Squeeze-and-Excitation Networks

Jie Hu[1*] hujie@momenta.ai
Li Shen[2*] lishen@robots.ox.ac.uk
Gang Sun[1] sungang@momenta.ai
[1] Momenta
[2] Department of Engineering Science, University of Oxford

Abstract

Convolutional neural networks are built upon the convolution operation, which extracts informative features by fusing spatial and channel-wise information together within local receptive fields. In order to boost the representational power of a network, several recent approaches have shown the benefit of enhancing spatial encoding. In this work, we focus 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 demonstrate that by stacking these blocks together, we can construct SENet architectures that generalise extremely well across challenging datasets. Crucially, we find that SE blocks produce significant performance improvements for existing state-of-the-art deep architectures at minimal additional computational cost. SENets formed the foundation of our ILSVRC 2017 classification submission which won first place and significantly reduced the top-5 error to 2.251%*, achieving a* $\sim 25\%$ *relative improvement over the winning entry of 2016. Code and models are available at* https://github.com/hujie-frank/SENet.

1. Introduction

Convolutional neural networks (CNNs) have proven to be effective models for tackling a variety of visual tasks [21, 27, 33, 45]. For each convolutional layer, a set of filters are learned to express local spatial connectivity patterns along input channels. In other words, convolutional filters are expected to be informative combinations by fusing spatial and channel-wise information together within local receptive fields. By stacking a series of convolutional layers interleaved with non-linearities and downsampling, CNNs are capable of capturing hierarchical patterns with global receptive fields as powerful image descriptions. Recent work has demonstrated that the performance of networks can be improved by explicitly embedding learning mechanisms that help capture spatial correlations without requiring additional supervision. One such approach was popularised by the Inception architectures [16, 43], which showed that the network can achieve competitive accuracy by embedding multi-scale processes in its modules. More recent work has sought to better model spatial dependence [1, 31] and incorporate spatial attention [19].

In this paper, we investigate a different aspect of architectural design - the channel relationship, by introducing a new architectural unit, which we term the "*Squeeze-and-Excitation*" (SE) block. Our goal is to improve the representational power of a network by explicitly modelling the interdependencies between the channels of its convolutional features. To achieve this, 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 basic structure of the SE building block is illustrated in Fig. 1. For any given transformation $\mathbf{F}_{tr} : \mathbf{X} \rightarrow \mathbf{U}$, $\mathbf{X} \in \mathbb{R}^{H' \times W' \times C'}$, $\mathbf{U} \in \mathbb{R}^{H \times W \times C}$, (e.g. a convolution or a set of convolutions), we can construct a corresponding SE block to perform feature recalibration as follows. The features $\mathbf{U}$ are first passed through a *squeeze* operation, which aggregates the feature maps across spatial dimensions $H \times W$ to produce a channel descriptor. This descriptor embeds the global distribution of channel-wise feature responses, enabling information from the global receptive field of the network to be leveraged by its lower layers. This is followed by an *excitation* operation, in which sample-specific activations, learned for each channel by a self-gating mechanism based on channel dependence, govern the excitation of each channel. The feature maps $\mathbf{U}$ are then reweighted to generate the output of the SE block which can then be fed directly into subsequent layers.

An SE network can be generated by simply stacking a collection of SE building blocks. SE blocks can also be used as a drop-in replacement for the original block at *any depth* in the architecture. However, while the template for the building block is generic, as we show in Sec. 6.4, the role it performs at different depths adapts to the needs of the network. In the early layers, it learns to excite informative

*Equal contribution.

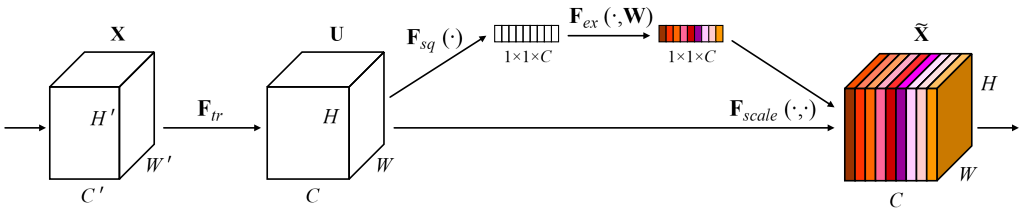


Figure 1: A Squeeze-and-Excitation block.

features in a class agnostic manner, bolstering the quality of the shared lower level representations. In later layers, the SE block becomes increasingly specialised, and responds to different inputs in a highly *class-specific* manner. Consequently, the benefits of feature recalibration conducted by SE blocks can be accumulated through the entire network.

