Gradually Updated Neural Networks for Large-Scale Image Recognition

Siyuan Qiao [1] **Zhishuai Zhang** [1] **Wei Shen** [1 2] **Bo Wang** [3] **Alan Yuille** [1]

Abstract

Depth is one of the keys that make neural networks succeed in the task of large-scale image recognition. The state-of-the-art network architectures usually increase the depths by cascading convolutional layers or building blocks. In this paper, we present an alternative method to increase the depth. Our method is by introducing computation orderings to the channels within convolutional layers or blocks, based on which we gradually compute the outputs in a channel-wise manner. The added orderings not only increase the depths and the learning capacities of the networks without any additional computation costs, but also eliminate the overlap singularities so that the networks are able to converge faster and perform better. Experiments show that the networks based on our method achieve the state-of-the-art performances on CIFAR and ImageNet datasets.

1. Introduction

Deep neural networks have become the state-of-the-art systems for image recognition (He et al., 2016a; Huang et al., 2017b; Krizhevsky et al., 2012; Qiao et al., 2017a; Simonyan & Zisserman, 2014; Szegedy et al., 2015; Wang et al., 2017; Zeiler & Fergus, 2013) as well as other vision tasks (Chen et al., 2015; Girshick et al., 2014; Long et al., 2015; Qiao et al., 2017b; Ren et al., 2015; Shen et al., 2015; Xie & Tu, 2015). The architectures keep going deeper, *e.g.*, from five convolutional layers (Krizhevsky et al., 2012) to 1001 layers (He et al., 2016b). The benefit of deep architectures is their strong learning capacities because each new layer can potentially introduce more non-linearities and typically uses larger receptive fields (Simonyan & Zisserman, 2014). In addition, adding certain types of layers (e.g. (He et al., 2016b)) will not harm the performance theoretically since they can just learn identity mapping. This makes stacking up layers more appealing in the network designs.

[1]Johns Hopkins University [2]Shanghai University [3]Hikvision Research. Correspondence to: Siyuan Qiao <siyuan.qiao@jhu.edu>.

Technical Report

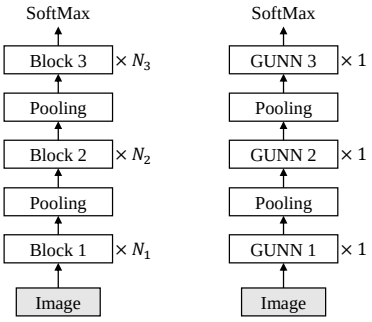

Figure 1. Comparing architecture designs based on cascading convolutional building blocks (left) and GUNN (right). Cascading-based architecture increases the depth by repeating the blocks. GUNN-based networks increases the depth by adding computation orderings to the channels of the building blocks.

Although deeper architectures usually lead to stronger learning capacities, cascading convolutional layers (*e.g.* VGG (Simonyan & Zisserman, 2014)) or blocks (*e.g.* ResNet (He et al., 2016a)) is not necessarily the only method to achieve this goal. In this paper, we present a new way to increase the depth of the networks as an alternative to stacking up convolutional layers or blocks. Figure 2 provides an illustration that compares our proposed convolutional network that gradually updates the feature representations against the traditional convolutional network that computes its output simultaneously. By only adding an ordering to the channels without any additional computation, the later computed channels become deeper than the corresponding ones in the traditional convolutional network. We refer to the neural networks with the proposed computation orderings on the channels as Gradually Updated Neural Networks (GUNN). Figure 1 provides two examples of architecture designs based on cascading building blocks and GUNN. Without repeating the building blocks, GUNN increases the depths of the networks as well as their learning capacities.

It is clear that converting plain networks to GUNN increases the depths of the networks without any additional computations. What is less obvious is that GUNN in fact eliminates the overlap singularities inherent in the loss landscapes of the cascading-based convolutional networks, which have been shown to adversely affect the training of deep neural networks as well as their performances (Wei et al., 2008;

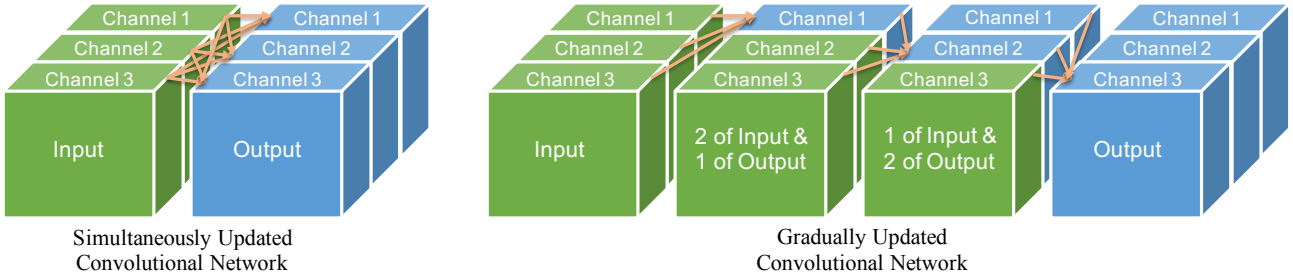

Figure 2. Comparing Simultaneously Updated Convolutional Network and Gradually Updated Convolutional Network. Left is a traditional convolutional network with three channels in both the input and the output. Right is our proposed convolutional network which decomposes the original computation into three sequential channel-wise convolutional operations. In our proposed GUNN-based architectures, the updates are done by *residual learning* (He et al., 2016a), which we do not show in this figure.

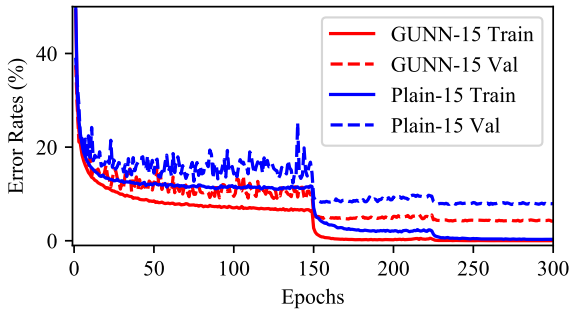

Figure 3. Training dynamics on CIFAR-10 dataset.

