MORE OR LESS: WHEN AND HOW TO BUILD NEURAL NETWORK ENSEMBLES

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural network models are applied to many tasks with rising complexity and data sizes. Researchers and practitioners seek to scale the representational power of these models by adding more parameters. Increasing parameters, though, requires additional critical resources such as memory and compute time leading to increased training and inference cost. Thus a consistent challenge is to obtain as high as possible accuracy within a parameter budget. As neural network designers navigate this complex landscape, they are guided by conventional wisdom. This comes from empirical studies of the design space. We identify a critical part of this design space that is not well-understood: That is how to decide between the alternatives of expanding a single network model or increasing the number of networks and using them together in an ensemble. We study this question in detail through extensive experimentation across various network architectures and data sets. Crucially, we provide a robust assessment, missing in previous studies, by fixing the number of parameters. We also consider a holistic set of metrics such as training time, inference time, and memory usage. Contrary to conventional wisdom, we show that when we perform a holistic and robust assessment, we uncover a wide design space, where ensembles not only provide better accuracy but also train faster, and deploy at a speed comparable to single convolutional networks with the same total number of parameters.

1 INTRODUCTION

Scaling capacity of deep learning models. Convolutional neural network models are becoming as accurate as humans on perceptual tasks and are now used to discover drugs, compress data, and automate gameplay (Pouyanfar et al., 2018). These models have grown rapidly in size, driven by two major trends: First, a rise in data complexity and sizes, which pushes for models with large number of parameters and layers (Szegedy et al., 2015; He et al., 2016; Huang et al., 2016; Shazeer et al., 2017). Second, the need for better accuracy in critical applications – such as self-driving cars and medical diagnosis – where, to achieve a target accuracy, model sizes are increased routinely (Grzywaczewski, 2017). This is especially pronounced in the areas of computer vision and natural language processing: Model sizes are three orders of magnitude larger than what they were just three years ago (Sanh et al., 2019). When we increase model sizes, we also increase the time, computation, and memory needed to train and deploy such models. It is a consistent challenge to synthesize or design models that maximize accuracy without requiring tons of resources (Lee et al., 2015b; Huang and Liu, 2017).

The question we explore in this paper is: Given a set of available resources, which maps to an amount of parameters we can use, how to design a convolutional neural network model?

The holistic design space is very complex. Designers of convolutional neural network models navigate a complex design landscape in order to address this question: To start off, they decide on a network architecture to use in their model. Then, they have to consider whether to use a single network or build an ensemble model with multiple networks. Additionally, they have to decide how many neural networks to use as well as their individual designs i.e., to find a desirable configuration of depth, width, and number of networks in their model. Modern applications with diverse requirements further complicate these decisions as what is desirable varies. Facebook, for instance, requires convolutional neural network models that strike specific tradeoffs between accuracy and inference time across 250 different types of smartphones (Wu et al., 2019). As a result, not just accuracy but a

diversity of metrics – such as inference time and memory usage – inform whether a model gets used (Sze et al., 2017b; Wu et al., 2019).

Scattered conventional wisdom. There are bits and pieces of scattered conventional wisdom to guide a neural network designer. These take the form of various empirical studies that demonstrate how depth and width in a single neural network model relate to accuracy and training time. First, it is generally known that deeper and wider networks can improve accuracy. In fact, recent convolutional architectures – such as ResNets and DenseNets – are designed precisely to enable this outcome (He et al., 2016; Huang and Liu, 2017; Huang et al., 2017). The caveat with beefing up a neural network is that accuracy runs into diminishing returns as we continue to add more layers or widen existing ones (Coates et al., 2011; Dauphin and Bengio, 2013). On the other hand, increasing the number of networks in the model, i.e., building ensembles, is considered to be a relatively robust but expensive approach to improve accuracy as ensemble models train and deploy k networks instead of one (Russakovsky et al., 2015; Wasay et al., 2020). The general consensus is to use ensembles when the goal is to achieve high accuracy without much regards to training cost, inference time, and memory usage e.g., competitions such as COCO and ImageNet (Lee et al., 2015a; Russakovsky et al., 2015; Huang et al., 2017; Ju et al., 2017). All these studies, however, exist in silos. Any form of cross-comparison is not possible as they use different data sets, network architectures, and hardware.

Lack of a robust and holistic assessment. Most empirical studies operate within the confines of a single convolutional network and do not consider the dimension of ensemble models. Those that do compare with ensembles, do so unfairly comparing ensembles with k networks against a single model that contains one such network. The problem with this is that the amount of parameters is not comparable, typically being k times more for ensembles, where k is the number of networks in the ensemble. In this way, such an analysis is an apples-to-oranges comparison and, as a result, it is not helpful in isolating the fundamental properties of ensembles versus single models.

In addition, past studies consider only accuracy and training cost. However, a holistic analysis needs to include also inference cost and memory usage, both of which are metrics that in practical settings are critical for applications (Sze et al., 2017a; Wu et al., 2019). The primary reason these metrics are excluded from the analysis is that the expectation is for an ensemble model to be k times slower for inference and k times more memory hungry compared to a single model.

Ensembles vs. single networks. In this paper, we bridge this gap in our understanding of the design space between single network models and ensemble models in a fair and holistic way. *Given modern applications with various requirements and constraints, should we train and deploy a convolutional model with a single network or one that contains an ensemble of networks? How should one approach the design of networks within an ensemble? As these constraints and requirements evolve, should we switch between these alternatives, why, and when?*

Method. We introduce the following methodology to accurately map the design space. Since there is no robust theoretical framework to consistently analyze the design space and the complex interactions among its many parameters and metrics, we develop a detailed and extensive experimental framework to isolate the impact of the critical design knobs: (i) depth, (ii) width, and (iii) number of networks, on all relevant metrics: (i) accuracy, (ii) training time, (iii) inference time, and (iv) memory usage. Crucially the number of parameters is a control knob in our framework and remains fixed across results that can be studied together. To establish the robustness of our findings, we experiment across various architectures, data complexities, and classification tasks. We present and analyze data amounting to over one year of GPU runtime. We also explain trends breaking down metrics into their constituents when necessary.