The development of new CNN architectures is a challenging engineering task, typically involving the selection of many new hyperparameters and layer configurations. By contrast, the design of the SE block outlined above is simple, and can be used directly with existing state-of-the-art architectures whose modules can be strengthened by direct replacement with their SE counterparts.

Moreover, as shown in Sec. 4, SE blocks are computationally lightweight and impose only a slight increase in model complexity and computational burden. To support these claims, we develop several SENets and provide an extensive evaluation on the ImageNet 2012 dataset [34]. To demonstrate their general applicability, we also present results beyond ImageNet, indicating that the proposed approach is not restricted to a specific dataset or a task.

Using SENets, we won the first place in the ILSVRC 2017 classification competition. Our top performing model ensemble achieves a 2.251% top-5 error on the test set[1]. This represents a $\sim 25\%$ relative improvement in comparison to the winner entry of the previous year (with a top-5 error of 2.991%).

2. Related Work

Deep architectures. VGGNets [39] and Inception models [43] demonstrated the benefits of increasing depth. Batch normalization (BN) [16] improved gradient propagation by inserting units to regulate layer inputs, stabilising the learning process. ResNets [10, 11] showed the effectiveness of learning deeper networks through the use of identity-based skip connections. Highway networks [40] employed a gating mechanism to regulate shortcut connections. Reformulations of the connections between network layers [5, 14] have been shown to further improve the learning and representational properties of deep networks.

An alternative line of research has explored ways to tune the functional form of the modular components of a network. Grouped convolutions can be used to increase cardinality (the size of the set of transformations) [15, 47]. Multi-branch convolutions can be interpreted as a generalisation of this concept, enabling more flexible compositions of operators [16, 42, 43, 44]. Recently, compositions which have been learned in an automated manner [26, 54, 55] have shown competitive performance. Cross-channel correlations are typically mapped as new combinations of features, either independently of spatial structure [6, 20] or jointly by using standard convolutional filters [24] with 1×1 convolutions. Much of this work 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.

Attention and gating mechanisms. Attention can be viewed, broadly, as a tool to bias the allocation of available processing resources towards the most informative components of an input signal [17, 18, 22, 29, 32]. The benefits of such a mechanism have been shown across a range of tasks, from localisation and understanding in images [3, 19] to sequence-based models [2, 28]. It is typically implemented in combination with a gating function (e.g. a softmax or sigmoid) and sequential techniques [12, 41]. Recent work has shown its applicability to tasks such as image captioning [4, 48] and lip reading [7]. In these applications, it is often used on top of one or more layers representing higher-level abstractions for adaptation between modalities. Wang et al. [46] introduce a powerful trunk-and-mask attention mechanism using an hourglass module [31]. This high capacity unit is inserted into deep residual networks between intermediate stages. In contrast, our proposed SE block is a lightweight gating mechanism, specialised to model channel-wise relationships in a computationally efficient manner and designed to enhance the representational power of basic modules throughout the network.

3. Squeeze-and-Excitation Blocks

The *Squeeze-and-Excitation* block is a computational unit which can be constructed for any given transformation $\mathbf{F}_{tr} : \mathbf{X} \rightarrow \mathbf{U}$, $\mathbf{X} \in \mathbb{R}^{H' \times W' \times C'}$, $\mathbf{U} \in \mathbb{R}^{H \times W \times C}$. For

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

simplicity, in the notation that follows we take $\mathbf{F}_{tr}$ to be a convolutional operator. Let $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_C]$ 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 of $\mathbf{F}_{tr}$ as $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_C]$, where

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

Here $*$ denotes convolution, $\mathbf{v}_c = [\mathbf{v}_c^1, \mathbf{v}_c^2, \ldots, \mathbf{v}_c^{C'}]$ and $\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{C'}]$ (to simplify the notation, bias terms are omitted), while $\mathbf{v}_c^s$ is a 2D spatial kernel, and therefore represents a single channel of $\mathbf{v}_c$ which acts on the corresponding channel of $\mathbf{X}$. Since the output is produced by a summation through all channels, the channel dependencies are implicitly embedded in $\mathbf{v}_c$, but these dependencies are entangled with the spatial correlation captured by the filters. Our goal is to ensure that the network is able to increase its sensitivity to informative features so that they can be exploited by subsequent transformations, and to suppress less useful ones. We propose to achieve this by explicitly modelling channel interdependencies to recalibrate filter responses in two steps, *squeeze* and *excitation*, before they are fed into next transformation. A diagram of an SE building 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. This is an issue that becomes more severe in the lower layers of the network whose receptive field sizes are small.

