Squeeze-and-Excitation Networks

Jie Hu[0000−0002−5150−1003] Li Shen[0000−0002−2283−4976] Samuel Albanie[0000−0001−9736−5134]
Gang Sun[0000−0001−6913−6799] Enhua Wu[0000−0002−2174−1428]

Abstract—The central building block of convolutional neural networks (CNNs) is the convolution operator, which enables networks to construct informative features by fusing both spatial and channel-wise information within local receptive fields at each layer. A broad range of prior research has investigated the spatial component of this relationship, seeking to strengthen the representational power of a CNN by enhancing the quality of spatial encodings throughout its feature hierarchy. In this work, we focus instead on the channel relationship and propose a novel architectural unit, which we term the "Squeeze-and-Excitation" (SE) block, that adaptively recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels. We show that these blocks can be stacked together to form SENet architectures that generalise extremely effectively across different datasets. We further demonstrate that SE blocks bring significant improvements in performance for existing state-of-the-art CNNs at slight additional computational cost. Squeeze-and-Excitation Networks formed the foundation of our ILSVRC 2017 classification submission which won first place and reduced the top-5 error to 2.251%, surpassing the winning entry of 2016 by a relative improvement of $\sim 25\%$. Models and code are available at https://github.com/hujie-frank/SENet.

Index Terms—Squeeze-and-Excitation, Image representations, Attention, Convolutional Neural Networks.

1 Introduction

Convolutional neural networks (CNNs) have proven to be useful models for tackling a wide range of visual tasks [1], [2], [3], [4]. At each convolutional layer in the network, a collection of filters expresses neighbourhood spatial connectivity patterns along input channels—fusing spatial and channel-wise information together within local receptive fields. By interleaving a series of convolutional layers with non-linear activation functions and downsampling operators, CNNs are able to produce image representations that capture hierarchical patterns and attain global theoretical receptive fields. A central theme of computer vision research is the search for more powerful representations that capture only those properties of an image that are most salient for a given task, enabling improved performance. As a widely-used family of models for vision tasks, the development of new neural network architecture designs now represents a key frontier in this search. Recent research has shown that the representations produced by CNNs can be strengthened by integrating learning mechanisms into the network that help capture spatial correlations between features. One such approach, popularised by the Inception family of architectures [5], [6], incorporates multi-scale processes into network modules to achieve improved performance. Further work has sought to better model spatial dependencies [7], [8] and incorporate spatial attention into the structure of the network [9].

In this paper, we investigate a different aspect of network design - the relationship between channels. We introduce a new architectural unit, which we term the *Squeeze-and-Excitation* (SE) block, with the goal of improving the quality of representations produced by a network by explicitly modelling the interdependencies between the channels of its convolutional features. To this end, we propose a mechanism that allows the network to perform feature recalibration, through which it can learn to use global information to selectively emphasise informative features and suppress less useful ones.

The structure of the SE building block is depicted in Fig. 1. For any given transformation $\mathbf{F}_{tr}$ mapping the input $\mathbf{X}$ to the feature maps $\mathbf{U}$ where $\mathbf{U} \in \mathbb{R}^{H \times W \times C}$, e.g. a convolution, we can construct a corresponding SE block to perform feature recalibration. The features $\mathbf{U}$ are first passed through a *squeeze* operation, which produces a channel descriptor by aggregating feature maps across their spatial dimensions ($H \times W$). The function of this descriptor is to produce an embedding of the global distribution of channel-wise feature responses, allowing information from the global receptive field of the network to be used by all its layers. The aggregation is followed by an *excitation* operation, which takes the form of a simple self-gating mechanism that takes the embedding as input and produces a collection of per-channel modulation weights. These weights are applied to the feature maps $\mathbf{U}$ to generate the output of the SE block which can be fed directly into subsequent layers of the network.

It is possible to construct an SE network (SENet) by simply stacking a collection of SE blocks. Moreover, these SE blocks can also be used as a drop-in replacement for the original block at a range of depths in the network architec-

- *Jie Hu and Enhua Wu are with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, 100190, China.*
  They are also with the University of Chinese Academy of Sciences, Beijing, 100049, China.
  Jie Hu is also with Momenta and Enhua Wu is also with the Faculty of Science and Technology & AI Center at University of Macau.
  E-mail: hujie@ios.ac.cn ehwu@umac.mo
- *Gang Sun is with LIAMA-NLPR at the Institute of Automation, Chinese Academy of Sciences. He is also with Momenta.*
  E-mail: sungang@momenta.ai
- *Li Shen and Samuel Albanie are with the Visual Geometry Group at the University of Oxford.*
  E-mail: {lishen,albanie}@robots.ox.ac.uk

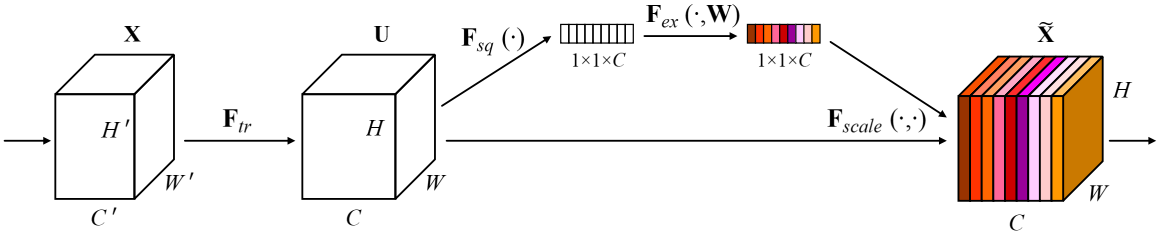


Fig. 1. A Squeeze-and-Excitation block.

ture (Section 6.4). While the template for the building block is generic, the role it performs at different depths differs throughout the network. In earlier layers, it excites informative features in a class-agnostic manner, strengthening the shared low-level representations. In later layers, the SE blocks become increasingly specialised, and respond to different inputs in a highly class-specific manner (Section 7.2). As a consequence, the benefits of the feature recalibration performed by SE blocks can be accumulated through the network.

The design and development of new CNN architectures is a difficult engineering task, typically requiring the selection of many new hyperparameters and layer configurations. By contrast, the structure of the SE block is simple and can be used directly in existing state-of-the-art architectures by replacing components with their SE counterparts, where the performance can be effectively enhanced. SE blocks are also computationally lightweight and impose only a slight increase in model complexity and computational burden.

To provide evidence for these claims, we develop several SENets and conduct an extensive evaluation on the ImageNet dataset [10]. We also present results beyond ImageNet that indicate that the benefits of our approach are not restricted to a specific dataset or task. By making use of SENets, we ranked first in the ILSVRC 2017 classification competition. Our best model ensemble achieves a 2.251% top-5 error on the test set[1]. This represents roughly a 25% relative improvement when compared to the winner entry of the previous year (top-5 error of 2.991%).

2 Related Work

Deeper architectures. VGGNets [11] and Inception models [5] showed that increasing the depth of a network could significantly increase the quality of representations that it was capable of learning. By regulating the distribution of the inputs to each layer, Batch Normalization (BN) [6] added stability to the learning process in deep networks and produced smoother optimisation surfaces [12]. Building on these works, ResNets demonstrated that it was possible to learn considerably deeper and stronger networks through the use of identity-based skip connections [13], [14]. Highway networks [15] introduced a gating mechanism to regulate the flow of information along shortcut connections. Following these works, there have been further reformulations of the connections between network layers [16], [17], which show promising improvements to the learning and representational properties of deep networks.

An alternative, but closely related line of research has focused on methods to improve the functional form of the computational elements contained within a network. Grouped convolutions have proven to be a popular approach for increasing the cardinality of learned transformations [18], [19]. More flexible compositions of operators can be achieved with multi-branch convolutions [5], [6], [20], [21], which can be viewed as a natural extension of the grouping operator. In prior work, cross-channel correlations are typically mapped as new combinations of features, either independently of spatial structure [22], [23] or jointly by using standard convolutional filters [24] with 1×1 convolutions. Much of this research has concentrated on the objective of reducing model and computational complexity, reflecting an assumption that channel relationships can be formulated as a composition of instance-agnostic functions with local receptive fields. In contrast, we claim that providing the unit with a mechanism to explicitly model dynamic, non-linear dependencies between channels using global information can ease the learning process, and significantly enhance the representational power of the network.

