Anytime Inference with Distilled Hierarchical Neural Ensembles

Adria Ruiz[1], Jakob Verbeek[2]

[1] Institut de Robòtica i Informàtica Industrial, CSIC-UPC
[2] Facebook AI Research
aruiz@iri.upc.edu, jjverbeek@fb.com

Abstract

Inference in deep neural networks can be computationally expensive, and networks capable of anytime inference are important in scenarios where the amount of compute or input data varies over time. In such networks the inference process can interrupted to provide a result faster, or continued to obtain a more accurate result. We propose Hierarchical Neural Ensembles (HNE), a novel framework to embed an ensemble of multiple networks in a hierarchical tree structure, sharing intermediate layers. In HNE we control the complexity of inference on-the-fly by evaluating more or less models in the ensemble. Our second contribution is a novel hierarchical distillation method to boost the predictions of small ensembles. This approach leverages the nested structure of our ensembles, to optimally allocate accuracy and diversity across the individual models. Our experiments show that, compared to previous anytime inference models, HNE provides state-of-the-art accuracy-computation trade-offs on the CIFAR-10/100 and ImageNet datasets.

Introduction

Deep learning models typically require a large amount of computation during inference, limiting their deployment in edge devices such as mobile phones or autonomous vehicles. For this reason, methods based on network pruning (Huang et al. 2018b), architecture search (Tan et al. 2019), as well as manual network design (Sandler et al. 2018), have all been used to find more efficient model architectures. Despite the promising results achieved by these approaches, there exist several applications where, instead of deploying a single efficient network, we are interested in dynamically adapting the inference latency depending on external constraints. Examples include scanning of incoming data streams on online platforms or autonomous driving, where either the amount of data to be processed or the number of concurrent processes is non-constant. In these situations, models must be able to scale the number of the operations on-the-fly depending on the amount of available compute at any point in time. In particular, we focus on methods capable of anytime inference, *i.e.* methods where the inference process can be interrupted for early results, or continued for more accurate results (Huang et al. 2018a). This contrasts with other
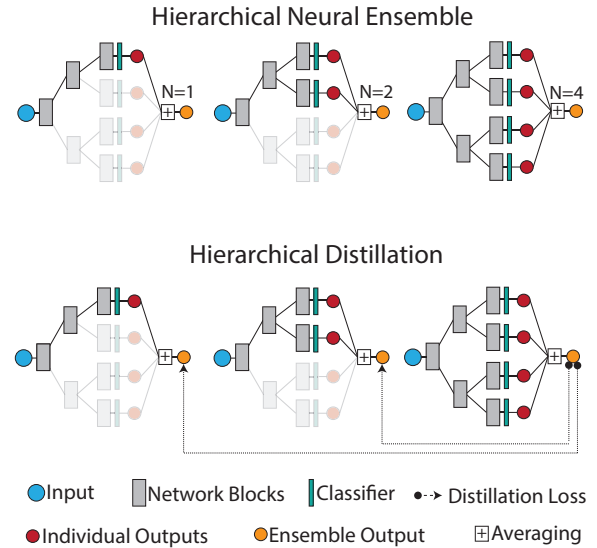
Figure 1: (Top) HNE shares parameters and computation in a hierarchical manner. Tree leafs represent separate models in the ensemble. Anytime inference is obtained via depth-first traversal of the tree, and using at any given time the ensemble prediction of the N models evaluated so far. (Bottom) Hierarchical distillation leverages the full ensemble to supervise parts of the tree that are used in small ensembles.

methods, where the accuracy-speed trade-off has be decided before the computation for inference starts.

We address this problem by introducing Hierarchical Neural Ensembles (HNE). Inspired by ensemble learning (Breiman 1996), HNE embeds a large number of networks whose combined outputs provide a more accurate prediction than any individual model. To reduce the computational cost of evaluating the networks, HNE employs a binary-tree structure to share a subset of intermediate layers between the different models. This scheme allows to control the inference complexity by deciding how many networks to use, *i.e.* how many branches of the tree to evaluate. To train HNE, we propose a novel distillation method adapted to its hierarchical structure. See Figure 1 for an overview of our approach.

Our contributions are summarised as follows: (i) To the

best of our knowledge, we are the first to explore hierarchical ensembles for deep models with any-time prediction. (ii) We propose a hierarchical distillation scheme to increase the accuracy of ensembles for adaptive inference cost. (iii) Focusing on image classification, we show that our framework can be used to design efficient CNN ensembles. In particular, we evaluate the different proposed components by conducting ablation studies on CIFAR-10/100 datasets. Compared to previous anytime inference methods, HNE provides state-of-the-art accuracy-speed trade-offs on the CIFAR datasets as well as the more challenging ImageNet dataset.

