3 Purpose

While previous research in the area has proven the usefulness of SENets, little work has been done to investigate the exact role of the reduction ratio (RR). Primary contributions in the field have typically utilized constant RRs throughout the whole network, e.g. an identical RR of 16 for all stages of the network [3]. Based on this gap in the research, this paper sets out to investigate the effects of using different RRs in different stages of the network. By varying the RR for different stages, it is possible that performance can be improved, as different layers of the network perform different roles.

4 Dataset

For our investigations, a range of experiments are conducted on the CIFAR-10 dataset, which consists of 60K 32x32 colored images divided evenly into 10 classes [7]. 50K of the images are used for training, whilst the remaining 10K are set aside for testing. The CIFAR-10 dataset is a standard, well-established dataset within image classification benchmarks and is a more lightweight alternative to the more popular ImageNet dataset, thus the CIFAR-10 dataset suits projects with limited computational resources.

Current state-of-the-art methods such as Big Transfer (BiT) [6] and GPipe [4] score accuracies of 99.3% and 99% respectively on CIFAR-10. However, both these networks utilize transfer learning to achieve such state-of-the-art accuracies. Despite some of the best-performing networks relying on transfer learning, there also exist a plethora of methods which does not and still achieve accuracies of up to 99%, for example TResNet (99.0%) [11], EfficientNet-B7 (98.9%) [14] and EnAET (98.01%) [16] to mention a few. The aim of this paper, however, is not to achieve state-of-the-art performance. Rather, it is to improve the performance of a SENet by more thorough hyperparameter tuning.

As CIFAR-10 is an established and widely-used dataset, it is not in need of any particular preprocessing. However, following standard practices, data augmentation consisting of random horizontal flips, and random spatial shifts were performed for each training batch. Both data augmentation methods were implemented using built-in TensorFlow functions. For the spatial shift, a width shift and height shift range of 0.1 were used.

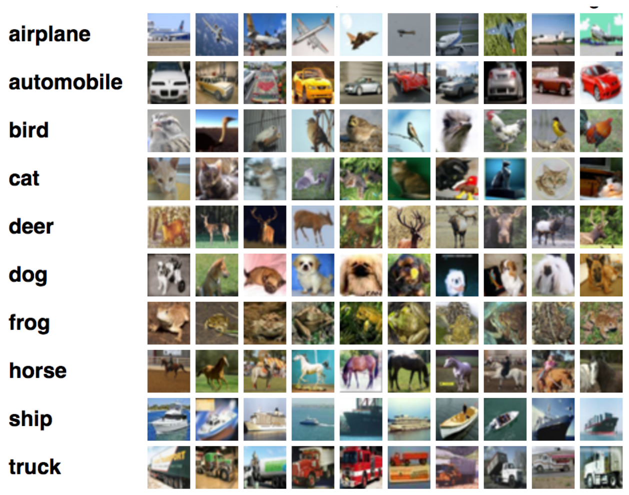

Figure 3: Example images of the CIFAR-10 dataset [7].

5 Approach

All experiments were conducted using Python 3.7 and TensorFlow 2.1.0. For our study, we based our models on ResNet as in [1] as well as SE-ResNet from [2] where the SE block is introduced. To validate the correctness of our implementations, the test accuracy of our ResNet-52 model was compared to that of the ResNet-56 from the original paper [1]. Our ResNet-52 model achieved a test accuracy of 90.97%, which compared to the 93.03% test accuracy of the original ResNet-56 [1] corresponds to a relative error of 2.26%. For our SE-ResNet-52 model with a constant reduction ratio, the accuracy achieved after 200 epochs was 91.17%, suggesting that our SE-modules were implemented correctly as it outperformed our vanilla ResNet-52 model.

5.1 Architecture

The implemented ResNet-52 and SE-ResNet-52 models are based on the paper *Deep Residual Learning for Image Recognition* [1]. They consist of a convolutional layer followed by three stages, each of which contains 8 residual modules stacked after each other, and ends with a fully connected layer. After each convolutional layer in the model, batch normalization is added before a ReLu activation function. The residual modules for the respective models are shown in Figure 4 and the details of the networks' parameters are displayed in Table 1.

