Convolutional Networks with Adaptive Inference Graphs

Andreas Veit Serge Belongie

Department of Computer Science & Cornell Tech, Cornell University, New York
{av443,sjb344}@cornell.edu

Abstract. Do convolutional networks really need a fixed feed-forward structure? What if, after identifying the high-level concept of an image, a network could move directly to a layer that can distinguish fine-grained differences? Currently, a network would first need to execute sometimes hundreds of intermediate layers that specialize in unrelated aspects. Ideally, the more a network already knows about an image, the better it should be at deciding which layer to compute next. In this work, we propose convolutional networks with adaptive inference graphs (ConvNet-AIG) that adaptively define their network topology conditioned on the input image. Following a high-level structure similar to residual networks (ResNets), ConvNet-AIG decides for each input image on the fly which layers are needed. In experiments on ImageNet we show that ConvNet-AIG learns distinct inference graphs for different categories. Both ConvNet-AIG with 50 and 101 layers outperform their ResNet counterpart, while using 20% and 33% less computations respectively. By grouping parameters into layers for related classes and only executing relevant layers, ConvNet-AIG improves both efficiency and overall classification quality. Lastly, we also study the effect of adaptive inference graphs on the susceptibility towards adversarial examples. We observe that ConvNet-AIG shows a higher robustness than ResNets, complementing other known defense mechanisms.

1 Introduction

Often, convolutional networks (ConvNets) are already confident about the high-level concept of an image after only a few layers. This raises the question of what happens in the remainder of the network that often comprises hundreds of layers for many state-of-the-art models. To shed light on this, it is important to note that due to their success, ConvNets are used to classify increasingly large sets of visually diverse categories. Thus, most parameters model high-level features that, in contrast to low-level and many mid-level concepts, cannot be broadly shared across categories. As a result, the networks become larger and slower as the number of categories rises. Moreover, for any given input image the number of computed features focusing on unrelated concepts increases.

What if, after identifying that an image contains a bird, a ConvNet could move directly to a layer that can distinguish different bird species, without executing intermediate layers that specialize in unrelated aspects? Intuitively, the

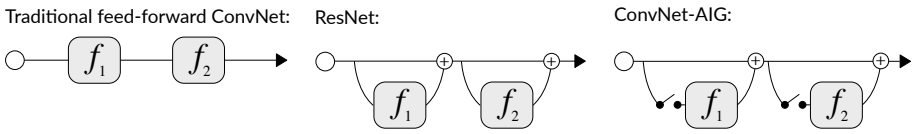

Fig. 1. ConvNet-AIG (right) follows a high level structure similar to ResNets (center) by introducing identity skip-connections that bypass each layer. The key difference is that for each layer, a gate determines whether to execute or skip the layer. This enables individual inference graphs conditioned on the input.

more the network already knows about an image, the better it could be at deciding which layer to compute next. This shares resemblance with decision trees that employ information theoretic approaches to select the most informative features to evaluate. Such a network could decouple inference time from the number of learned concepts. A recent study [31] provides a key insight towards the realization of this scenario. The authors study residual networks (ResNets) [11] and show that almost any individual layer can be removed from a trained ResNet without interfering with other layers. This leads us to the following research question: *Do we really need fixed structures for convolutional networks, or could we assemble network graphs on the fly, conditioned on the input?*

In this work, we propose ConvNet-AIG, a convolutional network that adaptively defines its inference graph conditioned on the input image. Specifically, ConvNet-AIG learns a set of convolutional layers and decides for each input image which layers are needed. By learning both general layers useful to all images and expert layers specializing on subsets of categories, it allows to only compute features relevant to the input image. It is worthy to note that ConvNet-AIG does not require special supervision about label hierarchies and relationships.

Figure 1 gives an overview of our approach. ConvNet-AIG follows a structure similar to a ResNet. The key difference is that for each residual layer, a gate determines whether the layer is needed for the current input image. The main technical challenge is that the gates need to make discrete decisions, which are difficult to integrate into convolutional networks that we would like to train using gradient descent. To incorporate the discrete decisions, we build upon recent work [4, 18, 24] that introduces differentiable approximations for discrete stochastic nodes in neural networks. In particular, we model the gates as discrete random variables over two states: to execute the respective layer or to skip it. Further, we model the gates conditional on the output of the previous layer. This allows to construct inference graphs adaptively based on the input and to train both the convolutional weights and the discrete gates jointly end-to-end.

In experiments on ImageNet [5], we demonstrate that ConvNet-AIG effectively learns to generate inference graphs such that for each input only relevant features are computed. In terms of accuracy both ConvNet-AIG 50 and ConvNet-AIG 101 outperform their ResNet counterpart, while at the same time using 20% and 33% less computations. We further show that, without specific supervision,

ConvNet-AIG discovers parts of the class hierarchy and learns specialized layers focusing on subsets of categories such as animals and man-made objects. It even learns distinct inference graphs for some mid-level categories such as birds, dogs and reptiles. By grouping parameters for related classes and only executing relevant layers, ConvNet-AIG both improves efficiency and overall classification quality. Lastly, we also study the effect of adaptive inference graphs on susceptibility towards adversarial examples. We show that ConvNet-AIG is consistently more robust than ResNets, independent of adversary strength and that the additional robustness persists even when applying additional defense mechanisms.