Results: The Ensemble Switchover Threshold (EST). (i) Contrary to conventional wisdom, we show that when we make a holistic and robust comparison between single convolutional networks and ensembles of networks, we discover a vast design space where ensembles provide not just better overall accuracy but also train faster compared to a single network. (ii) Specifically, we uncover the Ensemble Switchover Threshold (EST). This is the amount of resources (measured in terms of number of parameters and training epochs) beyond which ensembles provide superior generalization accuracy to a single model. (iii) We show that EST occurs consistently across various data sets and architectures. (iv) We demonstrate that the number of networks in an ensemble and their individual designs determine the EST. (v) Ensembles can also provide comparable inference times for a considerable part of the design space. (vi) We also show that for the same number of

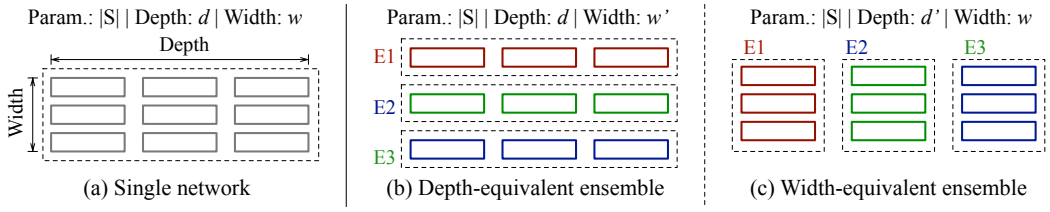


Figure 1: We explore a design space consisting of three design classes: (a) Single convolutional network models, (b) depth-equivalent ensembles, and (c) width-equivalent ensembles. The two ensemble design classes are created by distributing either the width factor or the depth corresponding to the single network amongst the ensemble networks while keeping the other factor fixed.

parameters, ensembles require significantly less memory to train and they are cheaper to train using cloud computing resources. (vii) Finally, we make available a web-based demo to enable visual exploration of our experimental results: `https://evd-demo.github.io`.

2 Defining the Design Space

The design space we explore consists of single convolutional neural network models and two classes of architecturally-homogenous ensembles. We first describe how we ensure a fair ranking within this design space and then explain the degrees of freedom we explore.

Establishing grounds for fair ranking. We rank various single network and ensemble model designs under equivalent resources. We do this by comparing designs having the same number of parameters. This allows us to separate the quality of the design itself from the amount of resources given to it. Crucially, a model designer with a certain amount of resources can use this resource-equivalent ranking to bootstrap their design process. We fix the number of parameters because of its two distinctive properties over other resource metrics (that could possibly be fixed) such as training time, inference time, or memory usage: First, the number of parameters of a network is directly proportional to all other resource-related metrics. Second, the number of parameters is independent of the hardware or the software platform used and can be computed exactly from a network specification.

The single network versus ensemble design space. We start with a single convolutional neural network architecture $S^{(w,d)}$ from a class of neural network architectures C having width factor w[1], depth d, and with $|S|$ number of parameters. We construct an ensemble $E = \{E_1 \dots E_k\}$ such that the total number of parameters it contains and the architecture class of its networks are the same as those of S i.e., $E_1 \dots E_k \in C$ and $|E_1| + \dots + |E_k| = |S|$. This ensemble is architecturally-homogenous i.e., all networks have the same architecture $E_1 = \dots = E_k$ and each ensemble network has $|S|/k$ parameters.

We construct ensembles in this way to reduce the intractably large space[2] of all possible ensembles given a single network to a size that we can feasibly and thoroughly experiment with and reason about. Furthermore, a considerable amount of neural network ensembles introduced in research as well as used in practice are similarly homogenous, for instance, SnapShot Ensembles and Fast Geometric Ensembles (Huang et al., 2017; Garipov et al., 2018). Additionally, our method provides a deterministic procedure of going between single network models and ensembles given a certain amount of parameters. Major sources of diversity in neural network ensembles are random weight initialization and stochastic training, both of which we incorporate in our experiments.

Depth-equivalent and width-equivalent ensembles. Convolutional neural network architectures are determined by two design knobs – the depth and the width factor. Corresponding to these two design knobs, we create two classes of ensembles: depth-equivalent ensembles and width-equivalent

[1] We use the term width factor (instead of width) to generalize to modern convolutional network architectures such as ResNets and DenseNets that don't have a fixed width (in terms of the number of convolutional filters) but expose knobs that determine how their width grows.

[2] Given a single network with $|S|$ number of parameters, there are $\left\{ {|S| \atop k} \right\}$ (Stirling number of the second kind) as many ways of forming ensembles of size k. This number grows at a similar rate to exponential polynomials (Boyadzhiev, 2009) e.g., $\left\{ {100 \atop 4} \right\} \approx 10^{59}$.

ensembles. These are depicted in Figure 1: In depth-equivalent ensembles the depth of the individual ensemble networks is the same as S (i.e., d) and the width factor is set to the highest possible value (i.e., w') without exceeding the parameter budget of $|S|$. In width-equivalent ensembles, on the other hand, the width factor is conserved across all ensemble networks (i.e., w) and the depth is modulated to the highest possible value (i.e., d') without exceeding $|S|$:

$$w' : k \cdot |E_i^{(w',d)}| \leq |S^{(w,d)}| \leq k \cdot |E_i^{(w'+1,d)}| \qquad d' : k \cdot |E_i^{(w,d')}| \leq |S^{(w,d)}| \leq k \cdot |E_i^{(w,d'+1)}|$$

It follows from the above definition that neural networks in depth-equivalent ensembles have higher depth than those in width-equivalent ensembles. Width-equivalent ensembles contain wider neural networks than their depth-equivalent counterparts. In this way, we isolate and study the effect of depth and width on ensemble accuracy and resource requirement.

Overall, our design space spans across three classes of convolutional neural network designs: (i) single network models, (ii) width-equivalent ensembles, and (iii) depth-equivalent ensembles. Every class contains several model designs that are instantiated by the four-tuple $\{w, d, |S|, C\}$. We design and conduct experiments across various configurations of these four-tuples.