Orhan & Pitkow, 2018). Overlap singularity is when internal neurons collapse into each other, *i.e.* they are unidentifiable by their activations. It happens in the networks, increases the training difficulties and degrades the performances (Orhan & Pitkow, 2018). However, if a plain network is converted to GUNN, the added computation orderings will break the symmetry between the neurons. We prove that the internal neurons in GUNN are impossible to collapse into each other. As a result, the effective dimensionality can be kept during training and the model will be free from the degeneracy caused by collapsed neurons. Reflected in the training dynamics and the performances, this means that converting to GUNN will make the plain networks *easier to train* and *perform better*. Figure 3 compares the training dynamics of a 15-layer plain network on CIFAR-10 dataset (Krizhevsky & Hinton, 2009) before and after converted to GUNN.

In this paper, we test our proposed GUNN on highly competitive benchmark datasets, *i.e.* CIFAR (Krizhevsky & Hinton, 2009) and ImageNet (Russakovsky et al., 2015). Experimental results demonstrate that our proposed GUNN-based networks achieve the state-of-the-art performances compared with the previous cascading-based architectures.

2. Related Work

The research focuses of image recognition have moved from feature designs (Dalal & Triggs, 2005; Lowe, 2004) to architecture designs (He et al., 2016a; Huang et al., 2017b; Krizhevsky et al., 2012; Sermanet et al., 2014; Simonyan & Zisserman, 2014; Szegedy et al., 2015; Xie et al., 2017; Zeiler & Fergus, 2013) due to the recent success of the deep neural networks. Highway Networks (Srivastava et al., 2015) proposed architectures that can be trained end-to-end with more than 100 layers. The main idea of Highway Networks is to use bypassing paths. This idea was further investigated in ResNet (He et al., 2016a), which simplifies the bypassing paths by using only identity mappings. As learning ultra-deep networks became possible, the depths of the models have increased tremendously. ResNet with pre-activation (He et al., 2016b) and ResNet with stochastic depth (Huang et al., 2016) even managed to train neural networks with more than 1000 layers. FractalNet (Larsson et al., 2016) argued that in addition to summation, concatenation also helps train a deep architecture. More recently, ResNeXt (Xie et al., 2017) used group convolutions in ResNet and outperformed the original ResNet. DenseNet (Huang et al., 2017b) proposed an architecture with dense connections by feature concatenation. Dual Path Net (Chen et al., 2017) finds a middle point between ResNet and DenseNet by concatenating them in two paths. Unlike the above cascading-based methods, GUNN eliminates the overlap singularities caused by the architecture symmetry. The detailed analyses can be found in Section 4.3.

Alternative to increasing the depth of the neural networks, another trend is to increase the widths of the networks. GoogleNet (Szegedy et al., 2015; 2016) proposed an *Inception* module to concatenate feature maps produced by different filters. Following ResNet (He et al., 2016a), the WideResNet (Zagoruyko & Komodakis, 2016) argued that compared with increasing the depth, increasing the width of the networks can be more effective in improving the performances. Besides varying the width and the depth, there are also other design strategies for deep neural networks (Hariharan et al., 2015; Kontschieder et al., 2015; Pezeshki et al., 2016; Rasmus et al., 2015; Yang & Ramanan, 2015). Deeply-Supervised Nets (Lee et al., 2014) used auxiliary classifiers to provide direct supervisions for the internal layers. Network in Network (Lin et al., 2013) adds micro perceptrons to the convolutional layers.

3. Model

3.1. Feature Update

We consider a feature transformation $\mathcal{F}: \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$, where n denotes the channel of the features and m denotes the feature location on the 2-D feature map. For example, $\mathcal{F}$ can be a convolutional layer with n channels for both the input and the output. Let $\boldsymbol{x} \in \mathbb{R}^{m\times n}$ be the input and $\boldsymbol{y} \in \mathbb{R}^{m\times n}$ be the output, we have

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}) \tag{1}$$

Suppose that $\mathcal{F}$ can be decomposed into channel-wise transformation $\mathcal{F}_c(\cdot)$ that are independent with eath other, then for any location k and channel c we have

$$y_c^k = \mathcal{F}_c(\boldsymbol{x}^{r(k)}) \tag{2}$$

where $\boldsymbol{x}^{r(k)}$ denotes the receptive field of the location k and $\mathcal{F}_c$ denotes the transformation on channel c.

Let U_C denote a feature update on channel set C, *i.e.*,

$$\begin{aligned} U_C(\boldsymbol{x}): \quad & y_c^k = \mathcal{F}_c(\boldsymbol{x}^{r(k)}), \forall c \in C, k \\ & y_c^k = x_c^k, \forall c \in \overline{C}, k \end{aligned} \tag{3}$$

Then, $U_C = \mathcal{F}$ when $C = \{1, ..., n\}$.

3.2. Gradually Updated Neural Networks

By defining the feature update U_C on channel set C, the commonly used one-layer CNN is a special case of feature updates where every channel is updated simultaneously. However, we can also update the channels gradually. For example, the proposed GUNN can be formulated by

$$\begin{aligned} &\text{GUNN}(\boldsymbol{x}) = (U_{c_l} \circ U_{c_{(l-1)}} \circ ... \circ U_{c_2} \circ U_{c_1})(\boldsymbol{x}) \\ &\text{where } \bigcup_{i=1}^{l} c_i = \{1, 2, ..., n\} \text{ and } c_i \cap c_j = \Phi, \ \forall i \neq j \end{aligned} \tag{4}$$

When $l = 1$, GUNN is equivalent to $\mathcal{F}$.

Note that the number of parameters and computation of GUNN are the same as those of the corresponding $\mathcal{F}$ for any partitions $c_1, ..., c_l$ of $\{1, ..., n\}$. However, by decomposing $\mathcal{F}$ into channel-wise transformations and sequentially applying them, the later computed channels are deeper than the previous ones. As a result, the depth of the network can be increased, as well as the network's learning capacity.

