Shallow-Deep Networks: Understanding and Mitigating Network Overthinking

Yigitcan Kaya [1] **Sanghyun Hong** [1] **Tudor Dumitraş** [1]

Abstract

We characterize a prevalent weakness of deep neural networks (DNNs)—*overthinking*—which occurs when a DNN can reach correct predictions before its final layer. Overthinking is computationally *wasteful*, and it can also be *destructive* when, by the final layer, a correct prediction changes into a misclassification. Understanding overthinking requires studying how each prediction *evolves* during a DNN's forward pass, which conventionally is opaque. For prediction transparency, we propose the Shallow-Deep Network (SDN), a generic modification to *off-the-shelf DNNs* that introduces *internal classifiers*. We apply SDN to four modern architectures, trained on three image classification tasks, to characterize the overthinking problem. We show that SDNs can mitigate the wasteful effect of overthinking with confidence-based *early exits*, which reduce the average inference cost by more than 50% and preserve the accuracy. We also find that the destructive effect occurs for 50% of misclassifications on natural inputs and that it can be induced, adversarially, with a recent backdooring attack. To mitigate this effect, we propose a new *confusion* metric to quantify the internal disagreements that will likely lead to misclassifications.

1 Introduction

Deep neural networks (DNNs) have enabled breakthroughs in many tasks, such as image classification (Krizhevsky et al., 2012) and speech recognition (Hinton et al., 2012). In these tasks, a DNN's sequence of layers resembles human perception in the way it combines simple representations, such as edges, into more complex ones, such as faces (Zeiler & Fergus, 2014). A fundamental difference is that people can learn simpler heuristics that allow them to perform even complex tasks, such as driving or playing the piano, with little mental effort (Kahneman, 2011). When simpler heuristics are adequate to complete the task, using excessively complex representations leads to *overthinking*. *Human overthinking* is considered wasteful because it leads to slow decision making, as people think too much or for too long. Furthermore, overthinking is potentially destructive, by causing confusion and mistakes when human attention is drawn to irrelevant details.

In contrast, the decision-making process of conventional DNNs—the forward pass—requires the same computational effort on all inputs, whether they are simple or difficult to classify. Building on this analogy, we ask: *Are deep neural networks also susceptible to overthinking?*. We consider that a network overthinks[1] on an input sample when its simpler representations at an earlier layer—relative to the final layer— are adequate to make a correct classification. Analogously to human overthinking, we hypothesize that further computation after this layer leads to waste and, potentially, a misclassification.

Our definition of overthinking relates to how a prediction *evolves* throughout the forward pass. Intuitively, a DNN produces predictions through a gradual process, as the subsequent layers recognize different features of the input. In conventional DNNs, however, this process remains mostly opaque as they are only able to provide a *final prediction*. The prior work that proposed early exits (Lee et al., 2015; Szegedy et al., 2015) or error indicators (Szegedy et al., 2013; Papernot & McDaniel, 2018) for off-the-shelf DNNs focused on producing and interpreting a single prediction for each input, rather than on illuminating how this prediction evolves from layer to layer.

Our first contribution is the *Shallow-Deep Network* (SDN): a generic modification to off-the-shelf DNNs for introducing *internal classifiers* (ICs). Our modification attaches ICs to various stages of the forward pass, as Figure 1 illustrates, and effectively combines shallower and deeper networks into one. The feature reduction in an IC acts as a regularizer and ensures that our method can scale up to large

[1]University of Maryland, Maryland, USA. Correspondence to: Yigitcan Kaya <cankaya@umiacs.umd.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

[1]Prior work has utilized this term to describe a property of the network as a whole, disregarding its internal state (Wang et al., 2017). In contrast, our definition is closely tied to the network's internal state and is analogous to human overthinking.

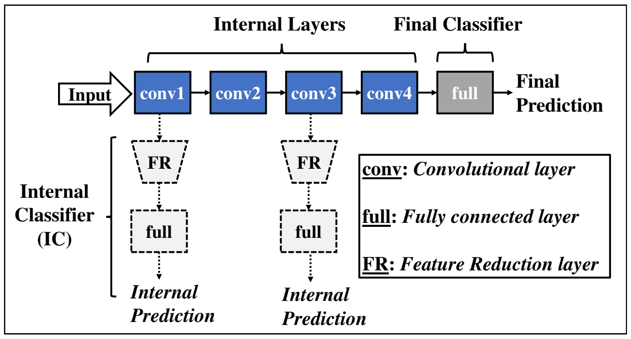

Figure 1: Modifying a standard convolutional neural network as a Shallow-Deep Network. Our modification introduces the dotted components—two ICs. The resulting network produces three predictions: two internal and one final.

DNNs. We can apply the modification to both pre-trained and untrained DNNs. The conversion from a pre-trained DNN is efficient as we only train the parameters in the ICs. Moreover, by using a weighted loss function, we can also train the original network from scratch jointly with the ICs. This mode of training, the *SDN training*, often improves the original accuracy and yields more accurate ICs. Our modification is practical for a range of existing architectures and allows us to explore the overthinking problem.

Our second contribution is to show that convolutional neural networks (CNNs) overthink on the majority of inputs. Our study exposes that CNN overthinking can be both *wasteful* and *destructive*. When a CNN reaches a correct internal prediction before its final layer, the remaining layers are effectively rendered redundant. Our experiments show that, for up to $\sim$95% of instances, overthinking leads to slow inferences and wasted computation. This suggests that the *complex* inputs requiring the full network depth are uncommon. More surprisingly, we observe the destructive effect of overthinking in up to $\sim$50% of a CNN's errors on natural inputs, where a correct internal prediction evolves into a misclassification. Moreover, we show that a recent backdooring attack on CNNs (Gu et al., 2017) induces the same effect on the victim network.