Algorithmic Architecture Search. Alongside the works described above, there is also a rich history of research that aims to forgo manual architecture design and instead seeks to learn the structure of the network automatically. Much of the early work in this domain was conducted in the neuro-evolution community, which established methods for searching across network topologies with evolutionary methods [25], [26]. While often computationally demanding, evolutionary search has had notable successes which include finding good memory cells for sequence models [27], [28] and learning sophisticated architectures for large-scale image classification [29], [30], [31]. With the goal of reducing the computational burden of these methods, efficient alternatives to this approach have been proposed based on Lamarckian inheritance [32] and differentiable architecture search [33].

By formulating architecture search as hyperparameter optimisation, random search [34] and other more sophisticated model-based optimisation techniques [35], [36] can also be used to tackle the problem. Topology selection as a path through a fabric of possible designs [37] and direct architecture prediction [38], [39] have been proposed as additional viable architecture search tools. Particularly strong results have been achieved with techniques from reinforcement learning [40], [41], [42], [43], [44]. SE blocks

1. http://image-net.org/challenges/LSVRC/2017/results

can be used as atomic building blocks for these search algorithms, and were demonstrated to be highly effective in this capacity in concurrent work [45].

Attention and gating mechanisms. Attention can be interpreted as a means of biasing the allocation of available computational resources towards the most informative components of a signal [46], [47], [48], [49], [50], [51]. Attention mechanisms have demonstrated their utility across many tasks including sequence learning [52], [53], localisation and understanding in images [9], [54], image captioning [55], [56] and lip reading [57]. In these applications, it can be incorporated as an operator following one or more layers representing higher-level abstractions for adaptation between modalities. Some works provide interesting studies into the combined use of spatial and channel attention [58], [59]. Wang et al. [58] introduced a powerful trunk-and-mask attention mechanism based on hourglass modules [8] that is inserted between the intermediate stages of deep residual networks. By contrast, our proposed SE block comprises a lightweight gating mechanism which focuses on enhancing the representational power of the network by modelling channel-wise relationships in a computationally efficient manner.

3 Squeeze-and-Excitation Blocks

A Squeeze-and-Excitation block is a computational unit which can be built upon a transformation $\mathbf{F}_{tr}$ mapping an input $\mathbf{X} \in \mathbb{R}^{H' \times W' \times C'}$ to feature maps $\mathbf{U} \in \mathbb{R}^{H \times W \times C}$. In the notation that follows we take $\mathbf{F}_{tr}$ to be a convolutional operator and use $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_C]$ to denote the learned set of filter kernels, where $\mathbf{v}_c$ refers to the parameters of the c-th filter. We can then write the outputs as $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_C]$, where

$$\mathbf{u}_c = \mathbf{v}_c * \mathbf{X} = \sum_{s=1}^{C'} \mathbf{v}_c^s * \mathbf{x}^s. \tag{1}$$

Here $*$ denotes convolution, $\mathbf{v}_c = [\mathbf{v}_c^1, \mathbf{v}_c^2, \ldots, \mathbf{v}_c^{C'}]$, $\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{C'}]$ and $\mathbf{u}_c \in \mathbb{R}^{H \times W}$. $\mathbf{v}_c^s$ is a 2D spatial kernel representing a single channel of $\mathbf{v}_c$ that acts on the corresponding channel of $\mathbf{X}$. To simplify the notation, bias terms are omitted. Since the output is produced by a summation through all channels, channel dependencies are implicitly embedded in $\mathbf{v}_c$, but are entangled with the local spatial correlation captured by the filters. The channel relationships modelled by convolution are inherently implicit and local (except the ones at top-most layers). We expect the learning of convolutional features to be enhanced by explicitly modelling channel interdependencies, so that the network is able to increase its sensitivity to informative features which can be exploited by subsequent transformations. Consequently, we would like to provide it with access to global information and recalibrate filter responses in two steps, *squeeze* and *excitation*, before they are fed into the next transformation. A diagram illustrating the structure of an SE block is shown in Fig. 1.

3.1 Squeeze: Global Information Embedding

In order to tackle the issue of exploiting channel dependencies, we first consider the signal to each channel in the output features. Each of the learned filters operates with a local receptive field and consequently each unit of the transformation output $\mathbf{U}$ is unable to exploit contextual information outside of this region.

To mitigate this problem, we propose to *squeeze* global spatial information into a channel descriptor. This is achieved by using global average pooling to generate channel-wise statistics. Formally, a statistic $\mathbf{z} \in \mathbb{R}^C$ is generated by shrinking $\mathbf{U}$ through its spatial dimensions $H \times W$, such that the c-th element of $\mathbf{z}$ is calculated by:

$$z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j). \tag{2}$$

Discussion. The output of the transformation $\mathbf{U}$ can be interpreted as a collection of the local descriptors whose statistics are expressive for the whole image. Exploiting such information is prevalent in prior feature engineering work [60], [61], [62]. We opt for the simplest aggregation technique, global average pooling, noting that more sophisticated strategies could be employed here as well.

3.2 Excitation: Adaptive Recalibration

To make use of the information aggregated in the *squeeze* operation, we follow it with a second operation which aims to fully capture channel-wise dependencies. To fulfil this objective, the function must meet two criteria: first, it must be flexible (in particular, it must be capable of learning a nonlinear interaction between channels) and second, it must learn a non-mutually-exclusive relationship since we would like to ensure that multiple channels are allowed to be emphasised (rather than enforcing a one-hot activation). To meet these criteria, we opt to employ a simple gating mechanism with a sigmoid activation:

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z})), \tag{3}$$

where δ refers to the ReLU [63] function, $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}$. To limit model complexity and aid generalisation, we parameterise the gating mechanism by forming a bottleneck with two fully-connected (FC) layers around the non-linearity, i.e. a dimensionality-reduction layer with reduction ratio r (this parameter choice is discussed in Section 6.1), a ReLU and then a dimensionality-increasing layer returning to the channel dimension of the transformation output $\mathbf{U}$. The final output of the block is obtained by rescaling $\mathbf{U}$ with the activations $\mathbf{s}$:

$$\widetilde{\mathbf{x}}_c = \mathbf{F}_{scale}(\mathbf{u}_c, s_c) = s_c \, \mathbf{u}_c, \tag{4}$$

where $\widetilde{\mathbf{X}} = [\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2, \ldots, \widetilde{\mathbf{x}}_C]$ and $\mathbf{F}_{scale}(\mathbf{u}_c, s_c)$ refers to channel-wise multiplication between the scalar s_c and the feature map $\mathbf{u}_c \in \mathbb{R}^{H \times W}$.

Discussion. The excitation operator maps the input-specific descriptor $\mathbf{z}$ to a set of channel weights. In this regard, SE blocks intrinsically introduce dynamics conditioned on the input, which can be regarded as a self-attention function on channels whose relationships are not confined to the local receptive field the convolutional filters are responsive to.

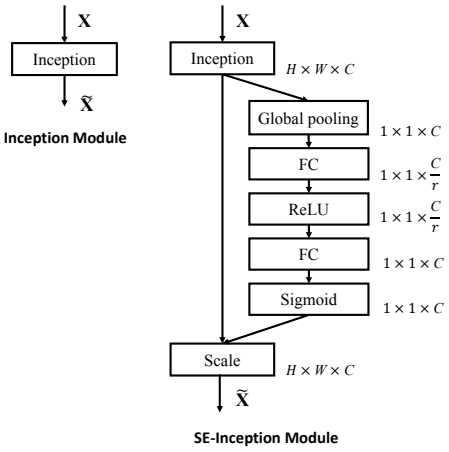


Fig. 2. The schema of the original Inception module (left) and the SE-Inception module (right).