2 Related Work

Our study is related to work in multiple fields. Several works have focused on **neural network composition** for visual question answering (VQA) [1, 2, 19] and zero-shot learning [25]. While these approaches include convolutional networks, they focus on constructing a fixed computational graph up front to solve tasks such as VQA. In contrast, the focus of our work is to construct a convolutional network conditioned on the input image on the fly during execution.

Our approach can be seen as an example of **adaptive computation** for neural networks. Cascaded classifiers [32] have a long tradition for computer vision by quickly rejecting "easy" negatives. Recently, similar approaches have been proposed for neural networks [22, 33]. In an alternative direction, [3, 26] propose to adjust the amount of computation in fully-connected neural networks. To adapt computation time in convolutional networks, [14, 30] propose architectures that add classification branches to intermediate layers. This allows stopping a computation early once a satisfying level of confidence is reached. Most closely related to our approach is the work on spatially adaptive computation time for residual networks [6]. In that paper, a ResNet adaptively determines after which layer to stop computation. Our work differs from this approach in that we do not perform early stopping, but instead determine which subset of layers to execute. This is key as it allows the grouping of parameters that are relevant for similar categories and thus enables distinct inference graphs for different categories.

Our work is further related to network **regularization with stochastic noise**. By randomly dropping neurons during training, Dropout [27] offers an effective way to prevent neural networks from over-fitting. Closely related is the work on stochastic depth [16], where entire layers of a ResNet are randomly removed during each training iteration. Our work resembles this approach in that it also includes stochastic nodes that decide whether to execute layers. However, in contrast to our work, layer removal in stochastic depth is independent from the input and aims to *increase* redundancy among layers. In our work, we construct the inference graph conditioned on the input image to *reduce* redundancy and allow the network to learn layers specialized on subsets of the data.

Lastly, our work can also be seen as an example of an **attention mechanism** in that we select specific layers of importance for each input image to assemble the inference graph. This is related to approaches such as highway networks [28]

and squeeze-and-excitation networks [13] where the output of a residual layer is rescaled according to the layer's importance. This allows these approaches to emphasize some layers and pay less attention to others. In contrast to our work, these are soft attention mechanisms and still require the execution of every single layer. Our work is a hard attention mechanism and thus enables decoupling computation time from the number of categories.

3 Adaptive Inference Graphs

Traditional feed-forward ConvNets can be considered as a set of N layers which are sequentially applied to an input image. Formally, let $\mathcal{F}_l(\cdot)$, $l \in \{1, ..., N\}$ denote the function computed by the l^{th} layer. With $\mathbf{x}_0$ as input image and $\mathbf{x}_l$ as output of the l^{th} layer, such a network can be recursively defined as

$$\mathbf{x}_l = \mathcal{F}_l(\mathbf{x}_{l-1}) \tag{1}$$

ResNets [11] change this definition by introducing identity skip-connections that bypass each layer, i.e., the input to each layer is also added to its output. This has been shown to greatly ease optimization during training. As gradients can propagate directly through the skip-connection, early layers still receive sufficient learning signal even in very deep networks. A ResNet can be defined as

$$\mathbf{x}_l = \mathbf{x}_{l-1} + \mathcal{F}_l\left(\mathbf{x}_{l-1}\right) \tag{2}$$

In a follow-up study [31] on the effects of the skip-connection, it has been shown that, although all layers are trained jointly, they exhibit a high degree of independence. Further, almost any individual layer can be removed from a trained ResNet without harming performance and interfering with other layers.

3.1 Gated Inference

Inspired by the observations in [31], we design ConvNet-AIG, a network that can define its topology on the fly. The architecture follows the basic structure of a ResNet with the key difference that instead of executing all layers, the network determines for each input image which subset of layers to execute. In particular, with layers focusing on different subgroups of categories, it can select only those layers necessary for the specific input. A ConvNet-AIG can be defined as

$$\begin{aligned} &\mathbf{x}_l = \mathbf{x}_{l-1} + z(\mathbf{x}_{l-1}) \cdot \mathcal{F}_l\left(\mathbf{x}_{l-1}\right) \\ &\text{where } z(\mathbf{x}_{l-1}) \in \{0, 1\} \end{aligned} \tag{3}$$

where $z(\mathbf{x}_{l-1})$ is a gate that, conditioned on the input to the layer, decides whether to execute the next layer. The gate chooses between two discrete states: 0 for 'off' and 1 for 'on', which can be seen as a *hard attention mechanism*.