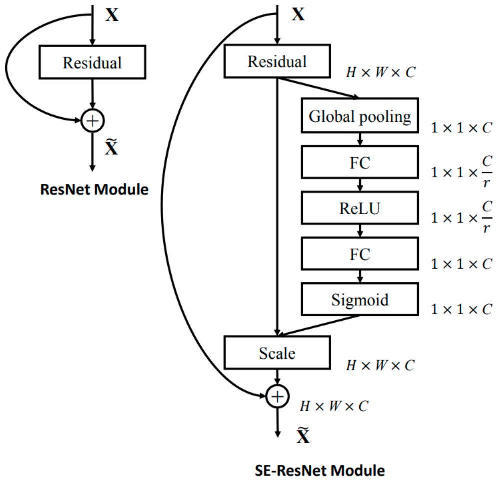

Figure 4: The schema of the original ResNet module (left) and SE-ResNet module (right) [2]. The "Residual" block represents two equally sized convolutional layers as in Figure 1.

Table 1: Operations with specific parameter settings and output size. The output size is presented as Height $\times$ Width $\times$ Channels. Residual blocks are listed inside the brackets and the number of stacked blocks in a stage is presented outside. The inner bracket shown for SE-ResNet-52 following *fc* represents the output dimensions of the two fully connected layers in the SE-module where r_i is the reduction ratio for stage *i*.

Name	Output size	ResNet-52	SE-ResNet-52
Initial Conv Layer	$32 \times 32 \times 16$	conv, 3×3, 16, stride 1	
Stage 1	$32 \times 32 \times 16$	$\begin{bmatrix} conv, 3 \times 3, 16 \\ conv, 3 \times 3, 16 \end{bmatrix} \times 8$	$\begin{bmatrix} conv, 3 \times 3, 16 \\ conv, 3 \times 3, 16 \\ fc, [16/r_1, 16] \end{bmatrix} \times 8$
Stage 2's 1st Shortcut Connection	$16 \times 16 \times 32$	conv, 1×1, 32, stride 2	
Stage 2	$16 \times 16 \times 32$	$\begin{bmatrix} conv, 3 \times 3, 32 \\ conv, 3 \times 3, 32 \end{bmatrix} \times 8$	$\begin{bmatrix} conv, 3 \times 3, 32 \\ conv, 3 \times 3, 32 \\ fc, [32/r_2, 32] \end{bmatrix} \times 8$
Stage 3's 1st Shortcut Connection	$8 \times 8 \times 64$	conv, 1×1, 64, stride 2	
Stage 3	$8 \times 8 \times 64$	$\begin{bmatrix} conv, 3 \times 3, 64 \\ conv, 3 \times 3, 64 \end{bmatrix} \times 8$	$\begin{bmatrix} conv, 3 \times 3, 64 \\ conv, 3 \times 3, 64 \\ fc, [64/r_3, 64] \end{bmatrix} \times 8$
Classification Layer	$1 \times 1 \times 10$	global average pool, 64-d *fc*, softmax	

5.2 Training

He normal initialization was used for initializing the weights. The cross-entropy loss was used with an Adam optimizer to update the network weights during training. As for hyperparameters, an L2 regularization term of 0.0001, a batch size of 32, and an initial learning rate of 0.001 were used. The learning rate was gradually reduced as training progressed. The learning rates used for different epochs in the training are displayed in Table 2.

Table 2: Epochs and their corresponding learning rates.

Epochs	Learning Rate
0-80	1e-3
81-120	1e-4
121-160	1e-5
161-180	1e-6
181-200	0.5e-6

6 Experiments

To attain a benchmark model, we first trained four SE-ResNet-52 models with constant reduction ratios (RRs) of 16, 8, 4, and 2 respectively. Out of these four networks, the highest-performing model was selected as our benchmark. As can be seen in Table 3, the highest-performing model was the one trained with a constant RR of 8 throughout all stages of the network.