3 Experimentation

Datasets and architectures. We include widely-used and state-of-the-art convolutional neural network architectures in our study: VGGNets, ResNets, DenseNets, and Wide ResNets (He at al., 2015; Zagoruyko and Komodakis, 2016; Huang and Liu, 2017). We also experiment on data sets with varying complexities and sizes: SVHN, CIFAR-10, CIFAR-100, and Tiny ImageNet (Krizhevsky, 2009; Netzer et al., 2011; Russakovsky et al., 2015).

Evaluation metrics. We evaluate single network and ensemble designs across five metrics: (i) generalization accuracy, (ii) training time per epoch, (iii) time to accuracy, (iv) inference time, and (v) memory usage. These metrics when considered together provide a holistic picture of the quality as well as the practicality of various designs.

Experimental setup. For all architectures, we adopt hyperparameters (including number of training epochs) from their respective papers. We experiment with various model sizes spanning the complete range of model sizes (in terms of number of parameters) reported in literature. Table A in the appendix summarizes these training details. We also ensure that we reach accuracies reported in literature for all architectures and sizes in our experiments. We implement our experimental framework in PyTorch. We use an Nvidia V100 GPU to run all our experiments.

4 Ensembles Outperform Single Network Models After a Low to Moderate Parameter Threshold

We observe that after a certain resource threshold, both classes of ensembles – depth- and width-equivalent – outperform single network models. We call this threshold the Ensemble Switchover Threshold (EST). Beyond the EST, ensemble models achieve 1 to 3 percent lower absolute test error rates (across various architectures and data sets) when compared with single network models having the same number of parameters.

To the best of our knowledge, this is the first observation of this phenomenon. The EST appears consistently across a wide range of data sets and architectures (Figure 2(a) through Figure 2(f)) as well as ensemble sizes (Figure 3(a) through Figure 3(c) and appendix Figure B(a) through Figure B(c)). In these figures, we use discrete heat maps to visualize which of the three design classes – single network models (single), depth-equivalent ensembles (deq), and width-equivalent ensembles (weq) – dominates in terms of generalization accuracy for a given resource budget. This resource budget takes the concrete form of number of parameters (on the x-axis) and number of epochs (on the y-axis). We also mark areas, where both classes of ensembles outperform single network models. For all experiments, Figure A in the appendix shows the test error rates achieved by various models.

The occurrence of EST both expands and questions the general consensus on the relative effectiveness of ensemble versus single network models. First, even when allocated the same amount

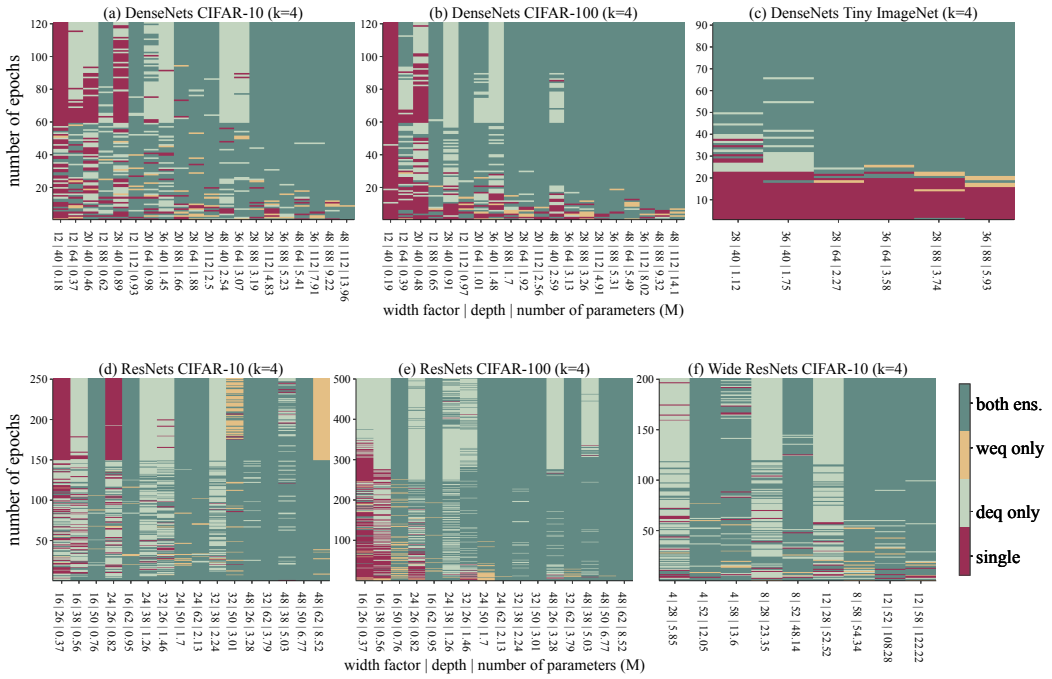


Figure 2: The Ensemble Switchover Threshold (EST) occurs consistently across various network architectures and data sets. Beyond this resource threshold, ensemble designs outperform single network models.

of resources, ensemble models still outperform single network models. This expands upon past empirical studies that only show how a k-network ensemble is more accurate than any of the single network models that it contains (Lee et al., 2015a; Huang et al., 2017). Second, the EST occurs in low- to moderate-resource settings. For instance, in all of our experiments, we observe the EST at the 1M to 1.5M parameter range[3] and after no later than half of the training epochs. This challenges the widespread notion that ensembles of neural networks are useful only when we have tons of resources at our disposal (Lee et al., 2015a; Ju et al., 2017). Overall, our results indicate that ensembles of convolutional models are preferable to single network models for a much wider range of use cases than previously understood.

We parse out how the data complexity and the composition of the ensemble networks affect the EST and in turn the ranking of the three design classes:

Ensembles are even more effective for more complex data sets. We observe that the EST shifts closer to the origin as the complexity of the training data set increases. This can be seen in Figure 2(a) through 2(c) where we train DenseNets on progressively more complex data sets (CIFAR-10, CIFAR-100, and Tiny ImageNet). This indicates that ensemble models are preferable to single models, when training on more complex data sets for an even wider range of available resources. This observation again expands the utility of ensembles. There is theoretical as well as empirical work establishing that ensembles do better for complex data (Bonab and Can, 2017; Huang et al., 2017). We, however, establish this phenomenon in the resource-equivalent setting as opposed to past studies that do so for ensembles and single networks with drastically different number of parameters.