For the gate to be effective, it needs to address a few key challenges. First, to estimate the relevance of its layer, the gate needs to understand its input

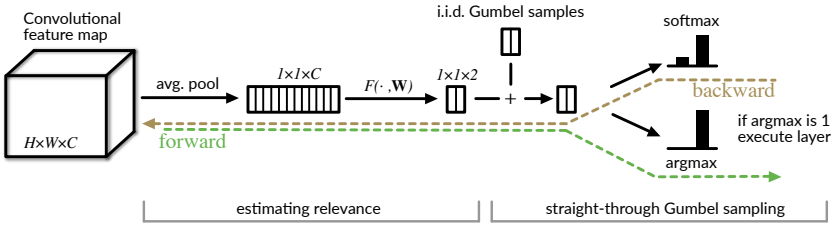

Fig. 2. Overview of gating unit. Each gate comprises two parts. The first part estimates the relevance of the layer to be executed. The second part decides whether to execute the layer given the estimated relevance. In particular, the Gumbel-Max trick and its softmax relaxation are used to allow for the propagation of gradients through the discrete decision.

features. To prevent mode collapse into trivial solutions that are independent of the input features, such as always or never executing a layer, we found it to be of key importance for the gate to be stochastic. We achieve this by adding noise to the estimated relevance. Second, the gate needs to make a discrete decision, while still providing gradients for the relevance estimation. We achieve this with the Gumbel-Max trick and its softmax relaxation. Third, the gate needs to operate with low computational cost. Figure 2 provides and overview of the two key components of the proposed gate. The first one efficiently estimates the relevance of the respective layer for the current image. The second component makes a discrete decision by sampling using Gumbel-Softmax [18, 24].

3.2 Estimating Layer Relevance

The goal of the gate's first component is to estimate its layer's relevance given the input features. The input to the gate is the output of the previous layer $\mathbf{x}_{l-1} \in \mathbb{R}^{W \times H \times C}$. Since operating on the full feature map is computationally expensive, we build upon recent studies [13, 17, 23] which show that much of the information in convolutional features is captured by the statistics of the different channels and their interdependencies. In particular, we only consider channel-wise means gathered by global average pooling. This compresses the input features into a $1 \times 1 \times C$ channel descriptor.

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x_{i,j,c} \tag{4}$$

To capture the dependencies between channels, we add a simple non-linear function of two fully-connected layers connected with a ReLU [7] activation function. The output of this operation is the relevance score for the layer. Specifically, it is a vector β containing unnormalized scores for the two actions of (a) computing and (b) skipping the following layer, respectively.

$$\beta = \mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{z}) \tag{5}$$

where σ refers to the ReLU, $\mathbf{W}_1 \in \mathbb{R}^{d \times C}$, $\mathbf{W}_2 \in \mathbb{R}^{2 \times d}$ and d is the dimension of the hidden layer. The lightweight design of the gating function leads to minimal computational overhead. For a ConvNet-AIG based on ResNet 101 for ImageNet, the gating function adds only a computational overhead of 0.04%, but allows to skip 33% of its layers on average.

3.3 Greedy Gumbel Sampling

The goal of the second component is to make a discrete decision based on the relevance scores. For this, we build upon recent work that propose approaches for propagating gradients through stochastic neurons [4, 20]. In particular, we utilize the Gumbel-Max trick [9] and its recent continuous relaxation [18, 24].

A naïve attempt would be to choose the maximum of the two relevance scores to decide whether to execute or skip the layer. However, this approach leads to rapid mode collapse as it does not account for the gate's uncertainty and it is further not differentiable. Ideally, we would like to choose among the two options proportional to their relevance scores. A standard way to introduce such stochasticity is to add noise to the scores.

We choose the Gumbel distribution for the noise, because of its key property that is known as the Gumbel-Max trick [9]. A random variable G follows a Gumbel distribution if $G = \mu - \log(-\log(U))$, where μ is a real-valued location parameter and U a sample from the uniform distribution $U \sim \text{Unif}[0, 1]$. Then, the Gumbel-Max trick states that if we samples from K Gumbel distributions with location parameters $\{\mu_{k'}\}_{k'=1}^{K}$, the outcome of the k^{th} Gumbel is the largest exactly with the softmax probability of its location parameter

$$P(\text{k is largest} | \{\mu_{k'}\}_{k'=1}^{K}\}) = \frac{e^{\mu_k}}{\sum_{k'=1}^{K} e^{\mu_{k'}}} \tag{6}$$

With this we can parameterize discrete distributions in terms of Gumbel random variables. In particular, let X be a discrete random variable with probabilities $P(X = k) \propto \alpha_k$ and let $\{G_k\}_{k \in \{1,\dots,K\}}$ be a sequence of i.i.d. Gumbel random variables with location $\mu = 0$. Then, we can sample from the discrete variable X by sampling from the Gumbel random variables

$$X = \underset{k \in \{1,\dots,K\}}{\arg\max} \left(\log \alpha_k + G_k\right) \tag{7}$$