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 spatial dimensions $H \times W$, where the c-th element of $\mathbf{z}$ is calculated by:

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

Discussion. The transformation output $\mathbf{U}$ can be interpreted as a collection of the local descriptors whose statistics are expressive for the whole image. Exploiting such information is prevalent in feature engineering work [35, 38, 49]. We opt for the simplest, global average pooling, noting that more sophisticated aggregation 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 opposed to one-hot activation. To meet these criteria, we opt to employ a simple gating mechanism with a sigmoid activation:

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

where δ refers to the ReLU [30] 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 parameters $\mathbf{W}_1$ with reduction ratio r (this parameter choice is discussed in Sec. 6.4), a ReLU and then a dimensionality-increasing layer with parameters $\mathbf{W}_2$. The final output of the block is obtained by rescaling the transformation output $\mathbf{U}$ with the activations:

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

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 feature map $\mathbf{u}_c \in \mathbb{R}^{H \times W}$ and the scalar s_c.

Discussion. The activations act as channel weights adapted to the input-specific descriptor $\mathbf{z}$. In this regard, SE blocks intrinsically introduce dynamics conditioned on the input, helping to boost feature discriminability.

3.3. Exemplars: SE-Inception and SE-ResNet

It is straightforward to apply the SE block to AlexNet [21] and VGGNet [39]. 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 integrating SE blocks into modern architectures with sophisticated designs.

For non-residual networks, such as Inception network, SE blocks are constructed for the network by taking the transformation $\mathbf{F}_{tr}$ to be an entire Inception module (see Fig. 2). By making this change for each such module in the architecture, we construct an *SE-Inception* network. Moreover, SE blocks are sufficiently flexible to be used in 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. More variants that integrate with ResNeXt [47], Inception-ResNet [42], MobileNet [13] and ShuffleNet [52] can be constructed by following the similar schemes. We describe the architecture of SE-ResNet-50 and SE-ResNeXt-50 in Table 1.

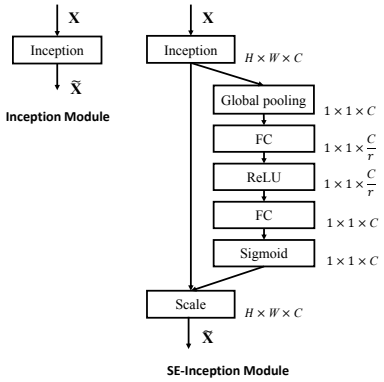


Figure 2: The schema of the original Inception module (left) and the SE-Inception module (right).

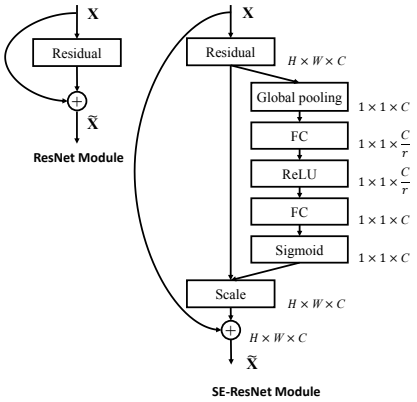


Figure 3: The schema of the original Residual module (left) and the SE-ResNet module (right).

4. Model and Computational Complexity

For the proposed SE block to be viable in practice, it must provide an effective trade-off between model complexity and performance which is important for scalability. We set the reduction ratio r to be 16 in all experiments, except where stated otherwise (more discussion can be found in Sec. 6.4). To illustrate the cost of the module, we take the comparison between ResNet-50 and SE-ResNet-50 as an example, where the accuracy of SE-ResNet-50 is superior to ResNet-50 and approaches that of a deeper ResNet-101 network (shown in Table 2). 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 fully connected layers in the *excitation* phase, followed by an inexpensive channel-wise scaling operation. In aggregate, SE-ResNet-50 requires $\sim$3.87 GFLOPs, corresponding to a 0.26% relative increase over the original ResNet-50.