We further utilize SDNs as a vehicle for mitigating the overthinking problem. As in human problem-solving, we cannot evaluate perfectly whether a classification is correct, but we can utilize heuristics against overthinking. Our third contribution is to propose two different heuristics: confidence-based *early exits* and analysis of *confusion*.

Our first heuristic uses the *confidence* of an internal prediction to assess its correctness. With this heuristic, we can reliably detect when the network should stop *thinking* and make an early prediction—an *early exit*. Without any loss of accuracy, we can reduce the average inference cost by up to $\sim$75% and mitigate the wasteful effect of overthinking. The destructive effect of overthinking, on the other hand, poses a greater challenge for an early exit mechanism. The disagreements among the internal predictions hint the state of *confusion* the network is in. The confusion hurts the confidence of correct internal predictions and, as a result, leads inputs to bypass the early exits. In our second heuristic, we devise a new *confusion metric* that quantifies the internal disagreements. We observe experimentally that confusion scores reliably indicate whether the network is likely to misclassify an input. Furthermore, by visualizing the confusion—the disagreements—we can investigate the input elements that cause the confusion. In addition to their practical application regarding diagnosing DNN errors, these visualizations provide a new perspective for reasoning about model interpretability.

We evaluate our techniques on three tasks: CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009), and Tiny ImageNet[2]; by applying the SDN modification to four off-the-shelf CNN architectures: VGG (Simonyan & Zisserman, 2014), ResNet (He et al., 2016), Wide ResNet (Zagoruyko & Komodakis, 2016) and MobileNets (Howard et al., 2017). An SDN's early exits mitigate the wasteful effect of overthinking and cut the average inference costs by more than 50% in CIFAR-10 and CIFAR-100, and by more than 25% in Tiny ImageNet. Further, early exits can improve a CNN's accuracy by up to 8% and recover the accuracy of a backdoored CNN from 12% to 84% on malicious inputs. Moreover, our normalized confusion scores suggest that a network is significantly less confused in a correct prediction (-0.3 on average) than in a wrong prediction (0.7 on average). We also release all of our source code[3].

2 Preliminaries

2.1 Related Work

Internal classifiers & Early exits. Deeply Supervised Networks (Lee et al., 2015) and Inception architecture (Szegedy et al., 2015) have proposed using internal classifiers for improving overall accuracy. In (Huang et al., 2018) and (Belilovsky et al., 2018), authors propose methods to train a DNN sequentially, block-by-block, for reducing the training costs. These studies discard the internal classifiers, as they are not designed, or trained, for accurate predictions. The MSDNets (Huang et al., 2017) architecture aims to provide any time predictions, in the case of insufficient computing resources. Here, the authors claim that attaching internal classifiers to existing architectures hurts the final performance. BranchyNets (Teerapittayanon et al., 2016) architecture augments DNNs for improving the inference times, but it requires training the network from

[2]http://tiny-imagenet.herokuapp.com/
[3]www.shallowdeep.network

scratch. Adaptive Neural Networks (Bolukbasi et al., 2017) and Runtime Neural Pruning (Lin et al., 2017) intend to reduce the evaluation time by selectively evaluating the components of a network. All these methods implicitly attempt to mitigate only the wasteful effect of overthinking; however, our primary goal is to fully understand then remedy the problem. Further, we design Shallow-Deep Networks as a modification, instead of a new architecture, for introducing internal classifiers to pre-trained off-the-shelf networks. Our modification achieves two goals previous studies have suggested to be contradictory: having accurate internal classifiers while preserving the original performance.

Neural Network Error Indicators. Widespread usage of DNNs makes indications of prediction correctness crucial for handling the potential errors. Prior work has shown that predicted probability distribution—*softmax scores*—makes an unreliable metric, as it is over-confident towards a single class (Szegedy et al., 2013). In this regard, prior work has described a calibration scheme (Guo et al., 2017) and Bayesian model uncertainty estimation (Gal & Ghahramani, 2016). The credibility metric (Papernot & McDaniel, 2018) uses the representational distances to neighboring training points. SDNs inherently allow the design of our *confusion* metric that does not require any calibration or expensive computations, and can be applied to pre-trained off-the-shelf DNNs. Confusion captures how consistently a network reaches a final prediction. Further, our method enables diagnosis of the errors by visualizing the input elements responsible for the confusion.

2.2 Experimental Setup

Datasets. In our experiments, we use three datasets for benchmarking: CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009) and Tiny ImageNet. The two CIFAR datasets consist of 32x32 pixels, colored natural images. CIFAR-10 and CIFAR-100 images are drawn from 10 and 100 classes, respectively; containing 50,000 training and 10,000 validation images. We use a standard data augmentation scheme: padding, random cropping, and random horizontal mirroring. The Tiny ImageNet dataset consists of a subset of ImageNet images (Deng et al., 2009), resized at 64x64 pixels. There are 200 classes, each of which has 500 training and 50 validation images. We augment the data with random crops, horizontal mirroring, and RGB intensity scaling.