A drawback of this approach is that the argmax operation is not continuous. To address this, a continuous relaxation of the Gumbel-Max trick has been proposed [18, 24], replacing the argmax with a softmax. Note that a discrete random variable can be expressed as a one-hot vector, where the realization of the variable is the index of the non-zero entry. With this notation, a sample from the Gumbel-Softmax relaxation can be expressed by the vector $\hat{X}$ as follows:

$$\hat{X}_k = \text{softmax}\left(\left(\log \alpha_k + G_k\right)/\tau\right) \tag{8}$$

where $\hat{X}_k$ is the k^{th} element in $\hat{X}$ and τ is the temperature of the softmax. With $\tau \to 0$, the softmax function approaches the argmax function and Equation 8 becomes equivalent to the discrete sampler. For $\tau \to \infty$ it becomes a uniform distribution. Since softmax is differentiable and G_k is independent noise, we can propagate gradients to the probabilities α_k. To generate samples, we set the log probabilities to the estimated relevance scores, $\log \alpha = \beta$.

One option to employ the Gumbel-softmax estimator is to use the continuous version from Equation 8 during training and obtain discrete samples with Equation 7 during testing. An alternative is the *straight-through* version [18] of the Gumbel-softmax estimator. There, during training, for the forward pass we get discrete samples from Equation 7, but during the backwards pass we compute the gradient of the softmax relaxation in Equation 8. Note that the estimator is biased due to the mismatch between forward and backward pass. However, we observe that empirically the straight-through estimator performs better and leads to inference graphs that are more category-specific. We illustrate the two different paths during the forward and backward pass in Figure 2.

3.4 Training Loss

For the network to learn when to use which layer, we constrain how often each layer is allowed to be used. Specifically, we use soft constraints by introducing an additional loss term that encourages each layer to be executed at a certain target rate t. This guides the optimization to solutions in which parameters that are relevant only to subsets of related categories are grouped together in separate layers, which minimizes the amount of unnecessary features to be computed. We approximate the execution rates for each layer over each mini-batch and penalize deviations from the target rate. Let $\overline{z}_l$ denote the fraction of images within a mini-batch that layer l is executed. Then, the target rate loss is defined as

$$\mathcal{L}_{target} = \sum_{l=1}^{N} (\overline{z}_l - t)^2 \tag{9}$$

The target rate provides an easy instrument to adjust computation time. ConvNet-AIG is robust to a wide range of target rates. We study the effect of the target rate on classification accuracy and inference time in the experimental section. With the standard multi-class logistic loss, $\mathcal{L}_{MC}$, the overall training loss is

$$\mathcal{L}_{AIG} = \mathcal{L}_{MC} + \mathcal{L}_{target} \tag{10}$$

We optimize this joint loss with mini-batch stochastic gradient descent.

4 Experiments

We perform a series experiments to evaluate the performance of ConvNet-AIG and whether it learns specialized layers and category-specific inference graphs. Lastly, we study its robustness by analyzing the effect of adaptive inference graphs on the susceptibility towards adversarial attacks.

Table 1. Test error on CIFAR 10 in %. ConvNet-AIG 110 clearly outperforms ResNet 110 while only using a subset of 82% of the layers. When executing all layers (ConvNet-AIG 110*), it also outperforms stochastic depth.

Model	**Error**	**#Params** (10^6)	**FLOPs** (10^9)
ResNet 110 [11]	6.61	1.7	0.5
Pre-ResNet 110 [12]	6.37	1.7	0.5
Stochastic Depth ResNet 110 [16]	5.25	1.7	0.5
ConvNet-AIG 110	5.76	1.78	0.41
ConvNet-AIG 110*	**5.14**	1.78	0.5

4.1 Results on CIFAR

We first perform a set of experiments on CIFAR-10 [21] to validate the proposed gating mechanism and its effectiveness to distribute computation among layers.

Model configurations and training details We build ConvNet-AIG based on the original ResNet 110 [11]. Besides the added gates, ConvNet-AIG follows the same architecture as ResNet 110. For the gates, we choose a hidden state of size $d = 16$. The additional gate per residual block, adds a fixed overhead of 0.01% more floating point operations and 4.8% more parameters compared to the standard ResNet-110. We follow a similar training scheme as [11] with momentum 0.9 and weight decay 5×10^{-4}. All models are trained for 350 epochs with a mini-batch size of 256. We use a step-wise learning rate starting at 0.1 and decaying by 10^{-1} after 150 and 250 epochs. We adopt a standard data-augmentation scheme, where images are padded with 4 pixels on each side, randomly cropped to 32×32 and with probability 0.5 horizontally flipped.

Results Table 1 shows test error on CIFAR 10 for ResNet [11], pre-activation ResNet [12], stochastic depth [16] and their ConvNet-AIG counterpart. The table also shows the number of model parameters and floating point operations (multiply-adds). We compare two variants: For standard ConvNet-AIG, we only execute layers with open gates. As a second variant, which we indicate by " * ", we execute all layers and analogous to Dropout [27] and stochastic depth [16] the output of each layer is scaled by its expected execution rate.