In practice, with a training mini-batch of 256 images, a single pass forwards and backwards through ResNet-50 takes 190 ms, compared to 209 ms for SE-ResNet-50 (both timings are performed on a server with 8 NVIDIA Titan X GPUs). We argue that this represents a reasonable overhead particularly since global pooling and small inner-product operations are less optimised in existing GPU libraries. Moreover, due to its importance for embedded device applications, we also benchmark CPU inference time for each model: for a 224×224 pixel input image, ResNet-50 takes 164 ms, compared to 167 ms for SE-ResNet-50. The small additional computational overhead required by the SE block is justified by its contribution to model performance.

Next, we consider the additional parameters introduced by the proposed block. All of them are contained in the two FC layers of the gating mechanism, which constitute a small fraction of the total network capacity. More precisely, the number of additional parameters introduced 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 (where each 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 repeated block number for stage s. SE-ResNet-50 introduces $\sim$2.5 million additional parameters beyond the $\sim$25 million parameters required by ResNet-50, corresponding to a $\sim 10\%$ increase. The majority of these parameters come from the last stage of the network, where excitation is performed across the greatest channel dimensions. However, we found that the comparatively expensive final stage of SE blocks could be removed at a marginal cost in performance ($<0.1\%$ top-1 error on ImageNet) to reduce the relative parameter increase to $\sim 4\%$, which may prove useful in cases where parameter usage is a key consideration (see further discussion in Sec. 6.4).

5. Implementation

Each plain network and its corresponding SE counterpart are trained with identical optimisation schemes. During training on ImageNet, we follow standard practice and perform data augmentation with random-size cropping [43] to 224×224 pixels (299×299 for Inception-ResNet-v2 [42] and SE-Inception-ResNet-v2) and random horizontal flipping. Input images are normalised through mean channel subtraction. In addition, we adopt the data balancing strategy described in [36] for mini-batch sampling. The networks 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 mini-batch size of 1024. The initial learning rate is set to 0.6 and decreased by a factor of 10 every 30 epochs. All models are trained

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		
	$\begin{bmatrix}\text{conv}, 1\times1, 64\\ \text{conv}, 3\times3, 64\\ \text{conv}, 1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}\text{conv}, 1\times1, 64\\ \text{conv}, 3\times3, 64\\ \text{conv}, 1\times1, 256\\ fc, [16, 256]\end{bmatrix}\times3$	$\begin{bmatrix}\text{conv}, 1\times1, 128\\ \text{conv}, 3\times3, 128 \quad C=32\\ \text{conv}, 1\times1, 256\\ fc, [16, 256]\end{bmatrix}\times3$
28 × 28	$\begin{bmatrix}\text{conv}, 1\times1, 128\\ \text{conv}, 3\times3, 128\\ \text{conv}, 1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}\text{conv}, 1\times1, 128\\ \text{conv}, 3\times3, 128\\ \text{conv}, 1\times1, 512\\ fc, [32, 512]\end{bmatrix}\times4$	$\begin{bmatrix}\text{conv}, 1\times1, 256\\ \text{conv}, 3\times3, 256 \quad C=32\\ \text{conv}, 1\times1, 512\\ fc, [32, 512]\end{bmatrix}\times4$
14 × 14	$\begin{bmatrix}\text{conv}, 1\times1, 256\\ \text{conv}, 3\times3, 256\\ \text{conv}, 1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}\text{conv}, 1\times1, 256\\ \text{conv}, 3\times3, 256\\ \text{conv}, 1\times1, 1024\\ fc, [64, 1024]\end{bmatrix}\times6$	$\begin{bmatrix}\text{conv}, 1\times1, 512\\ \text{conv}, 3\times3, 512 \quad C=32\\ \text{conv}, 1\times1, 1024\\ fc, [64, 1024]\end{bmatrix}\times6$
7×7	$\begin{bmatrix}\text{conv}, 1\times1, 512\\ \text{conv}, 3\times3, 512\\ \text{conv}, 1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}\text{conv}, 1\times1, 512\\ \text{conv}, 3\times3, 512\\ \text{conv}, 1\times1, 2048\\ fc, [128, 2048]\end{bmatrix}\times3$	$\begin{bmatrix}\text{conv}, 1\times1, 1024\\ \text{conv}, 3\times3, 1024 \quad C=32\\ \text{conv}, 1\times1, 2048\\ fc, [128, 2048]\end{bmatrix}\times3$
1 × 1	global average pool, 1000-d fc, softmax		

Table 1: (**Left**) ResNet-50. (**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.