Architectures and Hyper-Parameters. We experiment with four off-the-shelf CNNs: VGG (Simonyan & Zisserman, 2014), ResNet (He et al., 2016), Wide-ResNet (WRN) (Zagoruyko & Komodakis, 2016) and MobileNet (Howard et al., 2017). Specifically, we use VGG-16, ResNet-56 and WRN-32-4 configurations of these architectures. We train the CNNs for 100 epochs, using the hyper-parameters the original studies describe. In our experiments, we use these CNNs as our pre-trained, off-the-shelf networks. We denote the network after our SDN modification by appending *'SDN'* to its original name; e.g. *VGG-16* becomes *VGG-16-SDN*. To apply SDNs to pre-trained networks, we train the internal classifiers for 25 epochs, using the Adam optimizer (Kingma & Ba, 2014). If we start training a modified network from scratch, we train for 100 epochs; the same as the original networks.

Metrics. To quantify the test-time inference cost of a network, we measure the average number of floating point operations (*FLOPs*) a network performs to classify an input. As for the classification performance, we simply report the Top-1 accuracy on the test data (as a percentage).

3 The Shallow-Deep Network

Setting. We consider the supervised learning setting and the standard DNN structure: a sequence of *internal layers* ending with a *final classifier* (see Figure 1). Let $S = (x, y) \in (X, Y)$ be our input-output pairs, where x denotes a sample, and $y \in \{1, ...K\}$ is its correct ground-truth label. Given x, a DNN with M internal layers performs a *classification*, $\mathcal{F}_{final}(F_M(, ...F_1(x)))$, to predict the probability of x belonging to each class label. Here, F_m denotes the *learnable* function layer m applies, and $\mathcal{F}_{final}$ denotes the final classifier—usually a fully connected layer. To simplify the notation, we write the output of the layer m as $F_m(x)$ and the vector of predicted probabilities as $\mathcal{F}_{final}(x)$. We denote the *final prediction* on the sample x as $\hat{y}_{final} = \mathrm{argmax}_j\, \mathcal{F}^{(j)}_{final}(x)$, i.e. the predicted label with the highest probability.

Overview. In this work, we specifically focus on off-the-shelf CNNs and leave the extension to other architectures for future work. Figure 1 outlines how we apply our modification. We call a modified network a Shallow-Deep Network (SDN). An SDN contains *internal classifiers* (ICs) and essentially combines *shallow* networks (earlier ICs) and *deep* networks (later ICs and the final classifier) into a single structure. In addition to the final prediction $\hat{y}_{final}$, an SDN produces multiple *internal predictions* at its ICs. Formally, i^{th} IC (IC_i) following the layer m is a learnable function: $\mathcal{F}_{i,m}$, $1 \leq m \leq M$ and $1 \leq i \leq N$ (N is the total number of ICs). $\mathcal{F}_{i,m}$ takes $F_m(x)$ as its input and performs a classification on the sample x: $\mathcal{F}_{i,m}(F_m(x))$, or simply as $\mathcal{F}_i(x)$. We denote the i^{th} internal prediction as $\hat{y}_i = \mathrm{argmax}_j\, \mathcal{F}^{(j)}_i(x)$. The modification includes two steps: attaching several ICs (Section 3.1), and training these ICs (Section 3.2). We elaborate on the cost of SDNs in Section 3.3. We demonstrate the outcome of such modification and the accuracy of the ICs in Section 3.4.

3.1 Attaching the Internal Classifiers (ICs)

An internal classifier consists of two parts: a single fully connected layer that follows a feature reduction layer. Large output sizes of a CNN's internal layers necessitate the feature reduction for scalability. The feature reduction layer takes $F_m(x)$ and reduces its size. The fully connected layer, using the reduced $F_m(x)$, produces the internal prediction. Instead of using a more complex structure, such as convolutional layers or a multilayer perceptron (Szegedy et al., 2015; Teerapittayanon et al., 2016), we opt for using a fully connected layer to keep our modification minimal and generic. Further, this allows us to closely monitor how predictions evolve throughout the forward pass in Section 4 and highlight the prevalance of the overthinking problem without any interference from our IC mechanism.

Feature Reduction Layers. We observe that neither average nor max pooling consistently performs better for reducing the feature map size—the output dimension of one convolution filter. Here, we employ a *mixed* max-average pooling strategy (Lee et al., 2016), which learns the mixing proportion of two pooling methods from the data, without any manual tuning. We simply pick the pooling size such that any feature map size larger than 4x4 is pooled into 4x4—the smaller sizes remain the same. Without any reduction—with only the fully connected layer—ICs increase the size of the original network by up to $\sim$18x. This is brought down to at most $\sim$3x by feature reduction.

Placement of the ICs. We pick a subset of internal layers to attach the internal classifiers after. Our selection criterion is simple: we pick the internal layers closest to the 15%, 30%, 45%, 60%, 75% and 90% of the full network's inference cost—the number of *FLOPs*. Given an input, our SDNs produce a total of 6 internal predictions and a final prediction. For each IC, we denote the *relative inference cost to the whole network* as $\mathcal{C}_i : 1 \leq i \leq N$—for our SDNs, $N = 6$ and $\mathcal{C}_1 \approx 0.15 ... \mathcal{C}_6 \approx 0.9$. For the final classifier: $\mathcal{C}_{final} = 1$. Overall, this simplifies our analysis of overthinking and eliminates the need for tuning.

3.2 Training the Internal Classifiers

To complete our modification, we train the ICs using the training set, via backpropagation. The training strategy is based on whether the original CNN is pre-trained—the *IC-only training*—, or untrained—the *SDN training*.