From the results, we observe that ConvNet-AIG clearly outperforms its ResNet counterparts, even when using only a subset of the layers. In particular, ConvNet-AIG 110 with a target-rate of 0.7 uses only 82% of the layers in expectation. Since ResNet 110 might be over-parameterized for CIFAR-10, the regularization induced by dropping layers could be a key factor to performance. We observe that ConvNet-AIG 110* outperforms stochastic depth, implying benefits of adaptive inference graphs beyond regularization. In fact, ConvNet-AIG learns to identify layers of key importance such as downsampling layers and learns to always execute them, although they incur computation cost. We do not observe any downward outliers, i.e. layers that are dropped every time.

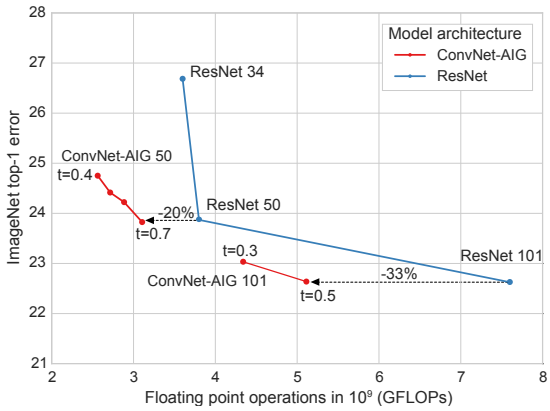

Fig. 3. Top-1 accuracy vs. computational cost on ImageNet. ConvNet-AIG 50 outperforms ResNet 50, while skipping 20% of its layers in expectation. Similarly, ConvNet-AIG 101 outperforms ResNet 101 while requiring 33% less computations. It is often more effective to decrease the target rate than to reduce the number of layers.

4.2 Results on ImageNet

In experiments on ImageNet [5], we study whether ConvNet-AIG learns to group parameters such that for each image only relevant features are computed. ImageNet is well suited for this study, as it contains a large set of categories with a wide variety including man-made objects, food, and many different animals.

Model configurations and training details We build ConvNet-AIGs based on ResNet 50 and ResNet 101 [11]. Again, we follow the same architectures as the original ResNets, with the sole exception of the added gates. The size of the hidden state is again $d = 16$, adding a fixed overhead of 3.9% more parameters and 0.04% more floating point operations. For ConvNet-AIG 50, all 16 residual layers have gates. For ConvNet-AIG 101, we fix the early layers up to the second downsampling operation to be always executed. The main reason is that early layers to not yet distinguish between object categories.

We follow the standard ResNet training procedure, with mini-batch size of 256, momentum of 0.9 and weight decay of 10^{-4}. All models are trained for 100 epochs with step-wise learning rate starting at 0.1 and decaying by 10^{-1} every 30 epochs. We use the data-augmentation procedure as in [11] and at test time first rescale images to 256×256 followed by a 224×224 center crop. The gates are initialized to open at a rate of 85% at the beginning of training.

Quantitative comparison Figure 3 shows top-1 error on ImageNet and computational cost in terms of GFLOPs for ConvNet-AIG with 50 and 101 layers and the respective ResNets of varying depth. We further show the impact of different target rates on performance and efficiency. We use target rates from 0.4

Table 2. Test error on ImageNet in % for ConvNet-AIG 50, ConvNet-AIG 101 and the respective ResNets of varying depth. Both ConvNet-AIGs outperform their ResNet counterpart, while at the same time using only a subset of the layers. This demonstrates that ConvNet-AIG is more efficient and also improves overall classification quality.

Model	**Top 1**	**Top 5**	**#Params** (10^6)	**FLOPs** (10^9)
ResNet 34 [11]	26.69	8.58	21.80	3.6
ResNet 50 [11]	24.7	7.8	25.56	3.8
ResNet 50 (our)	23.87	7.12	25.56	3.8
ResNet 101 [11]	23.6	7.1	44.54	7.6
ResNet 101 (our)	**22.63**	6.45	44.54	7.6
Stochastic Depth ResNet 50	27.75	9.14	25.56	3.8
Stochastic Depth ResNet 101	22.80	6.44	44.54	7.6
ConvNet-AIG 50 [t=0.4]	24.75	7.61	26.56	2.56
ConvNet-AIG 50 [t=0.5]	24.42	7.42	26.56	2.71
ConvNet-AIG 50 [t=0.6]	24.22	7.21	26.56	2.88
ConvNet-AIG 50 [t=0.7]	**23.82**	**7.08**	26.56	3.06
ConvNet-AIG 101 [t=0.3]	23.02	6.58	46.23	4.33
ConvNet-AIG 101 [t=0.5]	**22.63**	**6.26**	46.23	5.11

to 0.7 for ConvNet-AIG 50 and 0.3 to 0.5 for ConvNet-AIG 101. Details about the models' complexities and further baselines are presented in Table 2.

From the results we make the following key observations. Both ConvNet-AIG 50 and ConvNet-AIG 101 outperform their ResNet counterpart, while also using only a subset of the layers. In particular, ConvNet-AIG 50 with a target rate of 0.7 saves about 20% of computation. Similarly, ConvNet-AIG 101 outperforms its respective ResNet while using 33% less computations.