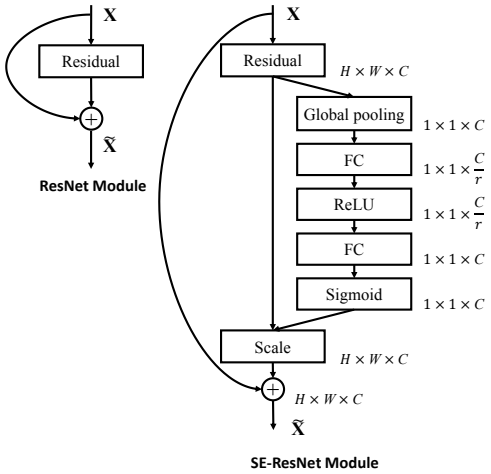


Fig. 3. The schema of the original Residual module (left) and the SE-ResNet module (right).

3.3 Instantiations

The SE block can be integrated into standard architectures such as VGGNet [11] by insertion after the non-linearity following each convolution. Moreover, the flexibility of the SE block means that it can be directly applied to transformations beyond standard convolutions. To illustrate this point, we develop SENets by incorporating SE blocks into several examples of more complex architectures, described next.

We first consider the construction of SE blocks for Inception networks [5]. Here, we simply take the transformation $\mathbf{F}_{tr}$ to be an entire Inception module (see Fig. 2) and by making this change for each such module in the architecture, we obtain an *SE-Inception* network. SE blocks can also be used directly with residual networks (Fig. 3 depicts the schema of an *SE-ResNet* module). Here, the SE block transformation $\mathbf{F}_{tr}$ is taken to be the non-identity branch of a residual module. *Squeeze* and *Excitation* both act before summation with the identity branch. Further variants that integrate SE blocks with ResNeXt [19], Inception-ResNet [21], MobileNet [64] and ShuffleNet [65] can be constructed by following similar schemes. For concrete examples of SENet architectures, a detailed description of SE-ResNet-50 and SE-ResNeXt-50 is given in Table 1.

One consequence of the flexible nature of the SE block is that there are several viable ways in which it could be integrated into these architectures. Therefore, to assess sensitivity to the integration strategy used to incorporate SE blocks into a network architecture, we also provide ablation experiments exploring different designs for block inclusion in Section 6.5.

4 Model and Computational Complexity

For the proposed SE block design to be of practical use, it must offer a good trade-off between improved performance and increased model complexity. To illustrate the computational burden associated with the module, we consider a comparison between ResNet-50 and SE-ResNet-50 as an example. ResNet-50 requires $\sim$3.86 GFLOPs in a single forward pass for a 224×224 pixel input image. Each SE block makes use of a global average pooling operation in the *squeeze* phase and two small FC layers in the *excitation* phase, followed by an inexpensive channel-wise scaling operation. In the aggregate, when setting the reduction ratio r (introduced in Section 3.2) to 16, SE-ResNet-50 requires $\sim$3.87 GFLOPs, corresponding to a 0.26% relative increase over the original ResNet-50. In exchange for this slight additional computational burden, the accuracy of SE-ResNet-50 surpasses that of ResNet-50 and indeed, approaches that of a deeper ResNet-101 network requiring $\sim$7.58 GFLOPs (Table 2).

In practical terms, a single pass forwards and backwards through ResNet-50 takes 190 ms, compared to 209 ms for SE-ResNet-50 with a training minibatch of 256 images (both timings are performed on a server with 8 NVIDIA Titan X GPUs). We suggest that this represents a reasonable runtime overhead, which may be further reduced as global pooling and small inner-product operations receive further optimisation in popular GPU libraries. Due to its importance for embedded device applications, we further benchmark CPU inference time for each model: for a 224×224 pixel input image, ResNet-50 takes 164 ms in comparison to 167 ms for SE-ResNet-50. We believe that the small additional computational cost incurred by the SE block is justified by its contribution to model performance.

We next consider the additional parameters introduced by the proposed SE block. These additional parameters result solely from the two FC layers of the gating mechanism and therefore constitute a small fraction of the total network capacity. Concretely, the total number introduced by the weight parameters of these FC layers is given by:

$$\frac{2}{r}\sum_{s=1}^{S} N_s \cdot {C_s}^2, \tag{5}$$

where r denotes the reduction ratio, S refers to the number of stages (a stage refers to the collection of blocks operating on feature maps of a common spatial dimension), C_s denotes the dimension of the output channels and N_s denotes the number of repeated blocks for stage s (when bias terms are used in FC layers, the introduced parameters and computational cost are typically negligible). SE-ResNet-50 introduces $\sim$2.5 million additional parameters beyond the

TABLE 1
(Left) ResNet-50 [13]. (Middle) SE-ResNet-50. (Right) SE-ResNeXt-50 with a 32×4d template. The shapes and operations with specific parameter settings of a residual building block are listed inside the brackets and the number of stacked blocks in a stage is presented outside. The inner brackets following by *fc* indicates the output dimension of the two fully connected layers in an SE module.

Output size	ResNet-50	SE-ResNet-50	SE-ResNeXt-50 (32×4d)
112×112	conv, 7×7, 64, stride 2		
56×56	max pool, 3×3, stride 2		
	$\begin{bmatrix} \text{conv}, 1 \times 1, 64 \\ \text{conv}, 3 \times 3, 64 \\ \text{conv}, 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1 \times 1, 64 \\ \text{conv}, 3 \times 3, 64 \\ \text{conv}, 1 \times 1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1 \times 1, 128 \\ \text{conv}, 3 \times 3, 128 \quad C = 32 \\ \text{conv}, 1 \times 1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$
28×28	$\begin{bmatrix} \text{conv}, 1 \times 1, 128 \\ \text{conv}, 3 \times 3, 128 \\ \text{conv}, 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} \text{conv}, 1 \times 1, 128 \\ \text{conv}, 3 \times 3, 128 \\ \text{conv}, 1 \times 1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$	$\begin{bmatrix} \text{conv}, 1 \times 1, 256 \\ \text{conv}, 3 \times 3, 256 \quad C = 32 \\ \text{conv}, 1 \times 1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$
14×14	$\begin{bmatrix} \text{conv}, 1 \times 1, 256 \\ \text{conv}, 3 \times 3, 256 \\ \text{conv}, 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv}, 1 \times 1, 256 \\ \text{conv}, 3 \times 3, 256 \\ \text{conv}, 1 \times 1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv}, 1 \times 1, 512 \\ \text{conv}, 3 \times 3, 512 \quad C = 32 \\ \text{conv}, 1 \times 1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 6$
7×7	$\begin{bmatrix} \text{conv}, 1 \times 1, 512 \\ \text{conv}, 3 \times 3, 512 \\ \text{conv}, 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1 \times 1, 512 \\ \text{conv}, 3 \times 3, 512 \\ \text{conv}, 1 \times 1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1 \times 1, 1024 \\ \text{conv}, 3 \times 3, 1024 \quad C = 32 \\ \text{conv}, 1 \times 1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$
1×1	global average pool, 1000-d fc, softmax		

TABLE 2
Single-crop error rates (%) on the ImageNet validation set and complexity comparisons. The *original* column refers to the results reported in the original papers (the results of ResNets are obtained from the website: https://github.com/Kaiminghe/deep-residual-networks). To enable a fair comparison, we re-train the baseline models and report the scores in the *re-implementation* column. The *SENet* column refers to the corresponding architectures in which SE blocks have been added. The numbers in brackets denote the performance improvement over the re-implemented baselines. † indicates that the model has been evaluated on the non-blacklisted subset of the validation set (this is discussed in more detail in [21]), which may slightly improve results. VGG-16 and SE-VGG-16 are trained with batch normalization.