The IC-only Training. To *convert* a pre-trained network to an SDN, we freeze original weights and train only the weights in the attached ICs. Freezing prevents any change to the off-the-shelf network, and, as a result, keeps its predictions fully intact. The IC-only training is fast as the backpropagation updates only the weights in the ICs.

Table 1: Computational comparison between off-the-shelf CNNs and SDNs. The left side of a cell is for the CNNs. **#prms** presents the total number of parameters. **#ops** reports the total number of FLOPs performed for each sample. **epc** lists the training time per epoch—For SDNs both the IC-only and the SDN training strategies are listed.

NETWORK	#PRMS (M)	#OPS (B)	EPC (MIN)
$\mathcal{V}$GG-16	21.2 | 26.8	2.5 | 2.5	3.4 | 1.4 - 3.8
$\mathcal{R}$ESNET-56	0.9 | 1.6	0.2 | 0.2	1.0 | 0.5 - 1.1
$\mathcal{W}$RN-32-4	7.4 | 10.3	2.4 | 2.4	3.5 | 1.2 - 3.9
$\mathcal{M}$OBILENET	3.4 | 11.2	3.0 | 3.0	1.9 | 0.8 - 2.1

The SDN Training. The IC-only training has a major drawback: the standard network training aims to improve only the final prediction accuracy. Attaching classifiers on top of these pre-trained internal layers, as a result, yields relatively weak ICs. Here, we propose a *weighted loss function* to train the original weights, from scratch, jointly with the ICs, similar to previous studies (Lee et al., 2015). However, unlike previous methods, we design an objective function for achieving high accuracy in all ICs in addition to the final classifier. Our objective multiplies each $\mathcal{L}_i$, the training loss values from i^{th} IC, with coefficients τ_i. It then adds their sum to the loss from the final classifier. Our selection of τ_i reflects two observations: (i) the initial phases of the training are less stable due to higher learning rate; and (ii) the earlier ICs have less learning capacity. In this regard, we start the training with $\tau_i = 0.01$ and linearly increase it until $\tau_i = \mathcal{C}_i$—the i^{th} IC's relative inference cost.

3.3 Cost of the Modification

Table 1 highlights the cost of attaching ICs on Tiny ImageNet task. Here, we do not consider any early exits at all; and the input samples traverse until the end of the SDN. The overhead on the required test-time computation—FLOPs per input—is minor; as the fully connected layers are computationally efficient. As expected, the IC-only training is $\sim$3x faster than original training—it also requires $\sim$4x fewer training epochs. In the SDN training strategy, on the other hand, we see a 10% increase in training time. Finally, because of the fully connected layers, the modification increases the number of parameters.

3.4 The Accuracy of the Internal Classifiers

Table 2 presents the relative accuracy of the ICs on Tiny ImageNet task. The SDN training consistently enables ICs that have higher accuracy than the original network. This improvement is more pronounced on more difficult tasks, increasing the original accuracy by up to 4.5%. The SDN training optimizes the internal layers for highly discriminative features, similar to (Lee et al., 2015). As a result, the

Table 2: The accuracy of the internal classifiers (ICs) on Tiny ImageNet task. **Network** lists the original CNNs and their test accuracies. ∼**80% IC** and ∼**90% IC** list the IC's index (from 1 to 6) which has accuracy closest to 80% and 90% of the CNN's. **Max% IC** presents the highest accuracy of an IC and its index. We highlight when an IC's accuracy exceeds the CNN's. The left and right sides of each cell are for the IC-only and the SDN training strategies.

Network	∼80% IC	∼90% IC	Max% IC
$\mathcal{V}$*(58.6)*	3 | 2	4 | 3	**59.3**(6) | **63.1**(6)
$\mathcal{R}$*(53.9)*	4 | 3	4 | 3	50.1(6) | **54.1**(6)
$\mathcal{W}$*(60.3)*	5 | 3	6 | 4	54.6(6) | **62.2**(6)
$\mathcal{M}$*(59.3)*	3 | 2	4 | 3	58.3(6) | **59.6**(6)

classifiers built on these features outperform their counterparts. Even with the IC-only training, ICs have relatively high accuracy that exceeds the original accuracy in one case. Further, the networks reach most of their original accuracy at an IC; which has significantly less inference cost. Altogether, the accuracy of the ICs demonstrates the extent of the computational waste overthinking causes.

4 Understanding the Overthinking Problem

Shallow-Deep Networks, by providing explicit internal predictions, allow us to study the overthinking problem in CNNs. In the context of an SDN, we define overthinking on an input sample as the network's ability to reach a correct internal prediction. Formally, the network overthinks on the input sample (x, y) if $\hat{y}_i = y$, i.e. the i^{th} internal prediction is a correct one.

Overthinking is problematic as the rest of the forward pass ends up being invoked for no clear benefit. The computational waste this leads to is *the wasteful effect* of overthinking (Section 4.1). Moreover, in some cases, excessive processing ultimately leads to a misclassification, i.e. $\hat{y}_i = y \neq \hat{y}_{final}$; or causes a vulnerability (Wang et al., 2018). We categorize this as *the destructive effect* of overthinking (Section 4.2). Note that, as opposed to *overfitting*, overthinking leads to both wasted computation and misclassifications. Further, whereas regularization, such as Dropout (Srivastava et al., 2014), can mitigate overfitting; there is no general remedy for overthinking.