3.3. Channel-wise Update by Residual Learning

We consider the residual learning proposed by ResNet (He et al., 2016a) in our model. Specifically, we consider the channel-wise transformation $\mathcal{F}_c: \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times 1}$ to be

$$\mathcal{F}_c(x) = \mathcal{G}_c(x) + x_c \tag{5}$$

Algorithm 1 Back-propagation for GUNN

Input : $U(\cdot) = (U_{c_l} \circ U_{c_{(l-1)}} \circ ... \circ U_{c_1})(\cdot)$, input x, output $y = U(x)$, gradients $\partial L/\partial y$, and parameters Θ for U.
Output : $\partial L/\partial \Theta$, $\partial L/\partial x$
$\partial L/\partial x \leftarrow \partial L/\partial y$
for $i \leftarrow l$ **to** 1 **do**
  $y_c \leftarrow x_c, \forall c \in c_i$
  $\partial L/\partial y, \partial L/\partial \Theta_{c_i} \leftarrow \text{BP}(y, \partial L/\partial x, U_{c_i}, \Theta_{c_i})$
  $(\partial L/\partial x)_c \leftarrow (\partial L/\partial y)_c, \forall c \in c_i$
  $(\partial L/\partial x)_c \leftarrow (\partial L/\partial x)_c + (\partial L/\partial y)_c, \forall c \notin c_i$
end

where $\mathcal{G}_c$ is a convolutional neural network $\mathcal{G}_c: \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times 1}$. The motivation of expressing $\mathcal{F}$ in a residual learning manner is to reduce overlap singularities (Orhan & Pitkow, 2018), which will be discussed in Section 4.

3.4. Learning GUNN by Backpropagation

Here we show the backpropagation algorithm for learning the parameters in GUNN that uses the same amount of computations and memory as in $\mathcal{F}$. In Eq. 4, let the feature update U_{c_i} be parameterized by Θ_{c_i}. Let $\text{BP}(x, \partial L/\partial y, f, \Theta)$ be the back-propagation algorithm for differentiable function $y = f(x; \Theta)$ with the loss L and the parameters Θ. Algorithm 1 presents the back-propagation algorithm for GUNN. Since U_{c_i} has the residual structures (He et al., 2016a), the last two steps can be merged into

$$(\partial L/\partial x)_c \leftarrow (\partial L/\partial x)_c + (\partial L/\partial y)_c, \ \ \forall c \tag{6}$$

which further simplifies the implementation. It is easy to see that converting networks to GUNN-based does not increase the memory usage in feed-forwarding. Given Algorithm 1, converting networks to GUNN will not affect the memory in both the training and the evaluation.

4. GUNN Eliminates Overlap Singularities

Overlap singularities are inherent in the loss landscapes of some network architectures which are caused by the non-identifiability of subsets of the neurons. They are identified and discussed in previous work (Wei et al., 2008; Anandkumar & Ge, 2016; Orhan & Pitkow, 2018), and are shown to be harmful for the performances of deep networks. Intuitively, overlap singularities exist in architectures where the internal neurons collapse into each other. As a result, the models are degenerate and the effective dimensionality is reduced. (Orhan & Pitkow, 2018) demonstrated through experiments that residual learning (see Eq. 5) helps to reduce the overlap singularities in deep networks, which partly explains the exceptional performances of ResNet (He et al., 2016a) compared with plain networks. In the following, we first use linear transformation as an example to demonstrate

how GUNN-based networks break the overlap singularities. Then, we generalize the results to ReLU DNN. Finally, we compare GUNN with the previous state-of-the-art network architectures from the perspective of singularity elimination.

4.1. Overlap Singularities in Linear Transformations

Consider a linear function $y = f(x) : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$y_i = \sum_{j=1}^{n} \omega_{i,j} x_j, \quad \forall i \in \{1, .., n\} \tag{7}$$

Suppose that there exists a pair of collapsed neurons y_p and y_q ($p < q$). Then, for $\forall x$, $y_p = y_q$, and the equality holds after any number of gradient descents, *i.e.* $\Delta y_p = \Delta y_q$.

Eq. 7 describes a plain network. The solution for the existence of y_p and y_q is that $\omega_{p,j} = \omega_{q,j}, \forall j$. This is the case that is mostly discussed previously, which happens in the networks and degrades the performances.

When we add the residual learning, Eq. 7 becomes

$$y_i = x_i + \sum_{j=1}^{n} \omega_{i,j} x_j, \quad \forall i \in \{1, .., n\} \tag{8}$$

Collapsed neurons require that $\omega_{p,p} + 1 = \omega_{q,p}$, $\omega_{q,q} + 1 = \omega_{p,q}$. This will make the collapse of y_p and y_q very hard when ω is initialized from a normal distribution $\mathcal{N}(0, \sqrt{2/n})$ as in ResNet, but still possible.

Next, we convert Eq. 8 to GUNN, *i.e.,*

$$y_i = x_i + \sum_{j=1}^{i-1} \omega_{i,j} y_j + \sum_{j=i}^{n} \omega_{i,j} x_j, \quad \forall i \in \{1, .., n\} \tag{9}$$

Suppose that y_p and y_q ($p < q$) collapse. Consider Δy, the value difference at x after one step of gradient descent on ω with input x, $\partial L / \partial y$ and learning rate ϵ. When $\epsilon \to 0$,

$$\Delta y_i = \epsilon \frac{\partial L}{\partial y_i} (\sum_{j=1}^{i-1} y_j^2 + \sum_{j=i}^{n} x_j^2) + \sum_{j=1}^{i-1} \omega_{i,j} \Delta y_j \tag{10}$$

As $\Delta y_p = \Delta y_q, \forall x$, we have $\omega_{q,j} = 0, \quad \forall j : p < j < q$. But this condition will be broken in the next update; thus, $q = p + 1$. Then, we derive that $y_p = y_q = 0$. But these will also be broken in the next step of gradient descent optimization. Hence, y_p and y_q cannot collapse into each other. The complete proof can be found in the appendix.