Following this step, the effects of varying the RR throughout different stages of the network were investigated. This was done by performing a greedy search. Due to limited computational resources and time-wise constraints, a more rigorous exploration of how varying the RR affects performance could not be conducted.

The search focused first on the first stage of the network. Letting the second and third stages of the network employ the constant RR of 8 obtained from our benchmark model, the RR of the first stage was now allowed to vary. RR values of 8, 4, and 2 were explored for the first stage. Out of the three values tested, the RR of 8 yielded the highest test accuracy, meaning that the RR for the first stage of the network was now set to a constant value of 8 for later experiments.

In the next phase of our greedy search, the RR of the second stage of the network was now allowed to vary, while the RRs for stages 1 and 3 of the network were kept constant. An RR of 2 for the second stage of the network yielded the highest test accuracy.

For the last phase of the greedy search, the RRs for stages 1 and 2 of the network were kept at a constant 8 and 2 respectively, as given by the results of the two previous phases. Letting the RR of the third stage of the network take on values of 32, 16, 4, and 2 respectively, a final, best-performing model employing an (8, 2, 16)-scheme for the RRs was identified. This model outperformed our benchmark (8, 8, 8) model, suggesting that non-constant RRs throughout SENets can result in models achieving better performance than SENets employing constant RRs throughout all stages of the network.

7 Effects of Varying the Reduction Ratios

The findings of our experiments suggest that utilizing different RRs in different stages of a SENet can indeed result in increases in performance. As for the reasons behind different optimal RRs being identified for the different stages of the network, one can argue that this relates to the distinct roles performed by different layers. It is, for example, possible that the more general shallow layers work best with lower degrees of complexity (a higher RR), as a higher number of parameters (a lower RR) could lead to overfitting. In other words, for identifying more general low-level features, a higher propensity to overfitting is undesirable. This is suggested from the results of phase 1 of our greedy search, in which an RR of 8 in the first stage of the network outperformed both an RR of 4 and of 2.

However, this theory is not as evident when looking at the results for the last stage of our network as the optimal RR for the third stage (as found by our greedy search) turned out to be 16, as opposed to any of the lower values tested. Even though one could assume the optimal RR for the last stage to be 4 or 2, if the current complexity already sufficiently models the task, lowering the RR further would only result in overfitting on the training data.

Adding to this discourse, one should note the possibility of the optimal RR of the last stage being different, had the optimal RR for the second stage not been 2 (which resulted in a maximum number of parameters for the second stage). This RR of the second stage introduced a comparatively high

Table 3: Results of the greedy search for reduction ratios (RRs). The first column shows the vanilla ResNet-52 model. The second column shows the results of searching for the best constant reduction ratio. The last 3 columns show the results from searching for the best combination of RRs. The numbers inside the brackets of the models' names represent the RR for the respective stage of the network. For example, (8, 4, 2) would mean stage one has a RR of 8, stage 2 has a RR of 4, and stage 3 has a RR of 2.

Experiment	Model	Test Acc	Test Loss
	ResNet-52	90.97%	0.4803
Constant Reduction Ratio	SE-ResNet-52 (2, 2, 2)	90.86%	0.4826
	SE-ResNet-52 (4, 4, 4)	91.12%	0.4660
	SE-ResNet-52 (8, 8, 8)	**91.17%**	**0.4669**
	SE-ResNet-52 (16, 16, 16)	91.06%	0.4516
Stage 1 Greedy RR Search	SE-ResNet-52 (2, 8, 8)	90.78%	0.4547
	SE-ResNet-52 (4, 8, 8)	90.78%	0.4724
	SE-ResNet-52 (8, 8, 8)	**91.17%**	**0.4669**
Stage 2 Greedy RR Search	**SE-ResNet-52 (8, 2, 8)**	**91.31%**	**0.4578**
	SE-ResNet-52 (8, 4, 8)	91.18%	0.4554
	SE-ResNet-52 (8, 8, 8)	91.17%	0.4669
	SE-ResNet-52 (8, 16, 8)	91.02%	0.4855
Stage 3 Greedy RR Search	SE-ResNet-52 (8, 2, 2)	91.39%	0.4661
	SE-ResNet-52 (8, 2, 4)	90.88%	0.5006
	SE-ResNet-52 (8, 2, 8)	91.31%	0.4578
	SE-ResNet-52 (8, 2, 16)	**91.53%**	**0.4652**
	SE-ResNet-52 (8, 2, 32)	91.22%	0.4625