Related Work

Efficient networks. Different approaches have been explored to reduce the inference complexity of deep neural networks. These include the design of efficient convolutional blocks (Howard et al. 2017; Ma et al. 2018; Sandler et al. 2018), neural architecture search (NAS) (Tan et al. 2019; Wu et al. 2019; Cai et al. 2020; Yu et al. 2020) and network pruning techniques (Huang et al. 2018b; Liu et al. 2017). In order to adapt the inference cost, other methods have proposed different mechanisms to reduce the number of feature channels (Yu and Huang 2019; Yu et al. 2018) or skip intermediate layers in a data-dependent manner (Veit and Belongie 2018; Wang et al. 2018; Wu et al. 2018). Whereas these approaches are effective to reduce the resources required by a single network, the desired speed-accuracy trade-off need to be selected before the inference process begins.

Anytime inference. In order to provide outputs at early inference stages, previous methods have considered to introduce intermediate classifiers on hidden network layers (Bolukbasi et al. 2017; Elbayad et al. 2020; Huang et al. 2018a; Li et al. 2019; Zhang, Ren, and Urtasun 2019). In particular, (Huang et al. 2018a) proposed a Multi Scale DenseNet architecture (MSDNet) where early-exit classifiers are used to compute predictions at any point during evaluation. More recently, MSDNets have been extended by using improved training techniques (Li et al. 2019) or exploiting multi-resolution inputs (Yang et al. 2020). On the other hand, Convolutional Neural Mixtures (Ruiz and Verbeek 2019) proposed a densely connected network that can be dynamically pruned. Finally, early-exits have been also combined with NAS by Zhang, Ren, and Urtasun (2019) to automatically find the optimal position of the classifiers.

Different from previous works relying on early-exit classifiers, we address anytime inference by exploiting hierarchical network ensembles. Additionally, our framework can be used with any base model in contrast to previous approaches which require specific network architectures (Huang et al. 2018a; Ruiz and Verbeek 2019). Therefore, our method is complementary to approaches relying on manual design, neural architecture search, or network pruning.

Network ensembles. Ensemble learning is a classic approach to improve generalization (Hansen and Salamon 1990; Rokach 2010). The success of this strategy relies on the reduction in variance resulting from averaging the output of different learned predictors (Breiman 1996). Seminal works (Hansen and Salamon 1990; Krogh and Vedelsby 1995; Naftaly, Intrator, and Horn 1997; Zhou, Wu, and Tang

2002) observed that a significant accuracy boost could be achieved by averaging the outputs of independently trained networks. Recent deep CNNs have also been shown to benefit from this strategy (Geiger et al. 2020; Ilg et al. 2018; Lan, Zhu, and Gong 2018; Lee and Chung 2020; Lee et al. 2015; Malinin, Mlodozeniec, and Gales 2020). A main limitation of deep network ensembles, however, is the linear increase in training and inference costs with the number of models in the ensemble. Whereas some strategies have been proposed to decrease the training time (Huang et al. 2017; Loshchilov and Hutter 2017), the high inference cost still remains as a bottleneck in scenarios where computational resources are limited. In this context, different works (Lan, Zhu, and Gong 2018; Lee et al. 2015; Minetto, Segundo, and Sarkar 2019) have proposed to build ensembles where the individual networks share a subset of parameters in order to reduce the inference cost.

Building on these ideas, our HNE uses a binary-tree structure to share intermediate layers between individual networks. Whereas hierarchical structures have been explored for different purposes, such as learning expert mixtures (Liu et al. 2019; Tanno et al. 2019; Kim et al. 2017), incremental learning (Roy, Panda, and Roy 2020), or ensemble implementation (Lee et al. 2015; Zhang et al. 2018), our work is the first to leverage this structure for anytime inference.

Diversity in network ensembles. Ensemble performance is affected by two factors (Ueda and Nakano 1996): the accuracy of each individual model, and the variance among the model predictions. Different works encourage model diversity by sub-sampling training data during optimization (Lakshminarayanan, Pritzel, and Blundell 2017; Lee et al. 2015) or using regularization mechanisms (Chen and Yao 2009). Using these strategies, however, the performance of each individual model is significantly reduced (Lee et al. 2015). For this reason, we instead use a simple but effective strategy to encourage model diversity. In particular, we train our HNE by using a different initialization for the parameters of each parallel branch in the tree structure. Previous work (Huang et al. 2017; Neal et al. 2018) has shown that networks trained from different initializations exhibit a significant variance within their predictions.