Figure 3 also visualizes the effect of the target rate. As expected, decreasing the target rate reduces computation time. Interestingly, penalizing computation first improves accuracy, before lowering the target rate further decreases accuracy. This demonstrates that ConvNet-AIG both improves efficiency and overall classification quality. Further, it appears often more effective to decrease the target rate compared to reducing layers in standard ResNets.

Due to surface resemblance, we also compare to stochastic depth [16]. We observe that for smaller ResNet models stochastic depth does not provide competitive results. Only very large models see benefits from stochastic depth regularization. The paper on stochastic depth [16] reports that even for the very large ResNet 152 performance remains below a basic ResNet. This highlights the opposite goals of ConvNet-AIG and stochastic depth. Stochastic depth aims to create redundant features by enforcing each subset of layers to model the whole dataset [31]. ConvNet-AIG aims to separate parameters that are relevant to different subsets of the dataset into different layers.

These results indicates that *convolutional networks do not need a fixed feed-forward structure* and that ConvNet-AIG is an effective means to enable adaptive inference graphs that are conditioned on the input image.

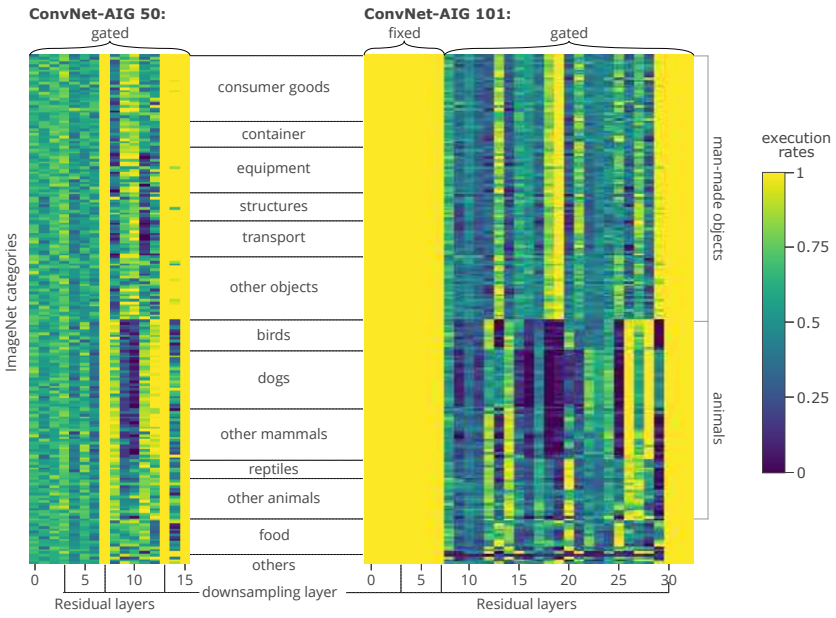

Fig. 4. Learned inference graphs on ImageNet. The histograms show for ConvNet-AIG 50 (left) and ConvNet-AIG 101 (right) how often each residual layer (x-axis) is executed for each of the 1000 classes in ImageNet (y-axis). We observe a clear difference between layers used for man-made objects and for animals and even for some mid-level categories such as birds, mammals and reptiles. Without specific supervision, the network discovers parts of the class hierarchy. Further, downsampling layers and the last layers appear of key importance and are executed for all images. Lastly, the left histogram shows that early layers are mostly agnostic to the different classes. Thus, we set early layers in ConvNet-AIG 101 to be always executed. The remaining layers are sufficient to provide different inference graphs for the various categories.

Analysis of learned inference graphs To analyze the learned inference graphs, we study the rates at which different layers are executed for images of different categories. Figure 4 shows the execution rates of each layer for ConvNet-AIG 50 on the left and ConvNet-AIG 101 on the right. The x-axis indicates the residual layers and the y-axis breaks down the execution rates by the 1000 classes in ImageNet. Further, the figure shows high-level and mid-level categories that contain large numbers of classes. The color in each cell indicates the percentage of validation images from a given category that the respective layer is executed.

From the figure, we see a clear difference between man-made objects and animals. Moreover, we even observe distinctions between mid-level animal categories such as birds, mammals and reptiles. This reveals that the network discovers part of the label hierarchy and groups parameters accordingly. Generally, we observe similar structures in ConvNet-AIG 50 and ConvNet-AIG 101. However,