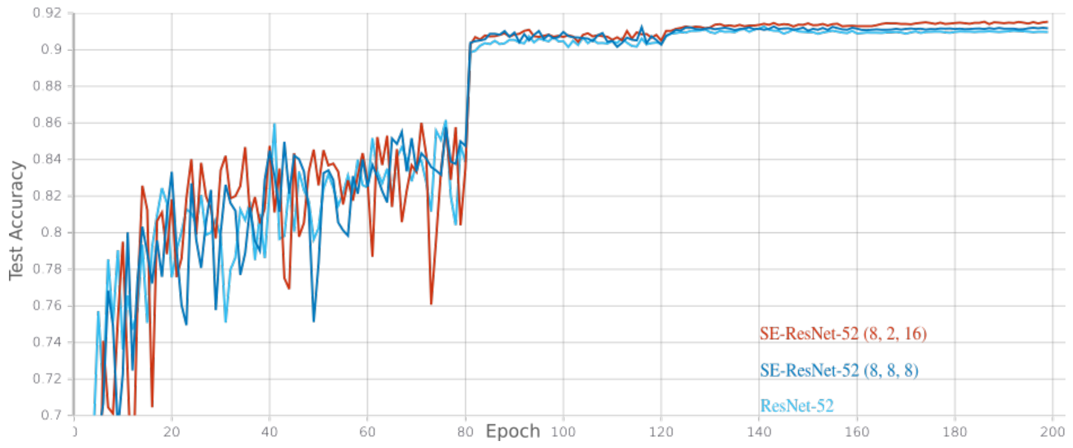

Figure 5: Training plot of test accuracy for each epoch for the models; ResNet-52 (light-blue), SE-ResNet-52 (8, 8, 8) (dark-blue) and SE-ResNet-52 (8, 2, 16) (Orange).