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [10]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [10]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [10]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [47]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [47]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
VGG-16 [39]	-	-	27.02	8.81	15.47	$25.22_{(1.80)}$	$7.70_{(1.11)}$	15.48
BN-Inception [16]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [42]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

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. 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 [42]), which may slightly improve results. VGG-16 and SE-VGG-16 are trained with batch normalization.

for 100 epochs from scratch, using the weight initialisation strategy described in [9].

When testing, we apply a centre crop evaluation on the validation set, where 224×224 pixels are cropped from each image whose 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).

6. Experiments

6.1. ImageNet Classification

The ImageNet 2012 dataset is comprised of 1.28 million training images and 50K validation images from 1000 classes. We train networks on the training set and report the top-1 and the top-5 errors.

Network depth. We first compare the SE-ResNet against ResNet architectures with different depths. The results in Table 2 shows 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 computational overhead (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%. Fig. 4 depicts the training and validation curves of SE-ResNet-50 and ResNet-50 (the

	original		re-implementation				SENet			
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	MFLOPs	Million Parameters	top-1 err.	top-5 err.	MFLOPs	Million Parameters
MobileNet [13]	29.4	-	29.1	10.1	569	4.2	$25.3_{(3.8)}$	$7.9_{(2.2)}$	572	4.7
ShuffleNet [52]	34.1	-	33.9	13.6	140	1.8	$31.7_{(2.2)}$	$11.7_{(1.9)}$	142	2.4

Table 3: Single-crop error rates (%) on the ImageNet validation set and complexity comparisons. Here, MobileNet refers to "1.0 MobileNet-224" in [13] and ShuffleNet refers to "ShuffleNet $1 \times (g = 3)$" in [52].

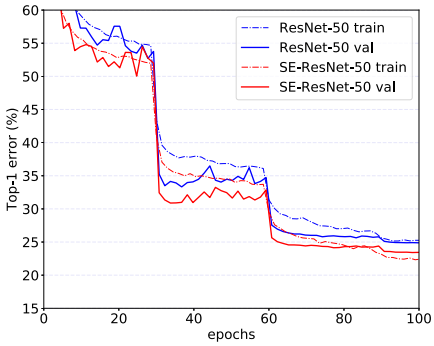


Figure 4: Training curves of ResNet-50 and SE-ResNet-50 on ImageNet.

curves of more networks are shown in supplementary material). 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 performance improvements are consistent through training across a range of different depths, suggesting that the improvements induced by SE blocks can be used in combination with increasing the depth of the base architecture.

Integration with modern architectures. We next investigate the effect of combining SE blocks with another two state-of-the-art architectures, Inception-ResNet-v2 [42] and ResNeXt (using the setting of 32×4d) [47], which both introduce prior structures in modules.

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). The results in Table 2 illustrate the significant performance improvement induced by SE blocks when introduced 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 double the number of parameters and computational overhead. As for the experiments of Inception-ResNet-v2, we conjecture the difference of cropping strategy might lead to the gap between their reported result and our re-implemented one, as their original image size has not been clarified in [42] while we crop the 299×299 region from a relatively larger image (where the shorter edge is resized to 352). SE-Inception-ResNet-v2 (4.79% top-5 error) outperforms our reimplemented Inception-ResNet-v2 (5.21% top-5 error) by 0.42% (a relative improvement of 8.1%) as well as the reported result in [42].