4.2. Overlap Singularities in ReLU DNN

In practice, architectures are usually composed of several linear layers and non-linearity layers. Analyzing all the possible architectures is beyond our scope. Here, we discuss the commonly used ReLU DNN, in which only linear transformations and ReLUs are used by simple layer cascading.

Following the notations in §3, we use $y = \mathcal{G}(x) + x$, in which $\mathcal{G}(x)$ is a ReLU DNN. Note that $\mathcal{G}$ is continuous piecewise linear (PWL) function (Arora et al., 2018), which means that there exists a finite set of polyhedra whose union is $\mathbb{R}^n$, and $\mathcal{G}$ is affine linear over each polyhedron.

Suppose that we convert $\mathcal{G}(x)+x$ to GUNN and there exists a pair of collapsed neurons y_p and y_q ($p < q$). Then, the set of polyhedra for y_p is the same as for y_q. Let $\mathbb{P}$ be a polyhedron for y_p and y_q defined above. Then, $\forall x, \mathbb{P}, i$,

$$y_i = x_i + \sum_{j=1}^{i-1} \omega_{i,j}(\mathbb{P}) y_j + \sum_{j=i}^{n} \omega_{i,j}(\mathbb{P}) x_j \tag{11}$$

where $\omega(\mathbb{P})$ denotes the parameters for polyhedron $\mathbb{P}$. Note that on each $\mathbb{P}$, y is a function of x in the form of Eq. 9; hence, y_p and y_q cannot collapse into each other. Since the union of all polyhedra is $\mathbb{R}^n$, we conclude that GUNN eliminates the overlap singularities in ReLU DNN.

4.3. Discussions and Comparisons

The previous two subsections consider the GUNN conversion where $|c_i| = 1, \forall i$ (see Eq. 4). But this will slow down the computation on GPU due to the data dependency. Without specialized hardware or library support, we decide to increase $|c_i|$ to > 10. The resulted models run at the speed between ResNeXt (Xie et al., 2017) and DenseNet (Huang et al., 2017b). But this change introduces singularities into the channels from the same set c_i. Then, the residual learning helps GUNN to reduce the singularities within the same set c_i since we initialize the parameters from a normal distribution $\mathcal{N}(0, \sqrt{2/n})$. We will compare the results of GUNN with and without residual learning in the experiments.

We compare GUNN with the state-of-the-art architectures from the perspective of overlap singularities. ResNet (He et al., 2016a) and its variants use residual learning, which reduces but cannot eliminate the singularities. ResNeXt (Xie et al., 2017) uses group convolutions to break the symmetry between groups, which further helps to avoid neuron collapses. DenseNet (Huang et al., 2017b) concatenates the outputs of layers as the input to the next layer. DenseNet and GUNN both create dense connections, while DenseNet reuses the outputs by concatenating and GUNN by adding them back to the inputs. But the channels within the same layer of DenseNet are still possible to collapse into each other since they are symmetric. In contrast, adding back makes residual learning possible in GUNN. This makes residual learning indispensable in GUNN-based networks.

5. Network Architectures

In this section, we will present the details of our architectures for the CIFAR (Krizhevsky & Hinton, 2009) and ImageNet (Russakovsky et al., 2015) datasets.

5.1. Simultaneously Updated Neural Networks and Gradually Updated Neural Networks

Since the proposed GUNN is a method for increasing the depths of the convolutional networks, specifying the architectures to be converted is equivalent to specifying the GUNN-based architectures. The architectures before conversion, the Simultaneously Updated Neural Networks (SUNN), become natural baselines for our proposed GUNN networks. We first study what baseline architectures can be converted.

There are two assumptions about the feature transformation $\mathcal{F}$ (see Eq. 1): (1) the input and the output sizes are the same, and (2) $\mathcal{F}$ is channel-wise decomposable. To satisfy the first assumption, we will first use a convolutional layer with Batch Normalization (Ioffe & Szegedy, 2015) and ReLU (Nair & Hinton, 2010) to transform the feature space to a new space where the number of the channels is wanted. To satisfy the second assumption, instead of directly specifying the transform $\mathcal{F}$, we focus on designing $\mathcal{F}_{c_i}$, where c_i is a subset of the channels (see Eq. 4). To be consistent with the term *update* used in GUNN and SUNN, we refer to $\mathcal{F}_{c_i}$ as the *update units* for channels c_i.

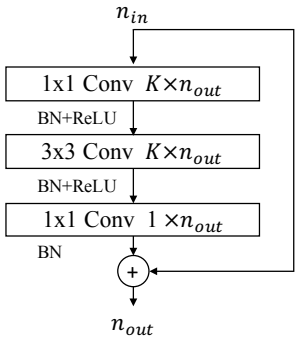

Figure 4. Bottleneck Update Units for both SUNN and GUNN.

Bottleneck Update Units In the architectures proposed in this paper, we adopt bottleneck neural networks as shown in Figure 4 for the update units for both the SUNN and GUNN. Suppose that the update unit maps the input features of channel size n_{in} to the output features of size n_{out}. Each unit contains three convolutional layers. The first convolutional layer transforms the input features to $K \times n_{\text{out}}$ using a 1×1 convolutional layer. The second convolutional layer is of kernel size 3×3, stride 1, and padding 1, outputting the features of size $K \times n_{\text{out}}$. The third layer computes the features of size n_{out} using a 1×1 convolutional layer. The output is then added back to the input, following the residual architecture proposed in ResNet (He et al., 2016a). We add batch normalization layer (Ioffe & Szegedy, 2015) and ReLU layer (Nair & Hinton, 2010) after the first and the second convolutional layers, while only adding batch normalization layer after the third layer. Stacking up M update units also generates a new one. In total, we have two hyperparameters for designing an update unit: the expansion rate K and the number of the 3-layer update units M.