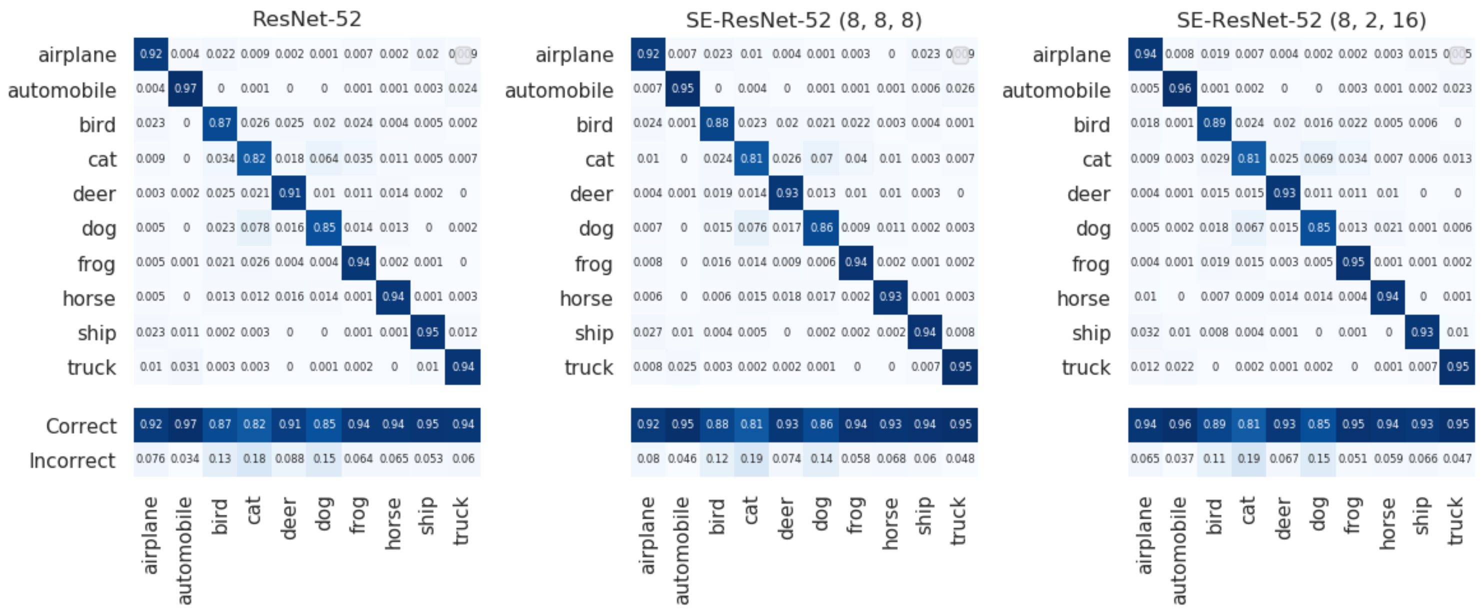

Figure 6: Confusion matrix of ResNet-52 (top-left), SE-ResNet-52 (8, 8, 8) (top-middle) and SE-ResNet-52 (8, 2, 16) (top-right). Below are the models' respective accuracy for each class.

degree of complexity, thus possibly reducing the need for the following stage (the last stage) to also contain a high number of parameters. With a different search method, it is likely that results would have been different. A disadvantage of the greedy search employed is its shortsightedness, which considers only one stage at a time in a consecutive order (hence "*greedy* search"). This means that the further the greedy search progresses, the stronger the influence of earlier results will be. We leave for future research the question as to whether the optimal RR scheme would have differed, had a more intelligent search method been employed.

8 Conclusions

In this paper, we investigated the impact of the RR hyperparameter introduced by the addition of SE blocks. It was shown that a constant RR through all SE blocks in a network does not always yield optimal performance. A greedy search for optimal RRs on the SE-ResNet-52 of our experiments showed that the (8, 2, 16)-scheme outperformed all schemes of constant RRs.

The (8, 2, 16)-scheme results in a higher degree of complexity than that of the (16, 16, 16) benchmark model. It is however known that SE blocks are still comparatively lightweight and impose only slight increases in computational burden [2]. One can, therefore, argue that the benefits of improving our SENet by employing non-constant RRs outweigh the drawbacks.

As for future research, a more rigorous search can be conducted. In our work, we were constrained by limited computational resources and by the relatively short time span of our study. Another promising avenue for future studies is to investigate whether improvements can only be achieved at the cost of higher degrees of complexity. In our case, the (8, 2, 16)-scheme identified by our greedy search results in a higher number of parameters than the (16, 16, 16)-scheme of our benchmark model. It is however not certain whether this is always the case.

References

[1] 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] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[3] Yang Hu, Guihua Wen, Mingnan Luo, Dan Dai, Jiajiong Ma, and Zhiwen Yu. Competitive inner-imaging squeeze and excitation for residual network. *arXiv preprint arXiv:1807.08920*, 2018.

[4] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2018.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[6] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning, 2019.

[7] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[9] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[11] Tal Ridnik, Hussam Lawen, Asaf Noy, and Itamar Friedman. Tresnet: High performance gpu-dedicated architecture, 2020.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] 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.

[14] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2019.

[15] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. *CoRR*, abs/1312.4659, 2013.

[16] Xiao Wang, Daisuke Kihara, Jiebo Luo, and Guo-Jun Qi. Enaet: Self-trained ensemble autoencoding transformations for semi-supervised learning, 2019.