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [13]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [13]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [13]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [19]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [19]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
VGG-16 [11]	-	-	27.02	8.81	15.47	$25.22_{(1.80)}$	$7.70_{(1.11)}$	15.48
BN-Inception [6]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [21]	19.9†	4.9†	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

$\sim$25 million parameters required by ResNet-50, corresponding to a $\sim 10\%$ increase. In practice, the majority of these parameters come from the final stage of the network, where the excitation operation is performed across the greatest number of channels. However, we found that this comparatively costly final stage of SE blocks could be removed at only a small cost in performance ($<0.1\%$ top-5 error on ImageNet) reducing the relative parameter increase to $\sim 4\%$, which may prove useful in cases where parameter usage is a key consideration (see Section 6.4 and 7.2 for further discussion).

5 Experiments

In this section, we conduct experiments to investigate the effectiveness of SE blocks across a range of tasks, datasets and model architectures.

5.1 Image Classification

To evaluate the influence of SE blocks, we first perform experiments on the ImageNet 2012 dataset [10] which comprises 1.28 million training images and 50K validation images from 1000 different classes. We train networks on the training set and report the top-1 and top-5 error on the validation set.

Each baseline network architecture and its corresponding SE counterpart are trained with identical optimisation schemes. We follow standard practices and perform data augmentation with random cropping using scale and aspect ratio [5] to a size of 224×224 pixels (or 299×299 for Inception-ResNet-v2 [21] and SE-Inception-ResNet-v2) and perform random horizontal flipping. Each input image is normalised through mean RGB-channel subtraction. All models are trained on our distributed learning system *ROCS* which is designed to handle efficient parallel training of large networks. Optimisation is performed using synchronous SGD with momentum 0.9 and a minibatch size of 1024. The initial learning rate is set to 0.6 and decreased by a factor of 10 every 30 epochs. Models are trained for 100 epochs from scratch, using the weight initialisation strategy described in [66]. The reduction ratio r (in Section 3.2) is set to 16 by default (except where stated otherwise).

When evaluating the models we apply centre-cropping so that 224×224 pixels are cropped from each image, after

TABLE 3
Single-crop error rates (%) on the ImageNet validation set and complexity comparisons. MobileNet refers to "1.0 MobileNet-224" in [64] and ShuffleNet refers to "ShuffleNet $1 \times (g = 3)$" in [65]. The numbers in brackets denote the performance improvement over the re-implementation.

	original		re-implementation				SENet			
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	MFLOPs	Params	top-1 err.	top-5 err.	MFLOPs	Params
MobileNet [64]	29.4	-	28.4	9.4	569	4.2M	$25.3_{(3.1)}$	$7.7_{(1.7)}$	572	4.7M
ShuffleNet [65]	32.6	-	32.6	12.5	140	1.8M	$31.0_{(1.6)}$	$11.1_{(1.4)}$	142	2.4M

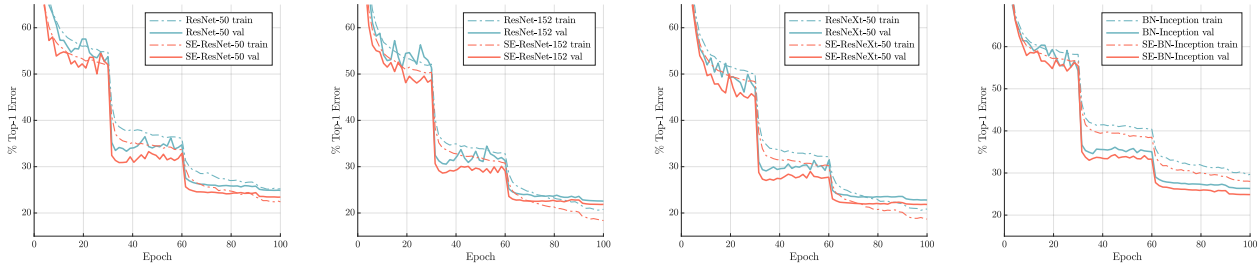


Fig. 4. Training baseline architectures and their SENet counterparts on ImageNet. SENets exhibit improved optimisation characteristics and produce consistent gains in performance which are sustained throughout the training process.

its shorter edge is first resized to 256 (299×299 from each image whose shorter edge is first resized to 352 for Inception-ResNet-v2 and SE-Inception-ResNet-v2).

Network depth. We begin by comparing SE-ResNet against ResNet architectures with different depths and report the results in Table 2. We observe that SE blocks consistently improve performance across different depths with an extremely small increase in computational complexity. Remarkably, SE-ResNet-50 achieves a single-crop top-5 validation error of 6.62%, exceeding ResNet-50 (7.48%) by 0.86% and approaching the performance achieved by the much deeper ResNet-101 network (6.52% top-5 error) with only half of the total computational burden (3.87 GFLOPs vs. 7.58 GFLOPs). This pattern is repeated at greater depth, where SE-ResNet-101 (6.07% top-5 error) not only matches, but outperforms the deeper ResNet-152 network (6.34% top-5 error) by 0.27%. While it should be noted that the SE blocks themselves add depth, they do so in an extremely computationally efficient manner and yield good returns even at the point at which extending the depth of the base architecture achieves diminishing returns. Moreover, we see that the gains are consistent across a range of different network depths, suggesting that the improvements induced by SE blocks may be complementary to those obtained by simply increasing the depth of the base architecture.

Integration with modern architectures. We next study the effect of integrating SE blocks with two further state-of-the-art architectures, Inception-ResNet-v2 [21] and ResNeXt (using the setting of 32×4d) [19], both of which introduce additional computational building blocks into the base network. We construct SENet equivalents of these networks, SE-Inception-ResNet-v2 and SE-ResNeXt (the configuration of SE-ResNeXt-50 is given in Table 1) and report results in Table 2. As with the previous experiments, we observe significant performance improvements induced by the introduction of SE blocks into both architectures. In particular, SE-ResNeXt-50 has a top-5 error of 5.49% which is superior to both its direct counterpart ResNeXt-50 (5.90% top-5 error) as well as the deeper ResNeXt-101 (5.57% top-5 error), a model which has almost twice the total number of parameters and computational overhead. We note a slight difference in performance between our re-implementation of Inception-ResNet-v2 and the result reported in [21]. However, we observe a similar trend with regard to the effect of SE blocks, finding that SE counterpart (4.79% top-5 error) outperforms our reimplemented Inception-ResNet-v2 baseline (5.21% top-5 error) by 0.42% as well as the reported result in [21].

We also assess the effect of SE blocks when operating on *non-residual* networks by conducting experiments with the VGG-16 [11] and BN-Inception architecture [6]. To facilitate the training of VGG-16 from scratch, we add Batch Normalization layers after each convolution. We use identical training schemes for both VGG-16 and SE-VGG-16. The results of the comparison are shown in Table 2. Similarly to the results reported for the residual baseline architectures, we observe that SE blocks bring improvements in performance on the non-residual settings.

To provide some insight into influence of SE blocks on the optimisation of these models, example training curves for runs of the baseline architectures and their respective SE counterparts are depicted in Fig. 4. We observe that SE blocks yield a steady improvement throughout the optimisation procedure. Moreover, this trend is fairly consistent across a range of network architectures considered as baselines.

Mobile setting. Finally, we consider two representative architectures from the class of mobile-optimised networks, MobileNet [64] and ShuffleNet [65]. For these experiments, we used a minibatch size of 256 and slightly less aggressive data augmentation and regularisation as in [65]. We trained the models across 8 GPUs using SGD with momentum (set to 0.9) and an initial learning rate of 0.1 which was reduced by a factor of 10 each time the validation loss plateaued. The total training process required ~ 400 epochs (enabling us to reproduce the baseline performance of [65]). The results reported in Table 3 show that SE blocks consistently improve

TABLE 4
Classification error (%) on CIFAR-10.

	original	SENet
ResNet-110 [14]	6.37	5.21
ResNet-164 [14]	5.46	4.39
WRN-16-8 [67]	4.27	3.88
Shake-Shake 26 2x96d [68] + Cutout [69]	2.56	2.12

TABLE 5
Classification error (%) on CIFAR-100.

	original	SENet
ResNet-110 [14]	26.88	23.85
ResNet-164 [14]	24.33	21.31
WRN-16-8 [67]	20.43	19.14
Shake-Even 29 2x4x64d [68] + Cutout [69]	15.85	15.41

TABLE 6
Single-crop error rates (%) on Places365 validation set.

	top-1 err.	top-5 err.
Places-365-CNN [72]	41.07	11.48
ResNet-152 (ours)	41.15	11.61
SE-ResNet-152	**40.37**	**11.01**

TABLE 7
Faster R-CNN object detection results (%) on COCO *minival* set.