One Resolution, One Representation Our architectures will have only one representation at one resolution besides the pooling layers and the convolutional layers that initialize the needed numbers of channels. Take the architecture in Table 1 as an example. There are two processes for each resolution. The first one is the transition process, which computes the initial features with the dimensions of the next resolution, then down samples it to $1/4$ using a 2×2 average pooling. A convolutional operation is needed here because $\mathcal{F}$ is assumed to have the same input and output sizes. The next process is using GUNN to update this feature space gradually. Each channel will only be updated once, and all channels will be updated after this process. Unlike most of the previous networks, after this two processes, the feature transformations at this resolution are complete. There will be no more convolutional layers or blocks following this feature representation, *i.e.*, *one resolution, one representation*. Then, the network will compute the initial features for the next resolution, or compute the final vector representation of the entire image by a global average pooling. By designing networks in this way, SUNN networks usually have about 20 layers before converting to GUNN-based networks.

Channel Partitions With the clearly defined update units, we can easily build SUNN and GUNN layers by using the units to update the representations following Eq. 4. The hyperparameters for the SUNN/GUNN layer are the number of the channels N and the partition over those channels. In our proposed architectures, we evenly partition the channels into P segments. Then, we can use N and P to represent the configuration of a layer. Together with the hyperparameters in the update units, we have four hyperparameters to tune for one SUNN/GUNN layer, *i.e.* $\{N, P, K, M\}$.

5.2. Architectures for CIFAR

We have implemented two neural networks based on GUNN to compete with the previous state-of-the-art methods on CIFAR datasets, *i.e.*, GUNN-15 and GUNN-24. Table 1 shows the big picture of GUNN-15. Here, we present the details of the hyperparameter settings for GUNN-15 and GUNN-24. For GUNN-15, we have three GUNN layers, Conv2, Conv3 and Conv4. The configuration for Conv2 is $\{N = 240, P = 20, K = 2, M = 1\}$, the configuration for Conv3 is $\{N = 300, P = 25, K = 2, M = 1\}$, and the configuration for Conv4 is $\{N = 360, P = 30, K = 2, M = 1\}$. For GUNN-24, based on GUNN-15, we change the number of output channels of Conv1 to 720, Trans1 to 900, Trans2 to 1080, and Trans3 to 1080. The hyperparameters are $\{N = 720, P = 20, K = 3, M = 2\}$ for

Stage	Output	WideResNet-28-10	GUNN-15
Conv1	32×32	$[3 \times 3, 16]$	$\begin{bmatrix} 3 \times 3, 64 \\ 1 \times 1, 240 \end{bmatrix}$
Conv2*	32×32	$\begin{bmatrix} 3 \times 3, 160 \\ 3 \times 3, 160 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, \times 2 \\ 3 \times 3, \times 2 \\ 1 \times 1, \times 1 \end{bmatrix}$
Trans1	16×16	–	$\begin{bmatrix} 1 \times 1, 300 \\ \text{AvgPool} \end{bmatrix}$
Conv3*	16×16	$\begin{bmatrix} 3 \times 3, 320 \\ 3 \times 3, 320 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, \times 2 \\ 3 \times 3, \times 2 \\ 1 \times 1, \times 1 \end{bmatrix}$
Trans2	8×8	–	$\begin{bmatrix} 1 \times 1, 360 \\ \text{AvgPool} \end{bmatrix}$
Conv4*	8×8	$\begin{bmatrix} 3 \times 3, 640 \\ 3 \times 3, 640 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, \times 2 \\ 3 \times 3, \times 2 \\ 1 \times 1, \times 1 \end{bmatrix}$
Trans3	1×1	$\begin{bmatrix} \text{GAPool} \\ \text{fc, softmax} \end{bmatrix}$	$\begin{bmatrix} 1 \times 1, 360 \\ \text{GAPool} \\ \text{fc, softmax} \end{bmatrix}$
GPU Memory		4.903GB@64	2.485GB@64
# Params		36.5M	1.6M
Error (C10/C100)		4.17 / 20.50	4.15 / 20.45

Table 1. Architecture comparison between WideResNet-28-10 (Zagoruyko & Komodakis, 2016) and GUNN-15 for CIFAR. (Left) WideResNet-28-10. (Right) GUNN-15. GUNN achieves comparable accuracies on CIFAR10/100 while using a smaller number of parameters and consuming less GPU memory during training. In GUNN-15, the convolution stages with stars are computed using GUNN while others are not.

Stage	Output	ResNet-152	GUNN-18
Conv1	112×112	$\begin{bmatrix} 7 \times 7, 64, 2 \\ 3 \times 3 \text{ MaxPool}, 2 \end{bmatrix}$	$\begin{bmatrix} 7 \times 7, 64, 2 \\ 3 \times 3 \text{ MaxPool}, 2 \\ 1 \times 1, 400 \end{bmatrix}$
Conv2*	112×112	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, \times 2 \\ 3 \times 3, \times 2 \\ 1 \times 1, \times 1 \end{bmatrix}$
Trans1	56×56	–	$\begin{bmatrix} 1 \times 1, 800 \\ \text{AvgPool} \end{bmatrix}$
Conv3*	56×56	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$	$\begin{bmatrix} 1 \times 1, \times 2 \\ 3 \times 3, \times 2 \\ 1 \times 1, \times 1 \end{bmatrix}$
Trans2	28×28	–	$\begin{bmatrix} 1 \times 1, 1600 \\ \text{AvgPool} \end{bmatrix}$
Conv4*	28×28	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$	$\begin{bmatrix} 1 \times 1, \times 2 \\ 3 \times 3, \times 2 \\ 1 \times 1, \times 1 \end{bmatrix}$
Trans3	14×14	–	$\begin{bmatrix} 1 \times 1, 2000 \\ \text{AvgPool} \end{bmatrix}$
Conv5*	14×14	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, \times 2 \\ 3 \times 3, \times 2 \\ 1 \times 1, \times 1 \end{bmatrix}$
Trans4	$\times 1$	$\begin{bmatrix} \text{GAPool} \\ \text{fc, softmax} \end{bmatrix}$	$\begin{bmatrix} \text{GAPool} \\ \text{fc, softmax} \end{bmatrix}$
# Params		60.2M	28.9M
Error (Top-1/5)		22.2 / 6.2	21.65 / 5.87

Table 2. Architecture comparison between ResNet (He et al., 2016a) and GUNN-18 for ImageNet-152. (Left) ResNet-152. (Right) GUNN-18. GUNN achieves better accuracies on ImageNet while using a smaller number of parameters.