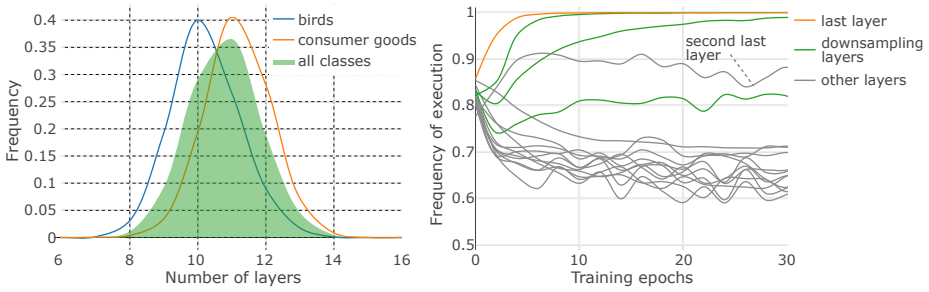

Fig. 5. Left: Distribution over the number of executed layers. For ConvNet-AIG 50 on ImageNet with target rate 0.4, in average 10.8 out of 16 residual layers are executed. Images of animals tend to use fewer layers than man-made objects. **Right: Execution rates per layer over first 30 epochs** of training. Layers are quickly separated into key and less critical layers. Downsampling layers and the last layer increase execution rate, while the remaining layers slowly approach the target rate.

the grouping of the mid-level categories is more distinct in ConvNet-AIG 101 due to the larger number of layers that can capture high-level features. This result demonstrates that ConvNet-AIG successfully learns layers that focus on specific subsets of categories. It is worthy to note that the training objective does not include an incentive to learn category specific layers. The specialization appears to emerge naturally when the computational budget gets constrained.

Further, we observe that downsampling layers and the last layers deviate significantly from the target rate and are executed for all images. This demonstrates their key role in the network (as similarly observed in [31]) and shows how ConvNet-AIG learns to effectively trade-off computational cost for accuracy.

Lastly, the figure shows that for ConvNet-AIG 50, inter-class variation is mostly present in the later layers of the network after the second downsampling layer. One reason for this could be that features from early layers are useful for all categories. Further, early layers might not yet capture sufficient semantic information to discriminate between categories. Thus, we keep the early layers of ConvNet-AIG 101 fixed to be always executed. The remaining layers still provide sufficient flexibility for different inference paths for the various categories.

Figure 5 shows on the right a typical trajectory of the execution rates during training for ConvNet-AIG 50. The layers are initialized to execute a rate of 85% at the start of training. The figure shows the first 30 training epochs and highlights how the layers are quickly separated into key layers and less critical layers. Important layers such as downsampling and the last layers increase their execution rate, while the remaining layers slowly approach the target rate.

Variable inference time Due to the adaptive inference graphs, computation time varies across images. Figure 5 shows on the left the distribution over how many of the 16 residual layers in ConvNet-AIG 50 are executed over all Ima-

Fig. 6. Validation images from ImageNet that use the fewest layers (top) and the most layers (bottom) within the categories of birds, dogs and musical instruments. The examples illustrate how instance difficulty translates into layer usage.

geNet validation images. On average 10.81 layers are executed with a standard deviation of 1.11. The figure also highlights the mid-level categories of birds and consumer goods. In expectation, images of birds use one layer less than images of consumer goods. From Figure 4 we further know the two groups also use different sets of layers. Figure 6 shows the validation images that use the fewest and the most layers within the categories of birds, dogs and musical instruments. The examples highlight that easy instances with iconic views require only a few layers. Difficult instances that are small or occluded need more computation.

4.3 Robustness to adversarial attacks

In a third set of experiments we aim to understand the effect of adaptive inference graphs on the susceptibility towards adversarial attacks. On one hand, if adversarial perturbations change the inference graph such that key layers of the network are skipped, performance might degrade. On the other hand, the stochasticity of the graph might improve robustness.

We perform a Fast Gradient Sign Attack [8] on ResNet 50 and ConvNet-AIG 50, both trained on ImageNet. The results are presented in Figure 7. In the graph on the left, the x-axis shows the strength of the adversary measured in the amount each pixel can to be changed. The y-axis shows top-1 accuracy on ImageNet. We observe that ConvNet-AIG is consistently more robust, independent of adversary strength. To investigate whether this additional robustness complements other defenses [10], we perform JPEG compression on the adversarial examples. We follow [10] and use a JPEG quality setting of 75%. While both networks greatly benefit from the defense, ConvNet-AIG remains more robust, indicating that the additional robustness can complement other defenses.

To understand the effect of the attack on the gates, we look at the execution rates before and after the attack. On the right side, Figure 7 shows the average execution rates per layer over all bird categories for ConvNet-AIG 50 before and after a FGSM attack with epsilon 0.047. Although the accuracy of the network drops from 74.62% to 11%, execution rates remain similar. One reason for the gates' resilience might be the stochasticity induced by the Gumbel noise which