	AP@IoU=0.5	AP
ResNet-50	57.9	38.0
SE-ResNet-50	61.0	40.4
ResNet-101	60.1	39.9
SE-ResNet-101	62.7	41.9

the accuracy by a large margin at a minimal increase in computational cost.

Additional datasets. We next investigate whether the benefits of SE blocks generalise to datasets beyond ImageNet. We perform experiments with several popular baseline architectures and techniques (ResNet-110 [14], ResNet-164 [14], WideResNet-16-8 [67], Shake-Shake [68] and Cutout [69]) on the CIFAR-10 and CIFAR-100 datasets [70]. These comprise a collection of 50k training and 10k test 32×32 pixel RGB images, labelled with 10 and 100 classes respectively. The integration of SE blocks into these networks follows the same approach that was described in Section 3.3. Each baseline and its SENet counterpart are trained with standard data augmentation strategies [24], [71]. During training, images are randomly horizontally flipped and zero-padded on each side with four pixels before taking a random 32×32 crop. Mean and standard deviation normalisation is also applied. The setting of the training hyperparameters (e.g. minibatch size, initial learning rate, weight decay) match those suggested by the original papers. We report the performance of each baseline and its SENet counterpart on CIFAR-10 in Table 4 and performance on CIFAR-100 in Table 5. We observe that in every comparison SENets outperform the baseline architectures, suggesting that the benefits of SE blocks are not confined to the ImageNet dataset.

5.2 Scene Classification

We also conduct experiments on the Places365-Challenge dataset [73] for scene classification. This dataset comprises 8 million training images and $36,500$ validation images across 365 categories. Relative to classification, the task of scene understanding offers an alternative assessment of a model's ability to generalise well and handle abstraction. This is because it often requires the model to handle more complex data associations and to be robust to a greater level of appearance variation.

We opted to use ResNet-152 as a strong baseline to assess the effectiveness of SE blocks and follow the training and evaluation protocols described in [72], [74]. In these experiments, models are trained from scratch. We report the results in Table 6, comparing also with prior work. We observe that SE-ResNet-152 (11.01% top-5 error) achieves a lower validation error than ResNet-152 (11.61% top-5 error), providing evidence that SE blocks can also yield improvements for scene classification. This SENet surpasses the previous state-of-the-art model Places-365-CNN [72] which has a top-5 error of 11.48% on this task.

5.3 Object Detection on COCO

We further assess the generalisation of SE blocks on the task of object detection using the COCO dataset [75]. As in previous work [19], we use the *minival* protocol, i.e., training the models on the union of the 80k training set and a 35k val subset and evaluating on the remaining 5k val subset. Weights are initialised by the parameters of the model trained on the ImageNet dataset. We use the Faster R-CNN [4] detection framework as the basis for evaluating our models and follow the hyperparameter setting described in [76] (i.e., end-to-end training with the '2x' learning schedule). Our goal is to evaluate the effect of replacing the trunk architecture (ResNet) in the object detector with SE-ResNet, so that any changes in performance can be attributed to better representations. Table 7 reports the validation set performance of the object detector using ResNet-50, ResNet-101 and their SE counterparts as trunk architectures. SE-ResNet-50 outperforms ResNet-50 by 2.4% (a relative 6.3% improvement) on COCO's standard AP metric and by 3.1% on AP@IoU=0.5. SE blocks also benefit the deeper ResNet-101 architecture achieving a 2.0% improvement (5.0% relative improvement) on the AP metric. In summary, this set of experiments demonstrate the generalisability of SE blocks. The induced improvements can be realised across a broad range of architectures, tasks and datasets.

5.4 ILSVRC 2017 Classification Competition

SENets formed the foundation of our submission to the ILSVRC competition where we achieved first place. Our winning entry comprised a small ensemble of SENets that employed a standard multi-scale and multi-crop fusion strategy to obtain a top-5 error of 2.251% on the test set. As part of this submission, we constructed an additional model, *SENet-154*, by integrating SE blocks with a modified ResNeXt [19] (the details of the architecture are provided in Appendix). We compare this model with prior work on the ImageNet validation set in Table 8 using standard crop

TABLE 8
Single-crop error rates (%) of state-of-the-art CNNs on ImageNet validation set with crop sizes 224×224 and 320×320 / 299×299.

	224×224		320×320 / 299×299	
	top-1 err.	top-5 err.	top-1 err.	top-5 err.
ResNet-152 [13]	23.0	6.7	21.3	5.5
ResNet-200 [14]	21.7	5.8	20.1	4.8
Inception-v3 [20]	-	-	21.2	5.6
Inception-v4 [21]	-	-	20.0	5.0
Inception-ResNet-v2 [21]	-	-	19.9	4.9
ResNeXt-101 (64×4d) [19]	20.4	5.3	19.1	4.4
DenseNet-264 [17]	22.15	6.12	-	-
Attention-92 [58]	-	-	19.5	4.8
PyramidNet-200 [77]	20.1	5.4	19.2	4.7
DPN-131 [16]	19.93	5.12	18.55	4.16
SENet-154	**18.68**	**4.47**	**17.28**	**3.79**

TABLE 9
Comparison (%) with state-of-the-art CNNs on ImageNet validation set using larger crop sizes/additional training data. †This model was trained with a crop size of 320×320.

	extra data	crop size	top-1 err.	top-5 err.
Very Deep PolyNet [78]	-	331	18.71	4.25
NASNet-A (6 @ 4032) [42]	-	331	17.3	3.8
PNASNet-5 (N=4,F=216) [35]	-	331	17.1	3.8
SENet-154†	-	320	16.88	3.58
AmoebaNet-C [79]	-	331	16.5	3.5
ResNeXt-101 32×48d [80]	✓	224	14.6	2.4

sizes (224×224 and 320×320). We observe that SENet-154 achieves a top-1 error of 18.68% and a top-5 error of 4.47% using a 224×224 centre crop evaluation, which represents the strongest reported result.

Following the challenge there has been a great deal of further progress on the ImageNet benchmark. For comparison, we include the strongest results that we are currently aware of in Table 9. The best performance using only ImageNet data was recently reported by [79]. This method uses reinforcement learning to develop new policies for data augmentation during training to improve the performance of the architecture searched by [31]. The best overall performance was reported by [80] using a ResNeXt-101 $32 \times 48d$ architecture. This was achieved by pretraining their model on approximately one billion weakly labelled images and finetuning on ImageNet. The improvements yielded by more sophisticated data augmentation [79] and extensive pretraining [80] may be complementary to our proposed changes to the network architecture.

6 Ablation Study

In this section we conduct ablation experiments to gain a better understanding of the effect of using different configurations on components of the SE blocks. All ablation experiments are performed on the ImageNet dataset on a single machine (with 8 GPUs). ResNet-50 is used as the backbone architecture. We found empirically that on ResNet architectures, removing the biases of the FC layers in the excitation operation facilitates the modelling of channel dependencies, and use this configuration in the following experiments. The data augmentation strategy follows the approach described in Section 5.1. To allow us to study the upper limit of performance for each variant, the learning rate is initialised to 0.1 and training continues until the validation loss plateaus[2] ($\sim$300 epochs in total). The learning rate is then reduced by a factor of 10 and then this process is repeated (three times in total). Label-smoothing regularisation [20] is used during training.

TABLE 10
Single-crop error rates (%) on ImageNet and parameter sizes for SE-ResNet-50 at different reduction ratios. Here, *original* refers to ResNet-50.