Conv2, $\{N = 900, P = 25, K = 3, M = 2\}$ for Conv3, and $\{N = 1080, P = 30, K = 3, M = 2\}$ for Conv3. The number of parameters of GUNN-15 is 1585746 for CIFAR-10 and 1618236 for CIFAR-100. The number of parameters of GUNN-24 is 29534106 for CIFAR-10 and 29631396 for CIFAR-100. The GUNN-15 is aimed to compete with the methods published in an early stage by using a much smaller model, while GUNN-24 is targeted at comparing with ResNeXt (Xie et al., 2017) and DenseNet (Huang et al., 2017b) to get the state-of-the-art performance.

5.3. Architectures for ImageNet

We implement a neural network GUNN-18 to compete with the state-of-the-art neural networks on ImageNet with a similar number of parameters. Table 2 shows the big picture of the neural network architecture of GUNN-18. Here, we present the detailed hyperparameters for the GUNN layers in GUNN-18. The GUNN layers include Conv2, Conv3, Conv4 and Conv5. The hyperparameters are $\{N = 400, P = 10, K = 2, M = 1\}$ for Conv2, $\{N = 800, P = 20, K = 2, M = 1\}$ for Conv3, $\{N = 1600, P = 40, K = 2, M = 1\}$ for Conv4 and $\{N = 2000, P = 50, K = 2, M = 1\}$ for Conv5. The number of parameters is 28909736. The GUNN-18 is targeted at competing with the previous state-of-the-art methods that have similar numbers of parameters, *e.g.*, ResNet-50 (Xie et al., 2017), ResNeXt-50 (Xie et al., 2017) and DenseNet-264 (Huang et al., 2017b).

We also implement a wider GUNN-based neural networks *Wide-GUNN-18* for better capacities. The hyperparameters are $\{N = 1200, P = 30, K = 2, M = 1\}$ for Conv2, $\{N = 1600, P = 40, K = 2, M = 1\}$ for Conv3, $\{N = 2000, P = 50, K = 2, M = 1\}$ for Conv4 and $\{N = 2000, P = 50, K = 2, M = 1\}$ for Conv5. The number of parameters is 45624936. The Wide-GUNN-18 is targeted at competing with ResNet-101, ResNext-101, DPN (Chen et al., 2017) and SENet (Hu et al., 2017).

Method	# layers	# params	C10	C100
Network in Network (Lin et al., 2013)			8.81	–
All-CNN (Springenberg et al., 2014)			7.25	33.71
Deeply Supervised Network (Lee et al., 2014)			7.97	34.57
Highway Network (Srivastava et al., 2015)			7.72	32.39
ResNet (He et al., 2016a; Huang et al., 2016)	110	1.7M	6.41	27.22
FractalNet (Larsson et al., 2016)	21	38.6M	5.22	23.30
Stochastic Depth (Huang et al., 2016)	1202	10.2M	4.91	24.58
ResNet with pre-act (He et al., 2016b)	1001	10.2M	4.62	22.71
WideResNet-28-10 (Zagoruyko & Komodakis, 2016)	28	36.5M	4.17	20.50
WideResNet-40-10 (Zagoruyko & Komodakis, 2016)	40	55.8M	3.80	18.30
ResNeXt (Xie et al., 2017)	29	68.1M	3.58	17.31
DenseNet (Huang et al., 2017b)	190	25.6M	3.46	17.18
Snapshot Ensemble (Huang et al., 2017a)	–	163.2M	3.44	17.41
GUNN-15	15	1.6M	4.15	20.45
GUNN-24	24	29.6M	**3.21**	**16.69**
GUNN-24 Ensemble	24×6	177.6M	**3.02**	**15.61**

Table 3. Classification errors (%) on the CIFAR-10/100 test set. All methods are with data augmentation. The third group shows the most recent state-of-the-art methods. The performances of GUNN are presented in the fourth group. A very small model GUNN-15 outperforms all the methods in the second group except WideResNet-40-10. A relatively bigger model GUNN-24 surpasses all the competing methods. GUNN-24 becomes more powerful with ensemble (Huang et al., 2017a).

6. Experiments

In this section, we demonstrate the effectiveness of the proposed GUNN on several benchmark datasets.

6.1. Benchmark Datasets

CIFAR CIFAR (Krizhevsky & Hinton, 2009) has two color image datasets: CIFAR-10 (C10) and CIFAR-100 (C100). Both datasets consist of natural images with the size of 32×32 pixels. The CIFAR-10 dataset has 10 categories, while the CIFAR-100 dataset has 100 categories. For both of the datasets, the training and test set contain $50,000$ and $10,000$ images, respectively. To fairly compare our method with the state-of-the-arts (He et al., 2016a; Huang et al., 2017b; 2016; Larsson et al., 2016; Lee et al., 2014; Lin et al., 2013; Romero et al., 2014; Springenberg et al., 2014; Srivastava et al., 2015; Xie et al., 2017), we use the same training and testing strategies, as well as the data processing methods. Specifically, we adopt a commonly used data augmentation scheme, *i.e.,* mirroring and shifting, for these two datasets. We use channel means and standard derivations to normalize the images for data pre-processing.

ImageNet The ImageNet dataset (Russakovsky et al., 2015) contains about 1.28 million color images for training and $50,000$ for validation. The dataset has 1000 categories. We adopt the same data augmentation methods as in the state-of-the-art architectures (He et al., 2016a;b; Huang et al., 2017b; Xie et al., 2017) for training. For testing, we use single-crop at the size of 224×224. Following the state-of-the-arts (He et al., 2016a;b; Huang et al., 2017b; Xie et al., 2017), we report the validation error rates.