	224×224		320×320 / 299×299	
	top-1 err.	top-5 err.	top-1 err.	top-5 err.
ResNet-152 [10]	23.0	6.7	21.3	5.5
ResNet-200 [11]	21.7	5.8	20.1	4.8
Inception-v3 [44]	-	-	21.2	5.6
Inception-v4 [42]	-	-	20.0	5.0
Inception-ResNet-v2 [42]	-	-	19.9	4.9
ResNeXt-101 (64×4d) [47]	20.4	5.3	19.1	4.4
DenseNet-264 [14]	22.15	6.12	-	-
Attention-92 [46]	-	-	19.5	4.8
Very Deep PolyNet [51] †	-	-	18.71	4.25
PyramidNet-200 [8]	20.1	5.4	19.2	4.7
DPN-131 [5]	19.93	5.12	18.55	4.16
SENet-154	**18.68**	**4.47**	**17.28**	**3.79**
NASNet-A (6@4032) [55] †	-	-	17.3‡	3.8‡
SENet-154 (post-challenge)	-	-	**16.88‡**	**3.58‡**

Table 4: Single-crop error rates of state-of-the-art CNNs on ImageNet validation set. The size of test crop is 224×224 and 320×320 / 299×299 as in [11]. † denotes the model with a larger crop 331×331. ‡ denotes the post-challenge result. SENet-154 (post-challenge) is trained with a larger input size 320×320 compared to the original one with the input size 224×224.

We also assess the effect of SE blocks when operating on *non-residual* networks by conducting experiments with the VGG-16 [39] and BN-Inception architecture [16]. As a deep network is tricky to optimise [16, 39], to facilitate the training of VGG-16 from scratch, we add a Batch Normalization layer after each convolution. We apply the identical scheme for training SE-VGG-16. The results of the comparison are shown in Table 2, exhibiting the same phenomena that emerged in the residual architectures.

Finally, we evaluate on two representative efficient architectures, MobileNet [13] and ShuffleNet [52] in Table 3, showing that SE blocks can consistently improve the accuracy by a large margin at minimal increases in computational cost. These experiments demonstrate that improvements induced by SE blocks can be used in combination with a wide range of architectures. Moreover, this result holds for both residual and non-residual foundations.

	top-1 err.	top-5 err.
Places-365-CNN [37]	41.07	11.48
ResNet-152 (ours)	41.15	11.61
SE-ResNet-152	**40.37**	**11.01**

Table 5: Single-crop error rates (%) on Places365 validation set.

Results on ILSVRC 2017 Classification Competition. SENets formed the foundation of our submission to the competition where we won 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 2.251% top-5 error on the test set. One of our high-performing networks, which we term *SENet-154*, was constructed by integrating SE blocks with a modified ResNeXt [47] (details are provided in supplemental material), the goal of which is to reach the best possible accuracy with less emphasis on model complexity. We compare it with the top-performing published models on the ImageNet validation set in Table 4. Our model achieved a top-1 error of 18.68% and a top-5 error of 4.47% using a 224×224 centre crop evaluation. To enable a fair comparison, we provide a 320×320 centre crop evaluation, showing a significant performance improvement on prior work. After the competition, we train an SENet-154 with a larger input size 320×320, achieving the lower error rate under both the top-1 (16.88%) and the top-5 (3.58%) error metrics.

6.2. Scene Classification

We conduct experiments on the Places365-Challenge dataset [53] for scene classification. It comprises 8 million training images and 36,500 validation images across 365 categories. Relative to classification, the task of scene understanding can provide a better assessment of the ability of a model to generalise well and handle abstraction, since it requires the capture of more complex data associations and robustness to a greater level of appearance variation.

We use ResNet-152 as a strong baseline to assess the effectiveness of SE blocks and follow the training and evaluation protocols in [37]. Table 5 shows the results of ResNet-152 and SE-ResNet-152. Specifically, 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 perform well on different datasets. This SENet surpasses the previous state-of-the-art model Places-365-CNN [37] which has a top-5 error of 11.48% on this task.

6.3. Object Detection on COCO

We further evaluate the generalisation of SE blocks on object detection task using COCO dataset [25] which contains 80k training images and 40k validation images, following [10]. We use Faster R-CNN [33] as the detection method and follow the basic implementation in [10].

	AP@IoU=0.5	AP
ResNet-50	45.2	25.1
SE-ResNet-50	46.8	26.4
ResNet-101	48.4	27.2
SE-ResNet-101	49.2	27.9

Table 6: Object detection results on the COCO 40k validation set by using the basic Faster R-CNN.

Here our intention is to evaluate the benefit of replacing the base architecture ResNet with SE-ResNet, so that the improvements can be attributed to better representations. Table 6 shows the results by using ResNet-50, ResNet-101 and their SE counterparts on the validation set, respectively. SE-ResNet-50 outperforms ResNet-50 by 1.3% (a relative 5.2% improvement) on COCO's standard metric AP and 1.6% on AP@IoU=0.5. Importantly, SE blocks are capable of benefiting the deeper architecture ResNet-101 by 0.7% (a relative 2.6% improvement) on the AP metric.