Large ensembles are effective under a large parameter budget. As we increase the number of networks k within an ensemble without increasing the parameter budget, the overall accuracy of ensemble designs diminishes pushing the EST to a higher resource limit. Figure 3 demonstrates this phenomenon for DenseNets and Figure B in the appendix shows it for ResNets. For instance, for DenseNet models, the EST moves from the 1.5M range for $k = 4$ to the 3M range for $k = 6$

[3] Wide ResNets are an exception here: This is because, compared to other convolutional architectures, Wide ResNets have an order of magnitude more parameters even for modest depths and width factors. For instance, networks presented in the Wide ResNet paper have anywhere between 8M to 37M parameters compared to the 1M to 10M range for DenseNets (Zagoruyko and Komodakis, 2016).

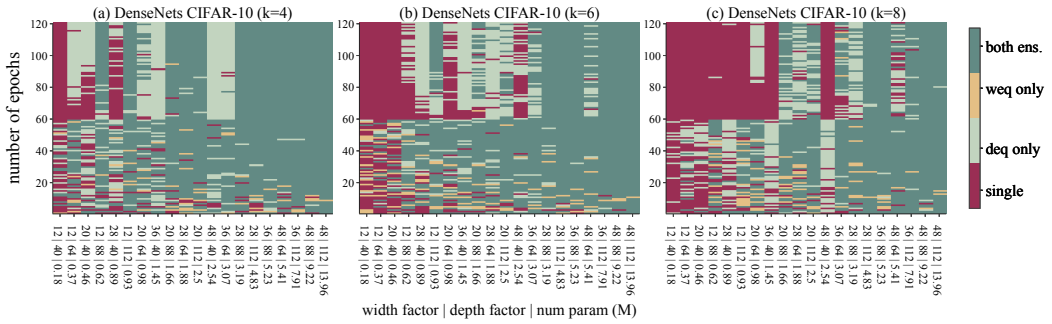
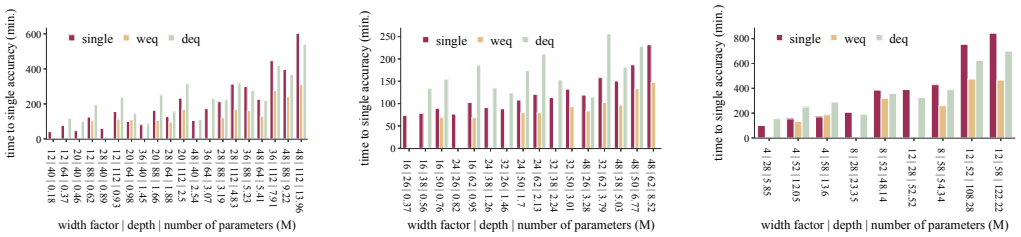


Figure 3: The Ensemble Switchover Threshold moves to the right as we increase the number of networks in the ensemble.



(a) DenseNets CIFAR-10 (k=4) (b) ResNets CIFAR-10 (k=4) (c) Wide ResNets CIFAR-10 (k=4)

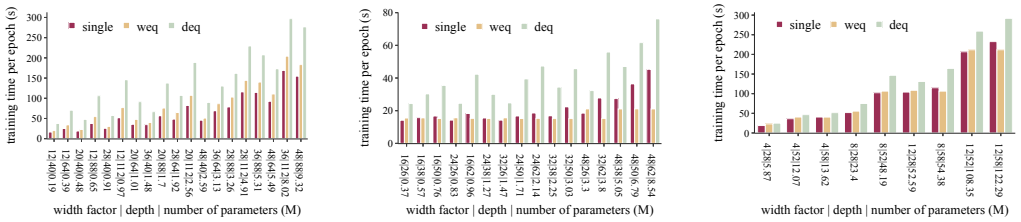
Figure 4: When ensemble designs can provide better accuracy, they can also do so faster than single network models (missing bars indicate that designs cannot reach single network model accuracy).

and, then, to the 5M range for $k = 8$. This can be explained by looking at individual accuracies of ensemble networks. Figure C in the appendix shows test error rates of the ensemble as a whole as well as the average test error rates of individual ensemble networks. We observe that as we increase the number of networks k, their individual test error rates (shown as dotted lines) increase along with the ensemble accuracy. This happens because the size of individual networks goes down (as we keep a fixed parameter budget). This implies that ensembles of larger size are desirable over those of smaller sizes only when we have a sufficient parameter budget to assign to every single network in the model. As opposed to previous work, our experiments decouple the parameter budget from the number of networks in the model. We discover that just increasing the ensemble size without increasing the total number of parameters does not result in improved accuracy.

Depth-equivalent ensembles outperform width-equivalent ensembles. We observe that depth-equivalent ensembles are overall more accurate than width-equivalent ensembles (as shown in Figure A in the appendix). They also consistently demonstrate EST at a lower resource range. This can be explained by the fact that modern convolutional neural network architectures are designed to provide better accuracy with increasing depth. Depth-equivalent ensembles have deeper ensemble networks with better individual accuracy. When used together in an ensemble they also provide better ensemble accuracy. When designing ensemble models for high accuracy, deeper networks are preferable to wider networks. Our observation also corroborates past empirical studies which show that deeper networks provide better accuracy keeping all other factors constant (Eigen et al., 2013; Urban et al., 2016). This fact holds also for ensembles.