6.2. Training Details

We train all of our networks using stochastic gradient descents. On CIFAR-10/100 (Krizhevsky & Hinton, 2009), the initial learning rate is set to 0.1, the weight decay is set to $1e^{-4}$, and the momentum is set to 0.9 without dampening. We train the models for 300 epochs. The learning rate is divided by 10 at 150th epoch and 225th epoch. We set the batch size to 64, following (Huang et al., 2017b). All the results reported for CIFAR, regardless of the detailed configurations, were trained using 4 NVIDIA Titan X GPUs with the data parallelism. On ImageNet (Russakovsky et al., 2015), the learning rate is also set to 0.1 initially, and decreases following the schedule in DenseNet (Huang et al., 2017b). The batch size is set to 256. The network parameters are also initialized following (He et al., 2016a). We use 8 Tesla V100 GPUs with the data parallelism to get the reported results. Our results are directly comparable with ResNet, WideResNet, ResNeXt and DenseNet.

6.3. Results on CIFAR

We train two models GUNN-15 and GUNN-24 for the CIFAR-10/100 dataset. Table 3 shows the comparisons between our method and the previous state-of-the-art methods. Our method GUNN achieves the best results in the test of both the single model and the ensemble test. Here, we use Snapshot Ensemble (Huang et al., 2017a).

Method	# layers	# params	top-1	top-5
VGG-16 (Simonyan & Zisserman, 2014)	16	138M	28.5	9.9
ResNet-50 (He et al., 2016a)	50	25.6M	24.0	7.0
ResNeXt-50 (Xie et al., 2017)	50	25.0M	22.2	6.0
DenseNet-264 (Huang et al., 2017b)	264	33.3M	22.15	6.12
SUNN-18*	18	28.9M	26.16	8.48
GUNN-18*	18	28.9M	**21.65**	**5.87**
ResNet-101 (He et al., 2016a)	101	44.5M	22.0	6.0
ResNeXt-101 (Xie et al., 2017)	101	44.1M	21.2	5.6
DPN-98 (Chen et al., 2017)	98	37.7M	20.73	5.37
SE-ResNeXt-101 (Hu et al., 2017)	101	49.0M	20.70	**5.01**
Wide GUNN-18*	18	45.6M	**20.59**	5.52

Table 4. Single-crop classification errors (%) on the ImageNet validation set. The test size of all the methods is 224×224. Ours: $*$.

Method	# layers	# params	C10	C100
GUNN-15-NoRes	15	1.6M	4.45	21.15
GUNN-15	15	1.6M	4.15	20.45
SUNN-15	15	1.6M	5.64	23.75
GUNN-15	15	1.6M	4.15	20.45
SUNN-24	24	29.6M	3.88	19.60
GUNN-24	24	29.6M	3.21	16.69

Table 5. Ablation study on residual learning and SUNN.

Baseline Methods Here we present the details of baseline methods in Table 3. The performances of ResNet (He et al., 2016a) are reported in Stochastic Depth (Huang et al., 2016) for both C10 and C100. The WideResNet (Zagoruyko & Komodakis, 2016) WRN-40-10 is reported in their official code repository on GitHub. The ResNeXt in the third group is of configuration 16×64d, which has the best result reported in the paper (Xie et al., 2017). The DenseNet is of configuration DenseNet-BC ($k = 40$), which achieves the best performances on CIFAR-10/100. The Snapshot Ensemble (Huang et al., 2017a) uses 6 DenseNet-100 to ensemble during inference. We do not compare with methods that use more data augmentation (*e.g.* (Zhang et al., 2017)) or stronger regularizations (*e.g.* (Gastaldi, 2017)) for the fairness of comparison.

Ablation Study For ablation study, we compare GUNN with SUNN, *i.e.*, the networks before the conversion. Table 5 shows the comparison results, which demonstrate the effectiveness of GUNN. We also compare the performances of GUNN with and without residual learning.

6.4. Results on ImageNet

We evaluate the GUNN on the ImageNet classification task, and compare our performances with the state-of-the-art methods. These methods include VGGNet (Simonyan & Zisserman, 2014), ResNet (He et al., 2016a), ResNeXt (Xie et al., 2017), DenseNet (Huang et al., 2017b), DPN (Chen et al., 2017) and SENet (Hu et al., 2017). The comparisons are shown in Table 4. The results of ours, ResNeXt, and DenseNet are directly comparable as these methods use the same framework for training and testing networks. Table 4 groups the methods by their numbers of parameters, except VGGNet which has 1.38×10^8 parameters.

The results presented in Table 4 demonstrate that with the similar number of parameters, GUNN can achieve comparable performances with the previous state-of-the-art methods. For GUNN-18, we also conduct an ablation experiment by comparing the corresponding SUNN with GUNN of the same configuration. Consistent with the experimental results on the CIFAR-10/100 dataset, the proposed GUNN improves the accuracy on ImageNet dataset.

7. Conclusions

In this paper, we propose Gradually Updated Neural Network (GUNN), a novel, simple yet effective method to increase the depths of neural networks as an alternative to cascading layers. GUNN is based on Convolutional Neural Networks (CNNs), but differs from CNNs in the way of computing outputs. The outputs of GUNN are computed gradually rather than simultaneously as in CNNs in order to increase the depth. Essentially, GUNN assumes the input and the output are of the same size and adds a computation ordering to the channels. The added ordering increases the receptive fields and non-linearities of the later computed channels. Moreover, it eliminates the overlap singularities inherent in the traditional convolutional networks. We test GUNN on the task of image recognition. The evaluations are done in three highly competitive benchmarks, CIFAR-10, CIFAR-100 and ImageNet. The experimental results demonstrate the effectiveness of the proposed GUNN on image recognition. In the future, since the proposed GUNN can be used to replace CNNs in other neural networks, we will study the applications of GUNN in other visual tasks, such as object detection and semantic segmentation.

References

Anandkumar, Animashree and Ge, Rong. Efficient approaches for escaping higher order saddle points in non-convex optimization. In *Proceedings of the Conference on Learning Theory*, 2016.

Arora, Raman, Basu, Amitabh, Mianjy, Poorya, and Mukherjee, Anirbit. Understanding deep neural networks with rectified linear units. *International Conference on Learning Representations*, 2018.