Ratio r	top-1 err.	top-5 err.	Params
2	22.29	6.00	45.7M
4	22.25	6.09	35.7M
8	22.26	5.99	30.7M
16	22.28	6.03	28.1M
32	22.72	6.20	26.9M
original	23.30	6.55	25.6M

6.1 Reduction ratio

The reduction ratio r introduced in Eqn. 5 is a hyperparameter which allows us to vary the capacity and computational cost of the SE blocks in the network. To investigate the trade-off between performance and computational cost mediated by this hyperparameter, we conduct experiments with SE-ResNet-50 for a range of different r values. The comparison in Table 10 shows that performance is robust to a range of reduction ratios. Increased complexity does not improve performance monotonically while a smaller ratio dramatically increases the parameter size of the model. Setting $r = 16$ achieves a good balance between accuracy and complexity. In practice, using an identical ratio throughout a network may not be optimal (due to the distinct roles performed by different layers), so further improvements may be achievable by tuning the ratios to meet the needs of a given base architecture.

6.2 Squeeze Operator

We examine the significance of using global average pooling as opposed to global max pooling as our choice of squeeze operator (since this worked well, we did not consider more sophisticated alternatives). The results are reported in Table 11. While both max and average pooling are effective, average pooling achieves slightly better performance, justifying its selection as the basis of the squeeze operation. However, we note that the performance of SE blocks is fairly robust to the choice of specific aggregation operator.

6.3 Excitation Operator

We next assess the choice of non-linearity for the excitation mechanism. We consider two further options: ReLU and tanh, and experiment with replacing the sigmoid with these

2. For reference, training with a 270 epoch fixed schedule (reducing the learning rate at 125, 200 and 250 epochs) achieves top-1 and top-5 error rates for ResNet-50 and SE-ResNet-50 of (23.21%, 6.53%) and (22.20%, 6.00%) respectively.

TABLE 11
Effect of using different squeeze operators in SE-ResNet-50 on ImageNet (error rates %).

Squeeze	top-1 err.	top-5 err.
Max	22.57	6.09
Avg	**22.28**	**6.03**

TABLE 12
Effect of using different non-linearities for the excitation operator in SE-ResNet-50 on ImageNet (error rates %).

Excitation	top-1 err.	top-5 err.
ReLU	23.47	6.98
Tanh	23.00	6.38
Sigmoid	**22.28**	**6.03**

alternative non-linearities. The results are reported in Table 12. We see that exchanging the sigmoid for tanh slightly worsens performance, while using ReLU is dramatically worse and in fact causes the performance of SE-ResNet-50 to drop below that of the ResNet-50 baseline. This suggests that for the SE block to be effective, careful construction of the excitation operator is important.

6.4 Different stages

We explore the influence of SE blocks at different stages by integrating SE blocks into ResNet-50, one stage at a time. Specifically, we add SE blocks to the intermediate stages: stage_2, stage_3 and stage_4, and report the results in Table 13. We observe that SE blocks bring performance benefits when introduced at each of these stages of the architecture. Moreover, the gains induced by SE blocks at different stages are complementary, in the sense that they can be combined effectively to further bolster network performance.

6.5 Integration strategy

Finally, we perform an ablation study to assess the influence of the location of the SE block when integrating it into existing architectures. In addition to the proposed SE design, we consider three variants: (1) SE-PRE block, in which the SE block is moved before the residual unit; (2) SE-POST block, in which the SE unit is moved after the summation with the identity branch (after ReLU) and (3) SE-Identity block, in which the SE unit is placed on the identity connection in parallel to the residual unit. These variants are illustrated in Figure 5 and the performance of each variant is reported in Table 14. We observe that the SE-PRE, SE-Identity and proposed SE block each perform similarly well, while usage of the SE-POST block leads to a drop in performance. This experiment suggests that the performance improvements produced by SE units are fairly robust to their location, provided that they are applied prior to branch aggregation.

TABLE 13
Effect of integrating SE blocks with ResNet-50 at different stages on ImageNet (error rates %).

Stage	top-1 err.	top-5 err.	GFLOPs	Params
ResNet-50	23.30	6.55	3.86	25.6M
SE_Stage_2	23.03	6.48	3.86	25.6M
SE_Stage_3	23.04	6.32	3.86	25.7M
SE_Stage_4	22.68	6.22	3.86	26.4M
SE_All	22.28	6.03	3.87	28.1M

TABLE 14
Effect of different SE block integration strategies with ResNet-50 on ImageNet (error rates %).

Design	top-1 err.	top-5 err.
SE	22.28	6.03
SE-PRE	22.23	6.00
SE-POST	22.78	6.35
SE-Identity	22.20	6.15

TABLE 15
Effect of integrating SE blocks at the 3x3 convolutional layer of each residual branch in ResNet-50 on ImageNet (error rates %).

Design	top-1 err.	top-5 err.	GFLOPs	Params
SE	22.28	6.03	3.87	28.1M
SE_3×3	22.48	6.02	3.86	25.8M

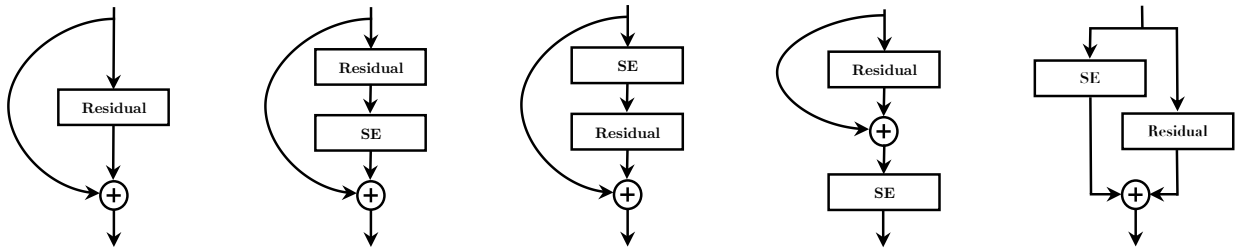
In the experiments above, each SE block was placed outside the structure of a residual unit. We also construct a variant of the design which moves the SE block inside the residual unit, placing it directly after the 3×3 convolutional layer. Since the 3×3 convolutional layer possesses fewer channels, the number of parameters introduced by the corresponding SE block is also reduced. The comparison in Table 15 shows that the SE_3×3 variant achieves comparable classification accuracy with fewer parameters than the standard SE block. Although it is beyond the scope of this work, we anticipate that further efficiency gains will be achievable by tailoring SE block usage for specific architectures.

7 Role of SE blocks

Although the proposed SE block has been shown to improve network performance on multiple visual tasks, we would also like to understand the relative importance of the squeeze operation and how the excitation mechanism operates in practice. A rigorous theoretical analysis of the representations learned by deep neural networks remains challenging, we therefore take an empirical approach to examining the role played by the SE block with the goal of attaining at least a primitive understanding of its practical function.