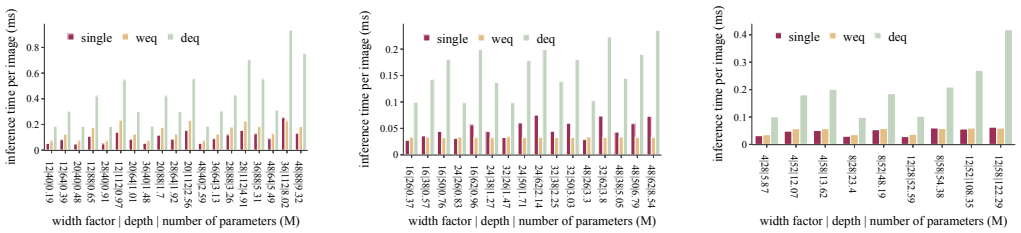
5 Ensembles Train Faster and Provide Comparable Inference Time

First, we analyze the training time. Despite taking longer per epoch, both ensemble classes achieve the accuracy of single network models significantly faster for a considerable part of the design space (e.g., $1.2\times$ to $5\times$ faster across our experiments). This happens after the EST has been reached i.e., when ensemble designs can provide better accuracy, they can also do so faster than single network models. This can be seen in Figure 4. Here, we plot the total training time needed for any of the three



(a) DenseNets CIFAR-10 (k=4) (b) ResNets CIFAR-10 (k=4) (c) Wide ResNets CIFAR-10 (k=4)

Figure 5: Depth-equivalent ensembles take longer to train per epoch as compared to single network models. Width-equivalent ensembles take comparable time.



(a) DenseNets CIFAR-10 (k=4) (b) ResNets CIFAR-10 (k=4) (c) Wide ResNets CIFAR-10 (k=4)

Figure 6: Width-equivalent ensembles take comparable time to single network models for inference. Depth-equivalent ensembles take significantly longer.

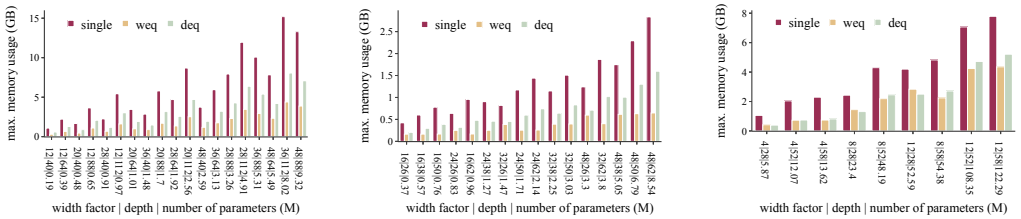
design classes to achieve the maximum accuracy provided by single network models under the same parameter budget. Figure 5 shows the corresponding training time per epoch.

The combined depth determines per epoch training time. We observe that both classes of ensembles, on average, take longer to train per epoch as compared to single network models as they train k networks instead of one. How much more time ensembles take per epoch depends heavily on the design of the ensemble networks: This additional time is negligible for width-equivalent ensembles whereas, for depth-equivalent ensembles, it results in $2\times$ more expensive per epoch training. This trend can be explained by how the training time per epoch scales with respect to the width and depth of convolutional neural network models. This ultimately connects to GPU capabilities.

We break down the training time per epoch of all designs into two constituents: time spent per layer and number of layers. Figure E in the appendix shows this breakdown for various architectures. We observed that the total number of layers in a model (for ensembles this is the sum of all networks' depth) majorly determines the training time per epoch. For the same parameter budget, depth-equivalent ensembles have proportionally more layers, whereas width-equivalent ensembles have proportionally more width. The average time per layer depends on the width and does not increase significantly as we move from depth-equivalent ensembles to width-equivalent ensembles. On the other hand, the total number of layers scale linearly with depth. For the same parameter budget, total number of layers are significantly higher for depth-equivalent ensembles as compared to the other two designs, resulting in higher per epoch training.

From a GPU computing perspective, for the same parameter budget, wider and shallower networks are more efficient to execute than narrower and deeper networks. This can be attributed to the massive amount of data parallelism in modern GPUs. Increasing the width of the network just increases the width of the kernels whereas, deepening a network introduces new operations that require additional synchronization steps.

Networks in ensemble models converge faster than single network models for the same parameter budget. The fact that ensemble designs can reach the same accuracy faster than single network models can be attributed to the fact that, for the same parameter budget, all networks in the ensemble model are smaller than the single network model. Smaller networks are known to converge faster albeit to a lower accuracy than larger networks. However, we observe that the distinct advantage ensemble designs provide over the single model is that when we use smaller networks in an ensemble, we get the best of both worlds. We converge faster at an individual network level and ensembling makes up for the generalization accuracy.



(a) DenseNets CIFAR-10 (k=4) (b) ResNets CIFAR-10 (k=4) (c) Wide ResNets CIFAR-10 (k=4)

Figure 7: Both classes of ensemble models are significantly more memory efficient.

Overall, our observations, again question the conventional wisdom of ensembles being significantly slower to train as compared to single network models. When we analyze the design space under a fixed parameter budget, we uncover that for a vast range of the design space: (i) width-equivalent ensembles introduce negligible overhead to per epoch training time as compared to single network models and (ii) both ensemble designs achieve and surpass accuracy of single network models in considerably less training time.

Width-equivalent ensembles provide competitive inference time. We provide the inference time per image in Figure 6 and observe a similar trend to training time per epoch. While depth-equivalent ensembles are significantly slower, width-equivalent ensembles provide comparable inference speed to single network models. Again, this questions conventional wisdom that expects ensembles to be substantially slower in inference.

6 Ensembles are Memory-Efficient

When we look at the metric of memory usage, we observe that the trend favors both classes of ensemble designs over single network models. Figure 7 provides the amount of memory used as we train depth-equivalent ensembles, width-equivalent ensembles, and single network models. This is the minimum amount of memory that a GPU needs to have to train any of these designs for the batch sizes provided in Table A. This memory is majorly used to store model parameters and intermediate results (Jain et al., 2020).

The superior memory-efficiency of ensemble models is due to the fact that when we train an ensemble of k networks, at any point in time, we only need as much memory to train only one of the k networks. This cost when compared with single networks with the same number of parameters is significantly less. This memory-efficiency has two important implications: First, for the same GPU, we can use higher batch sizes while training an ensemble of networks. This, for instance, is useful when training complex data sets such as ImageNet (Smith et al., 2018). Additionally, we can feasibly train the same number of parameters in an ensemble using GPUs with less memory. In the world of cloud computing, this translates to drastic reduction in costs.