To illustrate the overthinking problem, we focus on a case study: *VGG-16* network trained on Tiny Imagenet task (59% test accuracy). We convert this network to an SDN—to *VGG-16-SDN*—via the IC-only training strategy. The conversion allows us to monitor how the original network's predictions evolve, without altering them at all.

4.1 The Wasteful Effect of Overthinking

Quantifying the Effect. For *VGG-16-SDN*, assuming that a sample exits at an IC right after a correct internal prediction; 24%, 15%, 14%, 6%, 8% and 4% of the test samples would exit at each IC. The remaining 29% of the samples exit at the final classifier—4% correctly and 25% wrongly classified. Based on these *ideal* exit rates and $\mathcal{C}_i$'s (the IC's relative inference costs); the average inference cost would decrease by 43% as a result of eliminating overthinking. Further, on CIFAR-10, 95% of the samples do not require the full depth of a CNN —81% on CIFAR-100 and 69% on Tiny ImageNet—; whereas the rest exit at the final classifier. In consequence, eliminating overthinking would cut the average inference costs by 73% on CIFAR-10, by 55% on CIFAR-100, and by 40% on Tiny ImageNet. Evidently, overthinking leads to more severe waste of computation as the task gets less complex. Note that our assumption here is impractical as it requires a perfect way to discern a wrong classification from a correct one—Section 5.1 presents a realistic way.

Explaining the Behavior. Considering the perfect exit rates, 29% of the samples still need to be forwarded until the final classifier; whereas 24% can exit at the first IC. Figure 2 shows randomly sampled images—the input samples—that exit at each IC in *VGG-16-SDN*. The first column presents the samples that are correctly classified at the first IC (IC_1), and so on. In subsequent columns of the figure, notice how the input *complexity* progressively increases. These images suggest that the earlier ICs learn simple representations that allow them to recognize typical samples from a class, but that fail on atypical samples, e.g. where the subject is occluded or zoomed out. Overthinking leads to wasted computation as the network applies unnecessarily complex representations to classify the input. On the other hand, not having enough representational complexity causes wrong predictions, e.g. the IC_1 misclassifies the samples in the remaining columns of the figure.

4.2 The Destructive Effect of Overthinking

Quantifying the Effect. Out of 41% of the samples *VGG-16* misclassifies; 3% are actually correctly classified first at IC_1 of *VGG-16-SDN*—4% at IC_2, 3% at IC_3, 2% at IC_4, 3% at IC_5, and 1% at IC_6. As a result, the *cumulative accuracy* of *VGG-16-SDN* is 75%—16% higher than the original accuracy. In cumulative accuracy, we consider a sample to be correctly classified if there is *at least one* correct internal prediction—or the final prediction—of the SDN. This metric measures the ideal accuracy a network can reach after eliminating overthinking. The difference between the cumulative and the original accuracies quantifies the destructive effect: 4% on CIFAR-10, 13% on CIFAR-100, and 14% on Tiny ImageNet task. This effect also ex-

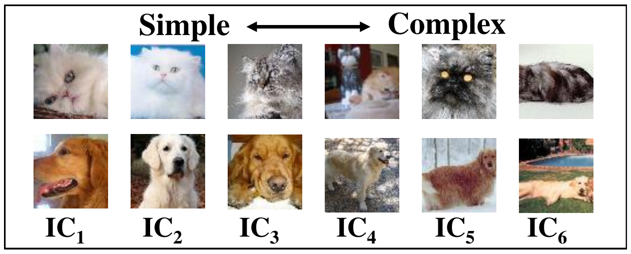

Figure 2: Sample images from the Tiny ImageNet classes *Persian Cat* (top row) and *Golden Retriever* (bottom row). Each column presents the samples that are correctly classified first at the given IC. We can see how the samples get progressively complex over ICs. The network is *VGG-16-SDN*, trained using the IC-only training strategy.

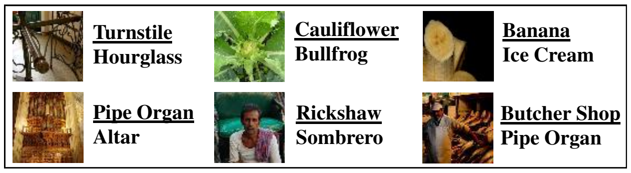

Figure 3: A set of Tiny ImageNet samples only the first internal classifier of *VGG-16-SDN* can correctly classify. For each image, two labels are the correct label and the wrong prediction at the final classifier, respectively.

plains the reason why an IC could outperform the final classifier in Table 2: at a certain IC, the destructive effect starts to outweigh the accuracy gain from increased feature complexity; after which the accuracy starts decreasing. Overall, the destructive effect can be seen in up to 50% of all misclassifications a CNN makes.

Explaining the Behavior. Figure 3 presents a selection of images that *only* the first IC can correctly classify—we identified 46 of such cases. *VGG-16* and all other ICs in *VGG-16-SDN* misclassify these samples. We hypothesize that these images consist of *confusing* elements, sometimes belonging to more than one class; while the first IC recognizes the objects displayed prominently, subsequent layers discover finer details that are irrelevant. These results suggest that correct classifications require the *appropriate* level of representational complexity for each sample, which further illustrates the utility of the ICs in an SDN. In Section 5.2, building on the idea of confusion, we design a new confusion metric and visually investigate the sources of confusion in these samples.