Chen, Liang-Chieh, Papandreou, George, Kokkinos, Iasonas, Murphy, Kevin, and Yuille, Alan L. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *International Conference on Learning Representations*, 2015.

Chen, Yunpeng, Li, Jianan, Xiao, Huaxin, Jin, Xiaojie, Yan, Shuicheng, and Feng, Jiashi. Dual path networks. *CoRR*, abs/1707.01629, 2017.

Dalal, N. and Triggs, B. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2005.

Gastaldi, Xavier. Shake-shake regularization. *CoRR*, abs/1705.07485, 2017.

Girshick, Ross B., Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEEConference on Computer Vision and Pattern Recognition*, 2014.

Hariharan, Bharath, Arbeláez, Pablo Andrés, Girshick, Ross B., and Malik, Jitendra. Hypercolumns for object segmentation and fine-grained localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 447–456, 2015.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016a.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. *ECCV*, 2016b.

Hu, Jie, Shen, Li, and Sun, Gang. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.

Huang, Gao, Sun, Yu, Liu, Zhuang, Sedra, Daniel, and Weinberger, Kilian Q. Deep networks with stochastic depth. *CoRR*, abs/1603.09382, 2016.

Huang, Gao, Li, Yixuan, Pleiss, Geoff, Liu, Zhuang, Hopcroft, John E., and Weinberger, Kilian Q. Snapshot ensembles: Train 1, get M for free. *CoRR*, abs/1704.00109, 2017a.

Huang, Gao, Liu, Zhuang, and Weinberger, Kilian Q. Densely connected convolutional networks. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017b.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

Kontschieder, Peter, Fiterau, Madalina, Criminisi, Antonio, and Bulò, Samuel Rota. Deep neural decision forests. In *IEEE International Conference on Computer Vision*, pp. 1467–1475, 2015.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, pp. 1097–1105. 2012.

Larsson, Gustav, Maire, Michael, and Shakhnarovich, Gregory. Fractalnet: Ultra-deep neural networks without residuals. *CoRR*, abs/1605.07648, 2016.

Lee, Chen-Yu, Xie, Saining, Gallagher, Patrick W., Zhang, Zhengyou, and Tu, Zhuowen. Deeply-supervised nets. *CoRR*, abs/1409.5185, 2014.

Lin, Min, Chen, Qiang, and Yan, Shuicheng. Network in network. *CoRR*, abs/1312.4400, 2013.

Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Lowe, David G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004. ISSN 1573-1405.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, 2010.

Orhan, A. Emin and Pitkow, Xaq. Skip connections eliminate singularities. *International Conference on Learning Representations*, 2018.

Pezeshki, Mohammad, Fan, Linxi, Brakel, Philemon, Courville, Aaron C., and Bengio, Yoshua. Deconstructing the ladder network architecture. In *International Conference on Machine Learning*, pp. 2368–2376, 2016.

Qiao, Siyuan, Liu, Chenxi, Shen, Wei, and Yuille, Alan L. Few-shot image recognition by predicting parameters from activations. *CoRR*, abs/1706.03466, 2017a.

Qiao, Siyuan, Shen, Wei, Qiu, Weichao, Liu, Chenxi, and Yuille, Alan L. Scalenet: Guiding object proposal generation in supermarkets and beyond. In *IEEE International Conference on Computer Vision*, 2017b.

Rasmus, Antti, Berglund, Mathias, Honkala, Mikko, Valpola, Harri, and Raiko, Tapani. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pp. 3546–3554, 2015.

Ren, Shaoqing, He, Kaiming, Girshick, Ross B., and Sun, Jian. Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28*, 2015.

Romero, Adriana, Ballas, Nicolas, Kahou, Samira Ebrahimi, Chassang, Antoine, Gatta, Carlo, and Bengio, Yoshua. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2014.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Sermanet, Pierre, Eigen, David, Zhang, Xiang, Mathieu, Michal, Fergus, Robert, and Lecun, Yann. Overfeat: Integrated recognition, localization and detection using convolutional networks. *International Conference on Learning Representations*, 2014.

Shen, Wei, Wang, Xinggang, Wang, Yan, Bai, Xiang, and Zhang, Zhijiang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Springenberg, Jost Tobias, Dosovitskiy, Alexey, Brox, Thomas, and Riedmiller, Martin A. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.

Srivastava, Rupesh Kumar, Greff, Klaus, and Schmidhuber, Jürgen. Training very deep networks. *CoRR*, abs/1507.06228, 2015.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. In *Computer Vision and Pattern Recognition*, 2015.

Szegedy, Christian, Vanhoucke, Vincent, Ioffe, Sergey, Shlens, Jonathon, and Wojna, Zbigniew. Rethinking the inception architecture for computer vision. In *Computer Vision and Pattern Recognition*, 2016.

Wang, Yan, Xie, Lingxi, Liu, Chenxi, Qiao, Siyuan, Zhang, Ya, Zhang, Wenjun, Tian, Q., and Yuille, Alan. SORT: Second-Order Response Transform for Visual Recognition. *International Conference on Computer Vision*, 2017.

Wei, Haikun, Zhang, Jun, Cousseau, Florent, Ozeki, Tomoko, and Amari, Shun-ichi. Dynamics of learning near singularities in layered networks. *Neural Computation*, 20(3):813–843, 2008.

Xie, Saining and Tu, Zhuowen. Holistically-nested edge detection. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, 2015.

Xie, Saining, Girshick, Ross B., Dollár, Piotr, Tu, Zhuowen, and He, Kaiming. Aggregated residual transformations for deep neural networks. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Yang, Songfan and Ramanan, Deva. Multi-scale recognition with dag-cnns. In *IEEE International Conference on Computer Vision*, pp. 1215–1223, 2015.

Zagoruyko, Sergey and Komodakis, Nikos. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

Zeiler, Matthew D. and Fergus, Rob. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

Zhang, Hongyi, Cissé, Moustapha, Dauphin, Yann N., and Lopez-Paz, David. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017.