Additional results. We show that the same resource-related trends hold for the rest of architectures and data sets from Table A. Figure D, Figure F, Figure G, and Figure H demonstrate these trends for metrics of time to accuracy, time per epoch, inference time, and memory usage respectively. We also provide results on the SVHN data set in Figure I in the appendix. Table B through Table D provide the conversion between the width factor and depth of single network models and the two ensemble design classes.

7 The Complete Picture

The results in this paper question conventional wisdom on convolutional model design with respect to the design decision on whether to use an ensemble of networks or not. By creating a detailed framework that (a) allows to fix resources, and (b) spans a large design space, we show that for a considerable part of the design space, given an amount of resources, ensembles (i) achieve better accuracy than single network models, (ii) train faster, (iii) provide comparable inference, and (iv) need much less memory. Future work includes the addition of fast ensemble training approaches (such as SnapShot and MotherNets (Huang et al., 2017; Wasay et al., 2020)) and to also consider parallel training, both of which can move EST further in favor of ensembles.

REFERENCES

H. R. Bonab and F. Can. Less is more: A comprehensive framework for the number of components of ensemble classifiers. *arXiv preprint arXiv:1709.02925*, 2017.

K. N. Boyadzhiev. Exponential polynomials, stirling numbers, and evaluation of some gamma integrals. In *Abstract and Applied Analysis*. Hindawi, 2009.

A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

Y. N. Dauphin and Y. Bengio. Big neural networks waste capacity. *arXiv preprint arXiv:1301.3583*, 2013.

D. Eigen, J. Rolfe, R. Fergus, and Y. LeCun. Understanding deep architectures using a recursive convolutional network. *arXiv preprint arXiv:1312.1847*, 2013.

T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, pages 8789–8798, 2018.

A. Grzywaczewski. Training ai for self-driving vehicles: the challenge of scale, 2017.

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

K. He at al. Delving deep into rectifiers. In *IEEE international conference on computer vision*, pages 1026–1034, 2015.

G. Huang and Z. Liu. Densely connected convolutional networks. 2017.

G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661. Springer, 2016.

G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

P. Jain, A. Jain, A. Nrusimha, A. Gholami, P. Abbeel, K. Keutzer, I. Stoica, and J. Gonzalez. Checkmate: Breaking the memory wall with optimal tensor rematerialization. 2020.

C. Ju, A. Bibaut, and M. J. van der Laan. The relative performance of ensemble methods with deep convolutional neural networks for image classification. *arXiv preprint arXiv:1704.01664*, 2017.

A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015a.

S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015b.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.

V. Sze, Y. Chen, J. S. Einer, A. Suleiman, and Z. Zhang. Hardware for machine learning: Challenges and opportunities. In *2017 IEEE Custom Integrated Circuits Conference, CICC 2017, Austin, TX, USA, April 30 - May 3, 2017*, pages 1–8, 2017a. doi: 10.1109/CICC.2017.7993626. URL https://doi.org/10.1109/CICC.2017.7993626.

V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017b.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. URL http://arxiv.org/abs/1409.4842.

G. Urban, K. J. Geras, S. E. Kahou, O. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson. Do deep convolutional nets really need to be deep and convolutional? *arXiv preprint arXiv:1603.05691*, 2016.

A. Wasay, B. Hentschel, Y. Liao, S. Chen, and S. Idreos. Mothernets: Rapid deep ensemble learning. In *Proceedings of the Conference on Machine Learning and Systems (MLSys)*, 2020.

C.-J. Wu, D. Brooks, K. Chen, D. Chen, S. Choudhury, M. Dukhan, K. Hazelwood, E. Isaac, Y. Jia, B. Jia, et al. Machine learning at facebook: Understanding inference at the edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 331–344. IEEE, 2019.

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

APPENDIX

Table A: For all networks, we use training hyperparameters listed in their respective papers (wd: weight decay).

Architecture	Data sets	Epochs	Lr schedule	batch size	Wd
DenseNet	C10 and C100	120	0.1, 0.01(60), 0.001(90)	128	0.001
DenseNet	Tiny ImageNet	90	0.1, 0.01(30), 0.001(60)	64	0.001
DenseNet	SVHN	100	0.1, 0.01(30), 0.001(60)	128	0.001
ResNet	C10	250	0.1, 0.01(100), 0.001(200)	128	0.0001
ResNet	C100	500	0.1, 0.01(250), 0.001(375)	128	0.0001
Wide ResNet	C10 and C100	200	0.1, 0.01(100), 0.001(200)	128	0.0001

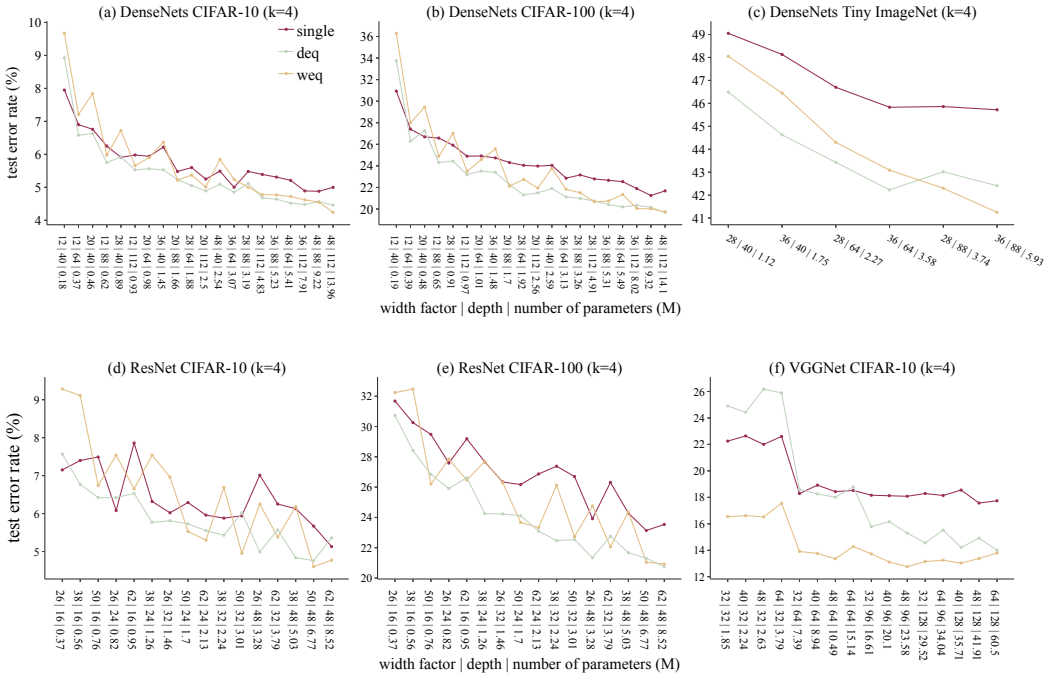


Figure A: Ensembles arrive at lower test error rates than single network models after the EST has been reached.