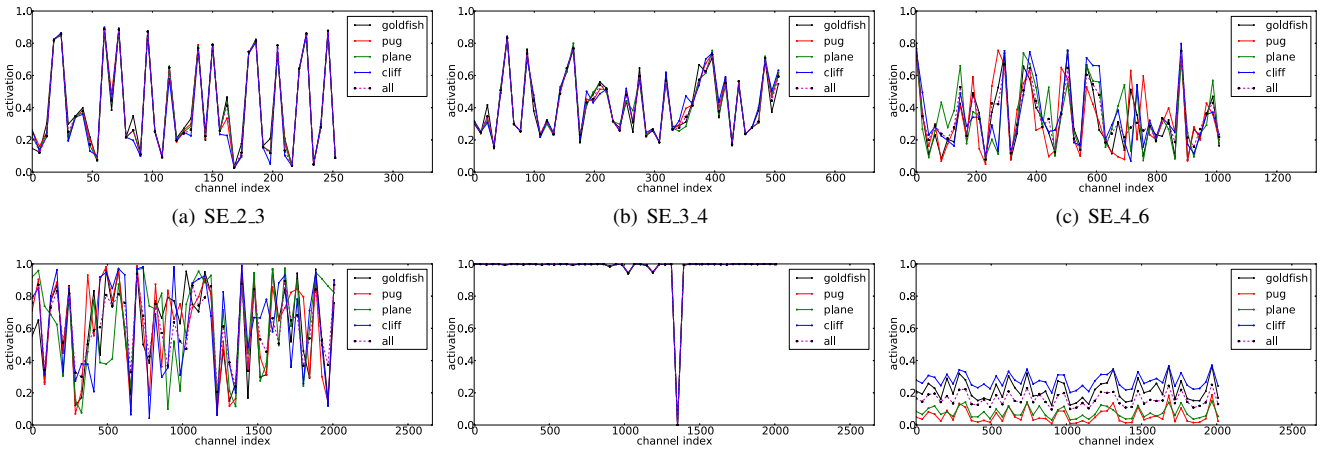
6.4. Analysis and Interpretation

Reduction ratio. The reduction ratio r introduced in Eqn. (5) is an important hyperparameter which allows us to vary the capacity and computational cost of the SE blocks in the model. To investigate this relationship, we conduct experiments based on SE-ResNet-50 for a range of different r values. The comparison in Table 7 reveals that performance does not improve monotonically with increased capacity. This is likely to be a result of enabling the SE block to overfit the channel interdependencies of the training set. In particular, we found that setting $r = 16$ achieved a good tradeoff between accuracy and complexity and consequently, we used this value for all experiments.

The role of Excitation. While SE blocks have been empirically shown to improve network performance, we would also like to understand how the self-gating *excitation* mechanism operates in practice. To provide a clearer picture of the behaviour of SE blocks, in this section we study example activations from the SE-ResNet-50 model and examine their distribution with respect to different classes at different blocks. Specifically, we sample four classes from the ImageNet dataset that exhibit semantic and appearance diver-

Ratio r	top-1 err.	top-5 err.	Million Parameters
4	23.21	6.63	35.7
8	23.19	6.64	30.7
16	23.29	6.62	28.1
32	23.40	6.77	26.9
original	24.80	7.48	25.6

Table 7: Single-crop error rates (%) on ImageNet validation set and parameter sizes for SE-ResNet-50 at different reduction ratios r. Here *original* refers to ResNet-50.



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

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

Figure 5: Activations induced by *Excitation* in the different modules of SE-ResNet-50 on ImageNet. The module is named as "*SE_stageID_blockID*".

sity, namely *goldfish*, *pug*, *plane* and *cliff* (example images from these classes are shown in supplemental material). 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 in each stage (immediately prior to downsampling) and plot their distribution in Fig. 5. For reference, we also plot the distribution of average activations across all 1000 classes.