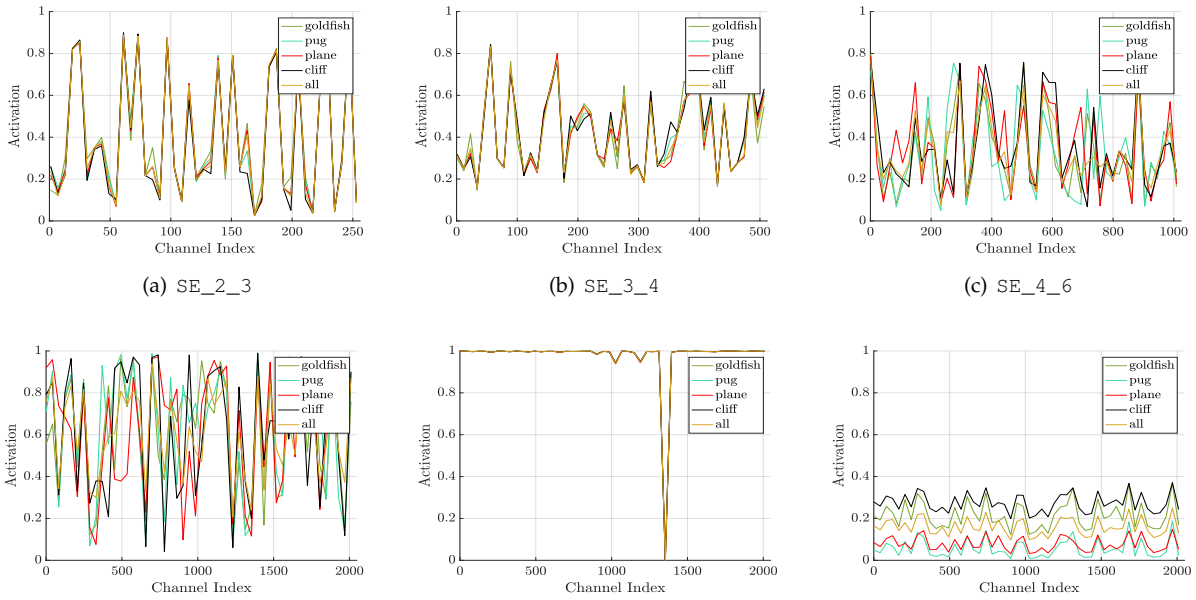
7.1 Effect of Squeeze

To assess whether the global embedding produced by the squeeze operation plays an important role in performance, we experiment with a variant of the SE block that adds an equal number of parameters, but does not perform global average pooling. Specifically, we remove the pooling operation and replace the two FC layers with corresponding 1×1 convolutions with identical channel dimensions in the excitation operator, namely *NoSqueeze*, where the excitation output maintains the spatial dimensions as input. In contrast to the SE block, these point-wise convolutions can only remap the channels as a function of the output of a local operator. While in practice, the later layers of a deep network will typically possess a (theoretical) global



(a) Residual block (b) Standard SE block (c) SE-PRE block (d) SE-POST block (e) SE-Identity block

Fig. 5. SE block integration designs explored in the ablation study.



(a) SE_2_3 (b) SE_3_4 (c) SE_4_6

(d) SE_5_1 (e) SE_5_2 (f) SE_5_3

Fig. 6. Activations induced by the Excitation operator at different depths in the SE-ResNet-50 on ImageNet. Each set of activations is named according to the following scheme: SE_stageID_blockID. With the exception of the unusual behaviour at SE_5_2, the activations become increasingly class-specific with increasing depth.

TABLE 16
Effect of Squeeze operator on ImageNet (error rates %).

	top-1 err.	top-5 err.	GFLOPs	Params
ResNet-50	23.30	6.55	3.86	25.6M
NoSqueeze	22.93	6.39	4.27	28.1M
SE	**22.28**	**6.03**	3.87	28.1M

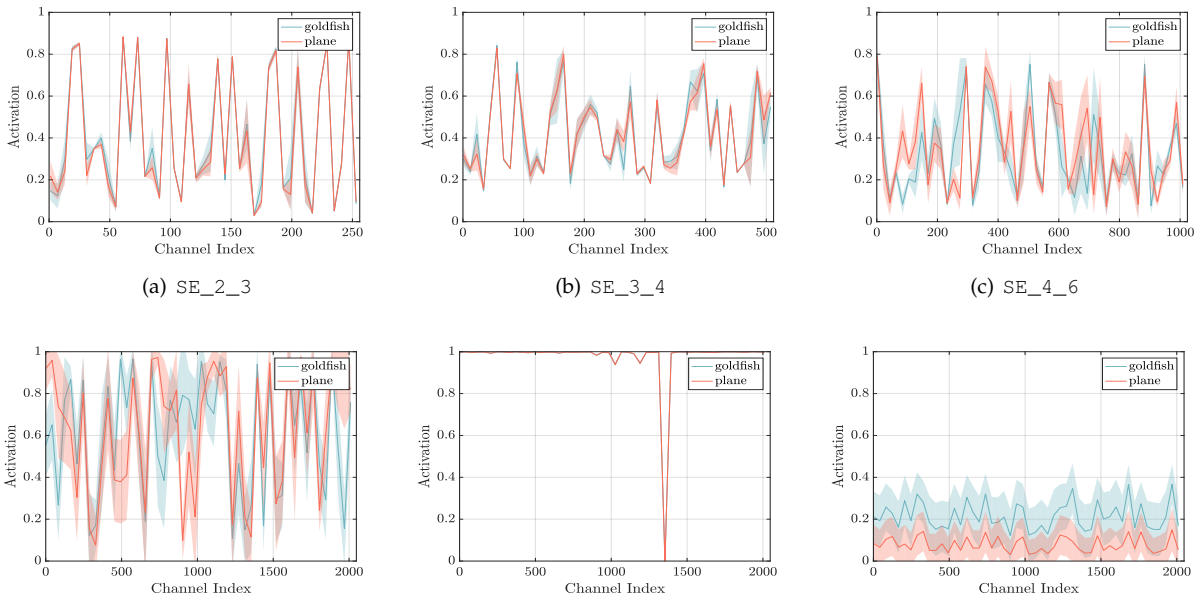
receptive field, global embeddings are no longer directly accessible throughout the network in the *NoSqueeze* variant. The accuracy and computational complexity of both models are compared to a standard ResNet-50 model in Table 16. We observe that the use of global information has a significant influence on the model performance, underlining the importance of the squeeze operation. Moreover, in comparison to the *NoSqueeze* design, the SE block allows this global information to be used in a computationally parsimonious manner.

7.2 Role of Excitation

To provide a clearer picture of the function of the excitation operator in SE blocks, in this section we study example activations from the SE-ResNet-50 model and examine their distribution with respect to different classes and different input images at various depths in the network. In particular, we would like to understand how excitations vary across images of different classes, and across images within a class.

We first consider the distribution of excitations for different classes. Specifically, we sample four classes from the ImageNet dataset that exhibit semantic and appearance diversity, namely *goldfish*, *pug*, *plane* and *cliff* (example images from these classes are shown in Appendix). We then draw fifty samples for each class from the validation set and compute the average activations for fifty uniformly sampled channels in the last SE block of each stage (immediately prior to downsampling) and plot their distribution in Fig. 6. For reference, we also plot the distribution of the mean activations across all of the 1000 classes.

We make the following three observations about the role of the *excitation* operation. First, the distribution across different classes is very similar at the earlier layers of the network, e.g. SE_2_3. This suggests that the importance of feature channels is likely to be shared by different classes in the early stages. The second observation is that at greater



(a) SE_2_3 (b) SE_3_4 (c) SE_4_6

(d) SE_5_1 (e) SE_5_2 (f) SE_5_3

Fig. 7. Activations induced by *Excitation* in the different modules of SE-ResNet-50 on image samples from the goldfish and plane classes of ImageNet. The module is named "SE_stageID_blockID".

depth, the value of each channel becomes much more class-specific as different classes exhibit different preferences to the discriminative value of features, e.g. SE_4_6 and SE_5_1. These observations are consistent with findings in previous work [81], [82], namely that earlier layer features are typically more general (e.g. class agnostic in the context of the classification task) while later layer features exhibit greater levels of specificity [83].

Next, we observe a somewhat different phenomena in the last stage of the network. SE_5_2 exhibits an interesting tendency towards a saturated state in which most of the activations are close to one. At the point at which all activations take the value one, an SE block reduces to the identity operator. At the end of the network in the SE_5_3 (which is immediately followed by global pooling prior before classifiers), a similar pattern emerges over different classes, up to a modest change in scale (which could be tuned by the classifiers). This suggests that SE_5_2 and SE_5_3 are less important than previous blocks in providing recalibration to the network. This finding is consistent with the result of the empirical investigation in Section 4 which demonstrated that the additional parameter count could be significantly reduced by removing the SE blocks for the last stage with only a marginal loss of performance.

Finally, we show the mean and standard deviations of the activations for image instances within the same class for two sample classes (*goldfish* and *plane*) in Fig. 7. We observe a trend consistent with the inter-class visualisation, indicating that the dynamic behaviour of SE blocks varies over both classes and instances within a class. Particularly in the later layers of the network where there is considerable diversity of representation within a single class, the network learns to take advantage of feature recalibration to improve its discriminative performance [84]. In summary, SE blocks produce instance-specific responses which nevertheless function to support the increasingly class-specific needs of the model at different layers in the architecture.