The Destructive Effect Leads to Malicious Outcomes. Recent work has shown the risk of backdoor attacks when the training was performed by a malicious trainer (Gu et al., 2017). A *backdoored* DNN, returned by the adversary, behaves normally on most inputs but causes targeted misclassifications when a trigger known only to the attacker is present. Our results suggest that backdooring attacks leverage the destructive effect of overthinking. In our experiments, we use a backdoored *VGG-16* network on CIFAR-10 task. The trigger is a small white square at the bottom right corner, and when it is present the network classifies any input being in the *dog* class. The accuracy of the backdoored network is 92% on clean samples and 12% on the samples containing the backdoor—on the backdoor inputs. The network classifies 98% of the backdoor inputs in the dog class; which demonstrates the impact of such a threat. Converting the backdoored network to an SDN–via the IC-only training strategy—illuminates the nature of the attack. The attack's success before the fourth IC (IC_4) is minimal: IC_4 makes correct predictions on 87% of the backdoor inputs. However, at IC_5 and IC_6, the correct predictions on the backdoor inputs suddenly drop to 76% and 38%—and 12% at the final classifier. This pattern fits in the destructive effect and indicates that a technique to mitigate overthinking might mitigate the attack as well—we present these results in Section 5.1.

5 Mitigating the Overthinking Problem

After understanding the overthinking problem, in this section, we propose new remedies to mitigate its adverse effects, i.e. being wasteful and destructive. For this purpose, we further utilize Shallow-Deep Networks as a vehicle for designing our techniques.

To prevent overthinking entirely, ideally, one could rely on early exits upon encountering a correct internal prediction. However, this requires a perfect way to determine the correctness of a given prediction. In human decision-making, people rely on imperfect heuristics to both conserve energy and avoid potential mistakes (Fiske & Taylor, 2013). Analogously, to mitigate network overthinking, we propose two SDN-based heuristics: the confidence-based *early exits* (Section 5.1) and network *confusion* analysis (Section 5.2).

5.1 Confidence-Based Early Exits

In Section 4.1, we show that an ideal, but *impractical*, early exit mechanism could eliminate overthinking entirely. Here, as a *practical* mechanism, we propose using the internal prediction *confidence* for determining when the network should stop thinking. Confidence allows us to simply decide between making an early exit or forwarding the input sample to subsequent layers. If none of the internal predictions—or the final prediction—are confident enough to for an exit; our mechanism outputs the most confident among them. We opt for a simple confidence mechanism for highlighting that we mitigate overthinking regardless of the mechanism, which could be improved with schemes, such as (Teerapittayanon et al., 2016). To quantify confidence, we use the estimated probability of the sample x be-

Table 3: Comparing the inference costs of the CNNs and SDNs with early exits. **N.** lists the original CNNs and their accuracies. **$\leq$25%**, **$\leq$50%**, and **$\leq$75%** report the early exit accuracy when we limit the average inference cost to at most 25%, 50% and 75% that of the original CNN's. **Max** reports the highest accuracy early exits can achieve. We highlight the cases where an SDN outperforms the original CNN. In each cell, the left and right accuracies are from the IC-only and the SDN training strategies.

N.	$\leq$25%		$\leq$50%		$\leq$75%		MAX	
CIFAR-10								
$\mathcal{V}$*(93.1)*	84.0	85.4	92.8	**93.2**	**93.2**	**93.4**	**93.2**	**93.4**
$\mathcal{R}$*(91.9)*	57.6	76.2	84.5	91.4	91.4	**92.1**	**91.9**	**92.1**
$\mathcal{W}$*(94.1)*	67.0	85.2	90.3	93.6	93.5	**94.1**	93.9	**94.1**
$\mathcal{M}$*(90.8)*	87.9	88.2	**91.1**	**91.5**	**91.3**	**91.5**	**91.3**	**91.5**
CIFAR-100								
$\mathcal{V}$*(70.9)*	53.3	55.5	68.9	**72.5**	**72.5**	**74.3**	**72.6**	**74.4**
$\mathcal{R}$*(68.8)*	46.1	46.7	61.2	65.7	67.2	**70.8**	**69.7**	**70.9**
$\mathcal{W}$*(75.1)*	50.4	65.8	66.5	74.7	74.3	**76.7**	**75.5**	**77.3**
$\mathcal{M}$*(64.9)*	55.9	57.5	**65.9**	**68.0**	**67.5**	**68.9**	**67.6**	**68.9**
TINY IMAGENET								
$\mathcal{V}$*(58.6)*	34.4	36.8	44.2	56.2	**58.5**	**63.1**	**60.4**	**63.4**
$\mathcal{R}$*(53.9)*	23.9	39.0	37.6	39.1	49.2	52.8	**54.0**	**55.1**
$\mathcal{W}$*(60.3)*	26.7	36.6	42.5	54.9	56.1	**62.6**	**61.0**	**62.8**
$\mathcal{M}$*(59.3)*	36.0	45.3	55.5	57.3	59.1	**61.5**	**60.3**	**61.8**

longing to the predicted class, i.e. $\max_j \mathcal{F}_i^{(j)}(x)$. We deem a prediction confident if this probability exceeds the threshold parameter q. The threshold facilitates on-the-fly adjustment of the early exits based on the resource availability and the performance requirements. A higher q value would make the early exits *conservative* and, in turn, reduce the early exit rates. In our experiments, we search for a q value ($0 \leq q \leq 1$) that satisfies our computational constraints on a small unlabeled holdout set.