We make the following three observations about the role of *Excitation*. First, the distribution across different classes is nearly identical in lower layers, 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 of the network. Interestingly however, the second observation is that at greater 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. The two observations are consistent with findings in previous work [23, 50], namely that lower layer features are typically more general (i.e. class agnostic in the context of classification) while higher layer features have greater specificity. As a result, representation learning benefits from the recalibration induced by SE blocks which adaptively facilitates feature extraction and specialisation to the extent that it is needed. Finally, 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 1 and the remainder is close to 0. At the point at which all activations take the value 1, this block would become a standard residual block. 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 slight 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 Sec. 4 which demonstrated that the overall parameter count could be significantly reduced by removing the SE blocks for the last stage with only a marginal loss of performance.

7. Conclusion

In this paper we proposed the SE block, a novel architectural unit designed to improve the representational capacity of a network by enabling it to perform dynamic channel-wise feature recalibration. Extensive experiments demonstrate the effectiveness of SENets which achieve state-of-the-art performance on multiple datasets. In addition, they provide some insight into the limitations of previous architectures in modelling channel-wise feature dependencies, which we hope may prove useful for other tasks requiring strong discriminative features. Finally, the feature importance induced by SE blocks may be helpful to related fields such as network pruning for compression.

Acknowledgements. We would like to thank Professor Andrew Zisserman for his helpful comments and Samuel Albanie for his discussions and writing edit for the paper. We would like to thank Chao Li for his contributions in the training system. Li Shen is supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract number 2014-14071600010. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon.

References

[1] 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. 1

[2] T. Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In *NIPS*, 2016. 2

[3] 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. 2

[4] 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. 2

[5] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. In *NIPS*, 2017. 2, 6

[6] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017. 2

[7] J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman. Lip reading sentences in the wild. In *CVPR*, 2017. 2

[8] D. Han, J. Kim, and J. Kim. Deep pyramidal residual networks. In *CVPR*, 2017. 6

[9] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 2015. 5

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 5, 6, 7

[11] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 2, 6

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997. 2

[13] 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. 3, 6

[14] G. Huang, Z. Liu, K. Q. Weinberger, and L. Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 2, 6

[15] Y. Ioannou, D. Robertson, R. Cipolla, and A. Criminisi. Deep roots: Improving CNN efficiency with hierarchical filter groups. In *CVPR*, 2017. 2

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 1, 2, 5, 6

[17] L. Itti and C. Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2001. 2

[18] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE TPAMI*, 1998. 2

[19] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015. 1, 2

[20] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014. 2

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 3

[22] H. Larochelle and G. E. Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *NIPS*, 2010. 2

[23] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009. 8

[24] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv:1312.4400*, 2013. 2

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Common objects in context. *ECCV*, 2014. 7

[26] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. *arXiv: 1711.00436*, 2017. 2

[27] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[28] A. Miech, I. Laptev, and J. Sivic. Learnable pooling with context gating for video classification. *arXiv:1706.06905*, 2017. 2

[29] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. In *NIPS*, 2014. 2

[30] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 3

[31] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 1, 2

[32] 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. 2

[33] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 7

[34] 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. *IJCV*, 2015. 2

[35] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *RR-8209, INRIA*, 2013. 3

[36] L. Shen, Z. Lin, and Q. Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016. 4

[37] L. Shen, Z. Lin, G. Sun, and J. Hu. Places401 and places365 models. https://github.com/lishen-shirley/Places2-CNNs, 2016. 7

[38] 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. 3

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2, 3, 5, 6

[40] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *NIPS*, 2015. 2

[41] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber. Deep networks with internal selective attention through feedback connections. In *NIPS*, 2014. 2

[42] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *ICLR Workshop*, 2016. 2, 3, 4, 5, 6

[43] 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. 1, 2, 4

[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In

CVPR, 2016. 2, 6

[45] A. Toshev and C. Szegedy. DeepPose: Human pose estimation via deep neural networks. In *CVPR*, 2014. 1

[46] 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. 2, 6

[47] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 2, 3, 5, 6, 7

[48] 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. 2

[49] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009. 3

[50] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014. 8

[51] X. Zhang, Z. Li, C. C. Loy, and D. Lin. Polynet: A pursuit of structural diversity in very deep networks. In *CVPR*, 2017. 6

[52] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv:1707.01083*, 2017. 3, 6

[53] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE TPAMI*, 2017. 7

[54] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 2

[55] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *arXiv: 1707.07012*, 2017. 2, 6