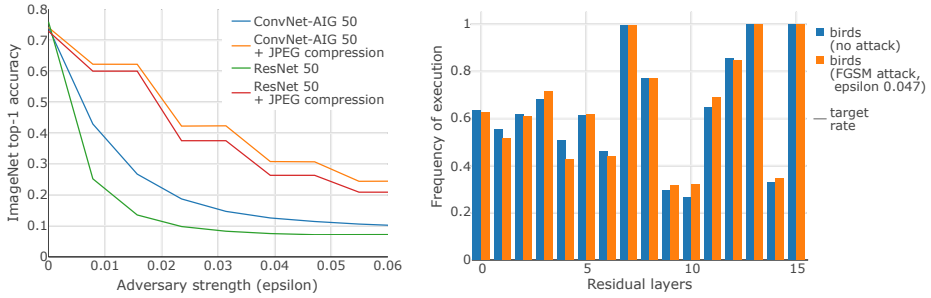

Fig. 7. Adversarial attack using Fast Gradient Sign Method. Left: ConvNet-AIG is consistently more robust than the plain ResNet, independent of adversary strength. The additional robustness persists even when applying additional defense mechanisms. **Right:** Average execution rates per layer for images of birds before and after the attack. The execution rates remain mostly unaffected by the attack.

might outweigh the noise introduced by the attack. Further, the global average pooling operation might cancel out some of the adversarial perturbations.

5 Conclusion

In this work, we have shown that convolutional networks do not need fixed feed-forward structures. With ConvNet-AIG, we introduced a ConvNet that adaptively assembles its inference graph on the fly based on the input image. Experiments on ImageNet show that ConvNet-AIG groups parameters for related classes into specialized layers and learns to only execute those layers relevant to the input. This allows decoupling inference time from the number of learned concepts and improves both efficiency as well as overall classification quality.

This work opens up numerous paths for future work. With respect to network architecture, it would be intriguing to extend this work beyond ResNets to other structures such as densely-connected [15] or inception-based [29] networks. From a practitioner's point of view, it might be exciting to extend this work into a framework where the set of executed layers is adaptive, but their number is fixed so as to achieve constant inference times. Further, we have seen that the gates are largely unaffected by basic adversarial attacks. For an adversary, it could be interesting to investigate attacks that specifically target the gating functions.

Acknowledgements

We would like to thank Ilya Kostrikov, Daniel D. Lee, Kimberly Wilber, Antonio Marcedone and Yiqing Hua for insightful discussions and feedback. This work was supported in part by the Oath Laboratory for Connected Experiences, a Google Focused Research Award, AWS Cloud Credits for Research and a Facebook equipment donation.

References

1. Andreas, J., Rohrbach, M., Darrell, T., Klein, D.: Learning to compose neural networks for question answering. Proceedings of NAACL-HLT (2016)
2. Andreas, J., Rohrbach, M., Darrell, T., Klein, D.: Neural module networks. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
3. Bengio, E., Bacon, P.L., Pineau, J., Precup, D.: Conditional computation in neural networks for faster models. arXiv preprint arXiv:1511.06297 (2015)
4. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
6. Figurnov, M., Collins, M.D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., Salakhutdinov, R.: Spatially adaptive computation time for residual networks. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
7. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: International Conference on Artificial Intelligence and Statistics (AISTATS) (2011)
8. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
9. Gumbel, E.J.: Statistical theory of extreme values and some practical applications: a series of lectures. No. 33, US Govt. Print. Office (1954)
10. Guo, C., Rana, M., Cisse, M., van der Maaten, L.: Countering adversarial images using input transformations. arXiv preprint arXiv:1711.00117 (2017)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
12. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision (ECCV) (2016)
13. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507 (2017)
14. Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K.Q.: Multi-scale dense convolutional networks for efficient prediction. arXiv preprint arXiv:1703.09844 (2017)
15. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
16. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: European Conference on Computer Vision (ECCV) (2016)
17. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: International Conference on Computer Vision (ICCV) (2017)
18. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144 (2016)
19. Johnson, J., Hariharan, B., van der Maaten, L., Hoffman, J., Fei-Fei, L., Zitnick, C.L., Girshick, R.: Inferring and executing programs for visual reasoning. International Conference on Computer Vision (ICCV) (2017)
20. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
21. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)

22. Li, H., Lin, Z., Shen, X., Brandt, J., Hua, G.: A convolutional neural network cascade for face detection. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
23. Li, Y., Wang, N., Liu, J., Hou, X.: Demystifying neural style transfer. arXiv preprint arXiv:1701.01036 (2017)
24. Maddison, C.J., Mnih, A., Teh, Y.W.: The concrete distribution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712 (2016)
25. Misra, I., Gupta, A., Hebert, M.: From red wine to red tomato: Composition with context. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
26. Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., Dean, J.: Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv:1701.06538 (2017)
27. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of machine learning research (JMLR) **15**(1), 1929–1958 (2014)
28. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv preprint arXiv:1505.00387 (2015)
29. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Conference on Computer Vision and Pattern Recognition (CVPR)
30. Teerapittayanon, S., McDanel, B., Kung, H.: Branchynet: Fast inference via early exiting from deep neural networks. In: Conference onPattern Recognition (ICPR) (2016)
31. Veit, A., Wilber, M.J., Belongie, S.: Residual networks behave like ensembles of relatively shallow networks. In: Advances in Neural Information Processing Systems (NIPS) (2016)
32. Viola, P., Jones, M.J.: Robust real-time face detection. International journal of computer vision (IJCV) **57**(2), 137–154 (2004)
33. Yang, F., Choi, W., Lin, Y.: Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)