8 Conclusion

In this paper we proposed the SE block, an architectural unit designed to improve the representational power of a network by enabling it to perform dynamic channel-wise feature recalibration. A wide range of experiments show the effectiveness of SENets, which achieve state-of-the-art performance across multiple datasets and tasks. In addition, SE blocks shed some light on the inability of previous architectures to adequately model channel-wise feature dependencies. We hope this insight may prove useful for other tasks requiring strong discriminative features. Finally, the feature importance values produced by SE blocks may be of use for other tasks such as network pruning for model compression.

Acknowledgments

The authors would like to thank Chao Li and Guangyuan Wang from Momenta for their contributions in the training system optimisation and experiments on CIFAR dataset. We would also like to thank Andrew Zisserman, Aravindh Mahendran and Andrea Vedaldi for many helpful discussions. The work is supported in part by NSFC Grants (61632003, 61620106003, 61672502, 61571439), National Key R&D Program of China (2017YFB1002701), and Macao FDCT Grant (068/2015/A2). Samuel Albanie is supported by EPSRC AIMS CDT EP/L015897/1.

Appendix: Details of SENet-154

SENet-154 is constructed by incorporating SE blocks into a modified version of the 64×4d ResNeXt-152 which extends

(a) goldfish (b) pug (c) plane (d) cliff

Fig. 8. Sample images from the four classes of ImageNet used in the experiments described in Sec. 7.2.

the original ResNeXt-101 [19] by adopting the block stacking strategy of ResNet-152 [13]. Further differences to the design and training of this model (beyond the use of SE blocks) are as follows: (a) The number of the first 1×1 convolutional channels for each bottleneck building block was halved to reduce the computational cost of the model with a minimal decrease in performance. (b) The first 7×7 convolutional layer was replaced with three consecutive 3×3 convolutional layers. (c) The 1×1 down-sampling projection with stride-2 convolution was replaced with a 3×3 stride-2 convolution to preserve information. (d) A dropout layer (with a dropout ratio of 0.2) was inserted before the classification layer to reduce overfitting. (e) Label-smoothing regularisation (as introduced in [20]) was used during training. (f) The parameters of all BN layers were frozen for the last few training epochs to ensure consistency between training and testing. (g) Training was performed with 8 servers (64 GPUs) in parallel to enable large batch sizes (2048). The initial learning rate was set to 1.0.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Conference on Neural Information Processing Systems*, 2012.

[2] A. Toshev and C. Szegedy, "DeepPose: Human pose estimation via deep neural networks," in *CVPR*, 2014.

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Conference on Neural Information Processing Systems*, 2015.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[7] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *CVPR*, 2016.

[8] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016.

[9] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Conference on Neural Information Processing Systems*, 2015.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, 2015.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[12] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization? (no, it is not about internal covariate shift)," in *Conference on Neural Information Processing Systems*, 2018.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *ECCV*, 2016.

[15] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Conference on Neural Information Processing Systems*, 2015.

[16] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Conference on Neural Information Processing Systems*, 2017.

[17] G. Huang, Z. Liu, K. Q. Weinberger, and L. Maaten, "Densely connected convolutional networks," in *CVPR*, 2017.

[18] Y. Ioannou, D. Robertson, R. Cipolla, and A. Criminisi, "Deep roots: Improving CNN efficiency with hierarchical filter groups," in *CVPR*, 2017.

[19] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *CVPR*, 2017.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR*, 2016.

[21] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI Conference on Artificial Intelligence*, 2016.

[22] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *BMVC*, 2014.

[23] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017.

[24] M. Lin, Q. Chen, and S. Yan, "Network in network," in *ICLR*, 2014.

[25] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms." in *ICGA*, 1989.

[26] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, 2002.

[27] J. Bayer, D. Wierstra, J. Togelius, and J. Schmidhuber, "Evolving memory cell structures for sequence learning," in *ICANN*, 2009.

[28] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *ICML*, 2015.

[29] L. Xie and A. L. Yuille, "Genetic CNN," in *ICCV*, 2017.

[30] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *ICML*, 2017.

[31] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *arXiv preprint arXiv:1802.01548*, 2018.

[32] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," *arXiv preprint arXiv:1804.09081*, 2018.

[33] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[34] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *JMLR*, 2012.

[35] C. Liu, B. Zoph, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *ECCV*, 2018.

[36] R. Negrinho and G. Gordon, "Deeparchitect: Automatically designing and training deep architectures," *arXiv preprint arXiv:1704.08792*, 2017.

[37] S. Saxena and J. Verbeek, "Convolutional neural fabrics," in *Conference on Neural Information Processing Systems*, 2016.

[38] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "SMASH: one-shot model architecture search through hypernetworks," in *ICLR*, 2018.

[39] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," in *ICLR Workshop*, 2018.

[40] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *ICLR*, 2017.

[41] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.

[42] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, 2018.

[43] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *ICLR*, 2018.

[44] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *ICML*, 2018.

[45] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," *arXiv preprint arXiv:1807.11626*, 2018.

[46] B. A. Olshausen, C. H. Anderson, and D. C. V. Essen, "A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information," *Journal of Neuroscience*, 1993.

[47] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.

[48] L. Itti and C. Koch, "Computational modelling of visual attention," *Nature reviews neuroscience*, 2001.

[49] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order boltzmann machine," in *Conference on Neural Information Processing Systems*, 2010.

[50] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Conference on Neural Information Processing Systems*, 2014.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Conference on Neural Information Processing Systems*, 2017.

[52] T. Bluche, "Joint line segmentation and transcription for end-to-end handwritten paragraph recognition," in *Conference on Neural Information Processing Systems*, 2016.

[53] A. Miech, I. Laptev, and J. Sivic, "Learnable pooling with context gating for video classification," *arXiv:1706.06905*, 2017.

[54] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu, D. Ramanan, and T. S. Huang, "Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks," in *ICCV*, 2015.

[55] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015.

[56] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T. Chua, "SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning," in *CVPR*, 2017.

[57] J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, "Lip reading sentences in the wild," in *CVPR*, 2017.

[58] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *CVPR*, 2017.

[59] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *ECCV*, 2018.

[60] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *CVPR*, 2009.

[61] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International Journal of Computer Vision*, 2013.

[62] L. Shen, G. Sun, Q. Huang, S. Wang, Z. Lin, and E. Wu, "Multi-level discriminative dictionary learning with application to large scale image classification," *IEEE TIP*, 2015.

[63] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[64] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017.

[65] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *CVPR*, 2018.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *ICCV*, 2015.

[67] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMVC*, 2016.

[68] X. Gastaldi, "Shake-shake regularization," *arXiv preprint arXiv:1705.07485*, 2017.

[69] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[70] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[71] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *ECCV*, 2016.

[72] L. Shen, Z. Lin, G. Sun, and J. Hu, "Places401 and places365 models," https://github.com/lishen-shirley/Places2-CNNs, 2016.

[73] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[74] L. Shen, Z. Lin, and Q. Huang, "Relay backpropagation for effective learning of deep convolutional neural networks," in *ECCV*, 2016.

[75] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, 2014.

[76] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https://github.com/facebookresearch/detectron, 2018.

[77] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *CVPR*, 2017.

[78] X. Zhang, Z. Li, C. C. Loy, and D. Lin, "Polynet: A pursuit of structural diversity in very deep networks," in *CVPR*, 2017.

[79] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[80] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the limits of weakly supervised pretraining," in *ECCV*, 2018.

[81] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *ICML*, 2009.

[82] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Conference on Neural Information Processing Systems*, 2014.

[83] A. S. Morcos, D. G. Barrett, N. C. Rabinowitz, and M. Botvinick, "On the importance of single directions for generalization," in *ICLR*, 2018.

[84] J. Hu, L. Shen, S. Albanie, G. Sun, and A. Vedaldi, "Gather-excite: Exploiting feature context in convolutional neural networks," in *Conference on Neural Information Processing Systems*, 2018.