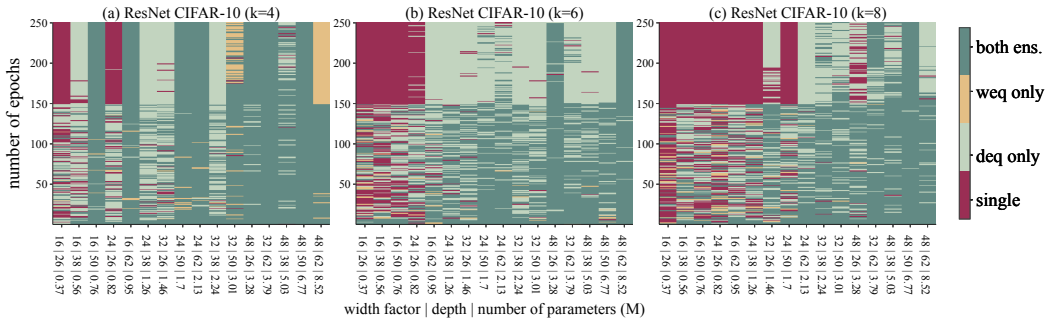


Figure B: The Ensemble Switchover Threshold moves to the right as we increase the number of networks in the ensemble. Here, we demonstrate this phenomenon for ResNet models.

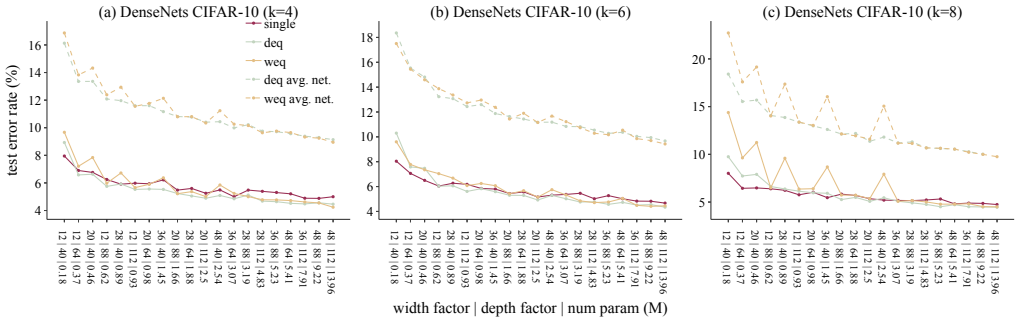
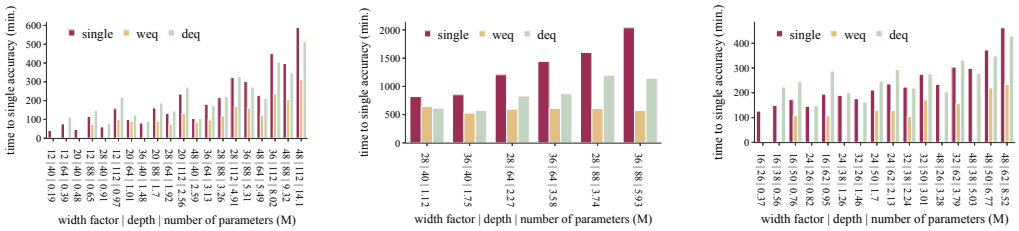
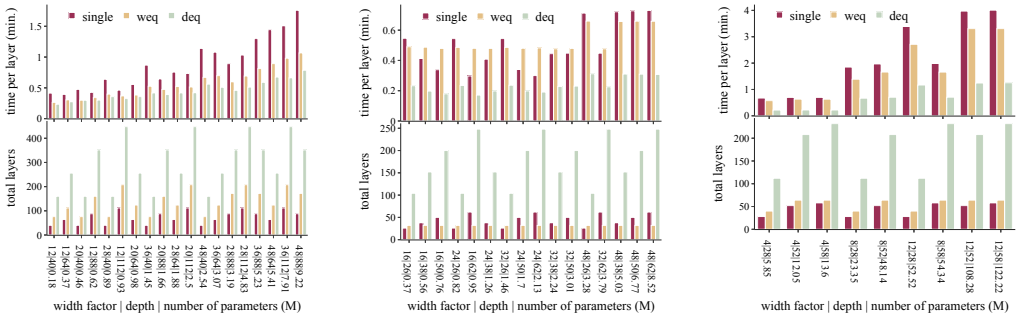


Figure C: As we increase the size of ensembles, accuracy of individual networks in the ensemble decreases. This results in an overall reduction in ensemble accuracy shifting the EST to the high-resource space.



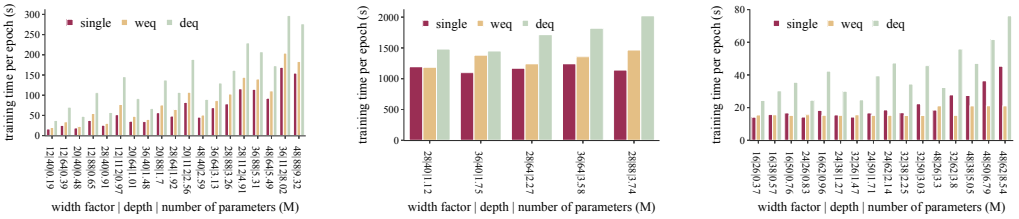
(a) DenseNets CIFAR-100 (k=4) (b) DenseNets Tiny ImageNet (k=4) (c) ResNets CIFAR-100 (k=4)

Figure D: When ensemble designs can provide better accuracy, they can also do so faster than single network models (missing bars indicate that designs cannot reach single network model accuracy).