Early Exits Mitigate the Wasteful Effect. Table 3 presents the early exit accuracy of our SDNs with a limited computational budget, in two training strategies. We calculate the computational cost when the input samples are evaluated individually; similar to previous studies (Teerapittayanon et al., 2016; Huang et al., 2017). We measure the accuracy under three budget settings: not exceeding 25%, 50% and 75% of the original CNN's inference cost. We control an SDN's average inference cost by adjusting the threshold parameter q. We also report the highest accuracy early exits can achieve, without any specific budget. First, the SDN training improves the early exit accuracy significantly; exceeding the original accuracy while reducing the inference costs more than 50% on CIFAR-10 and CIFAR-100 tasks. Even with the IC-only training strategy, early exits are still effective; allowing more than 25% reduced inference cost. In our most complex task, Tiny ImageNet, early exits can reduce the cost by more than 25%, usually without any accuracy loss. Overall, an SDN's early exits can mitigate the wasteful effect of overthinking.

Early Exits Mitigate the Backdoor Attack. In Section 4.2, we identified that a backdooring attack on *VGG-16* induces the destructive effect. Our early exit mechanism significantly reduces the success of this attack. When the threshold is at $q = 0.8$; the backdoored network makes correct predictions on 84% of the backdoor inputs—up from 12% without early exits. Further, the network classifies only 17% of the backdoor samples to the attacker's target class—down from 98%. Overall, our results suggest that early exits can mitigate this attack. We believe that Shallow-Deep Networks shed light on potential avenues for a defensive strategy against backdooring attacks.

The Destructive Effect Causes Low Confidence. We see that early exits increase the accuracy by up to 8% over the original CNN; especially on more difficult tasks. This hints that early exits, to a certain extent, can recover the accuracy lost to the destructive effect. However, the samples that trigger the destructive effect receive significantly lower correct prediction confidence— $\sim$0.3 on average vs. $\sim$0.6 in other correct predictions. These low confidence samples, as a result, pass up the early exits; even though they are correctly classified. The disagreements among internal predictions on these samples indicate that the network is *confused* while the predictions are evolving. The confusion reduces the benefit from early exits; however, it could also provide valuable insights into how the network reaches a prediction. Next, we explore the notion of network confusion further.

5.2 Network Confusion Analysis

An SDN's internal predictions reveal how consistently the network reaches its final prediction. Disagreements among them hint that the prediction is inconsistent and *confused*; whereas an agreement indicates consistency. The destructive effect of overthinking also displays a pattern of disagreement—$\hat{y}_i = y \neq \hat{y}_{final}$—, and confusion. We propose the *confusion* metric to capture this inconsistency. The confusion metric quantifies how much the final prediction diverged from the internal predictions. The divergence of the final prediction from an internal prediction is given by the L_1 distance between them, i.e. $\mathcal{D}_i(x) = ||\mathcal{F}_{final}(x) - \mathcal{F}_i(x)||_1$. We observe that, compared with other distance metrics, L_1 distance is more consistent for capturing the small differences between two predictions. The summation over all internal predictions, i.e. $\sum_{\forall i} D_i(x)$, gives the unbounded confusion score on the sample x. We normalize the unbounded scores using the mean and the standard variation of the training set samples.

Confusion Metric is an Error Indicator. Section 4.2 demonstrates the prevalence of the destructive effect on misclassifications. The confusion metric inherently captures this effect. As a result, we expect confusion scores to be a reliable indicator for the cases when the net-

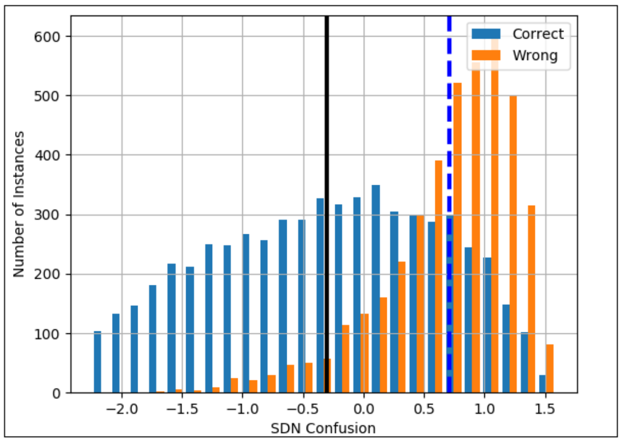

Figure 4: The distribution of the confusion scores *VGG-16-SDN* produces on Tiny ImageNet test samples. The dotted and straight lines indicate the average confusion scores in the wrong and correct predictions, respectively.

work misclassifies a sample. Such indications have practical significance for handling errors; for example, they can alert users about cases where the network is unable to make a good prediction. Here, we continue our case study on *VGG-16* and *VGG-16-SDN* on Tiny ImageNet task. Figure 4 presents *VGG-16-SDN*'s normalized confusion scores in correct and wrong final predictions on test samples. While correct predictions tend to have low confusion scores (-0.29 on average), the misclassifications are concentrated among instances with high confusion (0.71 on average)—more than one standard deviation difference. As a baseline, we consider the prediction confidence scores, i.e. $\max_j \mathcal{F}^{(j)}_{final}(x)$, *VGG-16* produces. The difference in confidence scores is less pronounced: 0.93 vs. 0.71 on the correct and wrong predictions, respectively—less than one standard deviation difference. Further, when used as an indicator for likely misclassifications, confusion also produces fewer false negatives than confidence. Compared to an average correct prediction, 5.5% of the misclassified instances (228 out of 4135) cause less confusion for the SDN; whereas 24% (980 out of 4135) misclassified instances obtain more confidence in the original CNN. Altogether, the SDN confusion metric is a reliable and non-expensive indicator of whether a misclassification is likely.