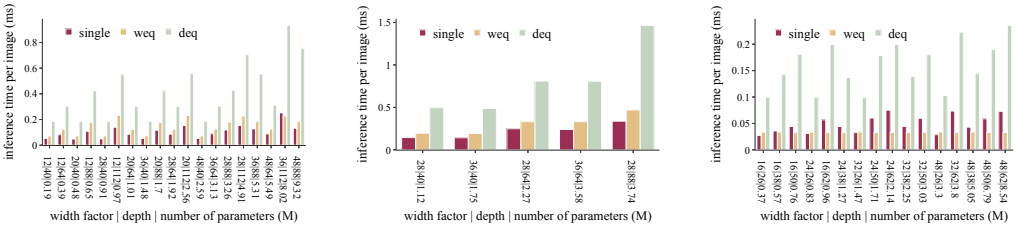
(a) DenseNets on CIFAR-10 (b) ResNets on CIFAR-10 (c) Wide ResNets on CIFAR-10

Figure E: We break down per epoch training time into: (i) time spent per layer and (ii) total number of layers. We observe that the total number of layers in the model more significantly determines the per epoch training time as compared to the width.



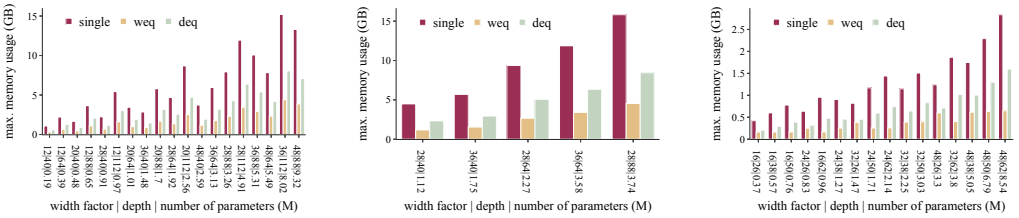
(a) DenseNets on CIFAR-100 (b) DenseNets on Tiny ImageNet (c) ResNets on CIFAR-100

Figure F: Depth-equivalent (deq) ensembles take longer to train per epoch as compared to single network models. Width-equivalent ensembles (weq) take comparable time.



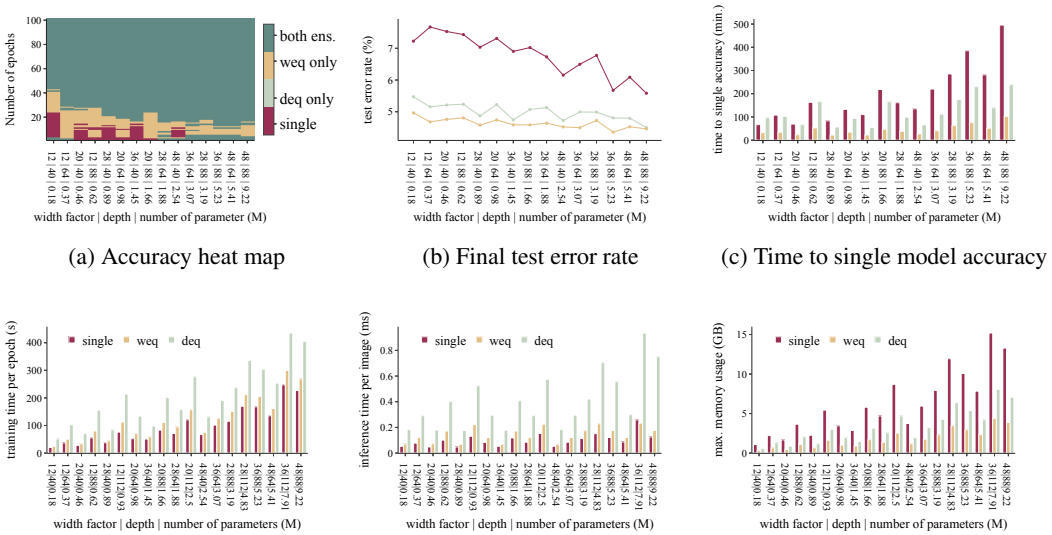
(a) DenseNets on CIFAR-100 (b) DenseNets on Tiny ImageNet (c) ResNets on CIFAR-100

Figure G: Ensemble models take longer to perform inference.



(a) DenseNets on CIFAR-10 (b) DenseNets on Tiny ImageNet (c) ResNets on CIFAR-100

Figure H: Ensemble models are significantly more memory efficient.



(a) Accuracy heat map (b) Final test error rate (c) Time to single model accuracy

(d) Training time per epoch (e) Inference time per image (f) Memory usage

Figure I: We observe similar accuracy and resource related trends for DenseNets trained on the SVHN data set.

Table B: DenseNet

single depth	single width factor	Weq depth	Deq width factor
40	12	19	5
40	20	19	9
40	28	19	13
40	36	19	17
40	48	19	23
64	12	28	6
64	20	31	10
64	28	31	14
64	36	31	18
64	48	31	24
88	12	40	6
88	20	40	10
88	28	43	14
88	36	43	18
88	48	43	24
112	12	52	6
112	20	52	10
112	28	52	14
112	36	52	18

Table C: ResNet

single depth	single width factor	Weq depth	Deq width factor
26	16	8	8
26	24	8	12
26	32	8	16
26	48	8	24
38	16	8	8
38	24	8	12
38	32	8	16
38	48	8	24
50	16	8	8
50	24	8	12
50	32	8	16
50	48	8	24
62	16	8	8
62	24	8	12
62	32	8	16
62	48	8	24

Table D: WideResNet

single depth	single width factor	Weq depth	Deq width factor
28	4	10	1
28	8	10	3
28	12	10	5
52	4	16	1
52	8	16	3
52	12	16	5
58	4	16	1
58	8	16	3
58	12	16	5