Visualizing Confusion Helps with Error Diagnosis. Towards error mitigation, our confusion metric can indicate whether the network is likely to make a mistake. As a further step, we propose visualizing the confusion for investigating sources of the destructive effect in the input. We hypothesize that the confusing elements in an input trigger disagreements and misclassifications. Specifically, given an input (x, y), we visualize the wrong final prediction and the disagreeing i^{th} correct internal prediction, i.e. $\hat{y}_i = y \neq \hat{y}_{final}$. We use Grad-CAM (Selvaraju et al., 2017) for visualizing the input elements that are influential in a prediction. In Figure 3, we presented some images that effectively confuse *VGG-16-SDN*. On these confusing images, the first IC, which predicts the correct labels, and the final classifier, which misclassifies, disagree. Figure 5 shows the Grad-CAM visualizations of the first IC and final classifier, on these images. We see that the first IC focuses on the general structure of the input, whereas the final classifier focuses on the fine details. As an example, consider the *banana* vs. *ice cream* image. Here, the first internal classifier focuses on the simple structure of the banana, which leads to a correct prediction. The confusing details, such as the exposed top part of the banana, mislead the final classifier into classifying the image as ice cream. Visualizing the confusion helps us to better understand how the network makes incorrect decisions; providing a new perspective towards interpretable deep learning.

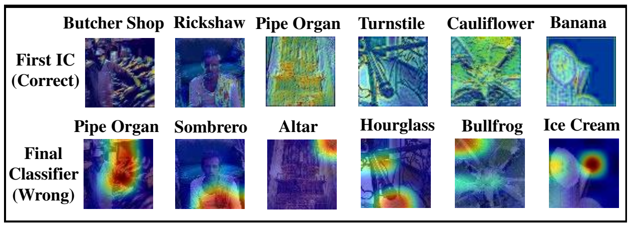

Figure 5: The visual interpretation of network confusion. Green color indicates the elements influential in the prediction, and blue color signifies less influence.

6 Conclusions

We introduce Shallow-Deep Networks (SDNs), a modification for obtaining accurate predictions from the internal layers of an off-the-shelf deep neural network (DNN). Our modification allows us to monitor how predictions evolve throughout the forward pass and provides a new way to understand DNNs. SDNs help us expose the prevalence of the *overthinking* problem in convolutional neural networks, which leads to wasted computation and to misclassifications on both natural and adversarial inputs. SDNs also enable new mechanisms for mitigating the problem. Confidence-based early exits significantly reduce the average inference cost while preserving the original performance. Moreover, our *confusion* analysis exposes disagreements among internal classifiers and reliably indicates the cases where the network is likely to misclassify an input. This paves the way for visually interpreting errors by revealing the input elements responsible for the confusion. We hope that our findings lay the foundations for more efficient and more accurate networks that are not susceptible to the prevalent overthinking problem.

Acknowledgements

We would like to thank Dr. Tom Goldstein, Dr. Nicolas Papernot, Virinchi Srinivas and Sarah Joseph for their valuable feedback. This research was partially supported by the Department of Defense.

References

Belilovsky, E., Eickenberg, M., and Oyallon, E. Greedy layerwise learning can scale to imagenet. *CoRR*, abs/1812.11446, 2018. URL `http://arxiv.org/abs/1812.11446`.

Bolukbasi, T., Wang, J., Dekel, O., and Saligrama, V. Adaptive neural networks for efficient inference. *arXiv preprint arXiv:1702.07811*, 2017.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. Ieee, 2009.

Fiske, S. T. and Taylor, S. E. *Social cognition: From brains to culture*. Sage, 2013.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL `http://arxiv.org/abs/1704.04861`.

Huang, F., Ash, J., Langford, J., and Schapire, R. Learning deep ResNet blocks sequentially using boosting theory. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2058–2067, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/huang18b.html`.

Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., and Weinberger, K. Q. Multi-scale dense convolutional networks for efficient prediction. *CoRR, abs/1703.09844*, 2, 2017.

Kahneman, D. *Thinking, fast and slow*. Farrar, Straus and Giroux / New York, 2011.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pp. 562–570, 2015.

Lee, C.-Y., Gallagher, P. W., and Tu, Z. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial Intelligence and Statistics*, pp. 464–472, 2016.

Lin, J., Rao, Y., Lu, J., and Zhou, J. Runtime neural pruning. In *Advances in Neural Information Processing Systems*, pp. 2181–2191, 2017.

Papernot, N. and McDaniel, P. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pp. 618–626, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Teerapittayanon, S., McDanel, B., and Kung, H. Branchynet: Fast inference via early exiting from deep neural networks. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pp. 2464–2469. IEEE, 2016.

Wang, J., Jia, R., Friedland, G., Li, B., and Spanos, C. One bit matters: Understanding adversarial examples as the abuse of redundancy. *arXiv preprint arXiv:1810.09650*, 2018.

Wang, X., Luo, Y., Crankshaw, D., Tumanov, A., Yu, F., and Gonzalez, J. E. Idk cascades: Fast deep learning by learning not to overthink. *arXiv preprint arXiv:1706.00885*, 2017.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.