Knowledge distillation. The accuracy of a low-capacity "student" network can be improved by training it on soft-labels generated by a high-capacity "teacher" network, rather than directly on the training data (Ba and Caruana 2014; Hinton, Vinyals, and Dean 2015; Romero et al. 2015). Knowledge distillation from the activation of intermediate network layers (Sun et al. 2019), and from soft-labels provided by one or more networks with the same architecture as the student (Furlanello et al. 2018) have also been shown to be effective. In co-distillation (Lan, Zhu, and Gong 2018; Zhang et al. 2018; Bhardwaj et al. 2019) the distinction between student and teacher networks is lost and, instead, models are jointly optimized and distilled online. For network ensembles, co-distillation has been shown effective to improve the accuracy of the individual models by transferring the knowledge from the full ensemble (Anil et al. 2018; Song and Chai 2018).

To improve the performance of HNE, we introduce hier-

archical distillation. Different from existing co-distillation strategies (Lan, Zhu, and Gong 2018; Song and Chai 2018), our approach transfers the knowledge from the full model to smaller sub-ensembles in a hierarchical manner. Our approach is specifically designed for neural ensembles, where the goal is not only to improve predictions requiring a low inference cost, but also to preserve the diversity between the individual network outputs.

Hierarchical Neural Ensembles

HNE embed an ensemble of deep networks computing an output $\mathbf{y}^E = \frac{1}{N} \sum_{n=1}^N F_n(\mathbf{x}; \theta_n)$ from an input $\mathbf{x}$, where F_n is a network with parameters θ_n and N is the total number of models. Furthermore, we assume that each network is a composition of $B + 1$ functions, or "blocks" as $F_n(\mathbf{x}; \theta_n) = f_{\theta_n^B} \circ \cdots \circ f_{\theta_n^1} \circ f_{\theta_n^0}(\mathbf{x})$, where each block $f_{\theta_n^b}(\cdot)$ is a set of layers with parameters θ_n^b. Typically, $f_{\theta_n^b}(\cdot)$ contains operations such as convolutions, batch normalization layers, and activation functions.

Hierarchical sharing of parameters and computation. If we use different parameters θ_n^b for all blocks and networks, then the set of the ensemble parameters is given by $\Theta = \{\theta_{1:N}^0, \theta_{1:N}^1, \ldots, \theta_{1:N}^B\}$, and the inference cost of computing the ensemble output is equivalent to evaluate N independent networks. In order to reduce the computational complexity, we design HNE to share parameters and computation employing a binary tree structure, where each node of the tree represents a computational block. Each of the $N = 2^B$ paths from the root to a leaf represents a different model composed of $B + 1$ computational blocks. The first (root) computational block is shared among all models, and after each block the computational path is continued along two branches, each with a different set of parameters from the next block onward. See Figure 1 for an illustration. Therefore, for each block b there are $K = 2^b$ independent sets of parameters θ_k^b. The parameters of an HNE composed of $N = 2^B$ models are collectively denoted as $\Theta = \{\theta_1^0, \ldots, \theta_{1:2^b}^b, \ldots, \theta_{1:2^B}^B\}$.

Anytime inference. The success of ensemble learning is due to the reduction in variance by averaging the predictions of different models. The expected resulting improvements in accuracy are therefore monotonic in the number of models in the ensemble. Given that the models in the ensemble can be evaluated sequentially, the speed-accuracy trade-off can be controlled by choosing how many models to evaluate to approximate the full ensemble output. In the case of HNE, this is achieved by evaluating only a subset of the paths from the root to the leafs, see Figure 1. More formally, we can choose any value $b \in \{0, 1, \ldots, B\}$ and compute the ensemble output using a subset of $N' = 2^b$ networks as

$$\mathbf{y}^b = \frac{1}{2^b} \sum_{n=1}^{2^b} F(\mathbf{x}; \theta_n). \tag{1}$$

The evaluated subset of $N' = 2^b$ networks is obtained by traversing the binary tree structure in a depth-first manner, where the first leaf model is always the same. Thus, we evaluate the first branch, as well as all the other $2^b - 1$ branches that share the last b blocks with this branch. See Figure 1.
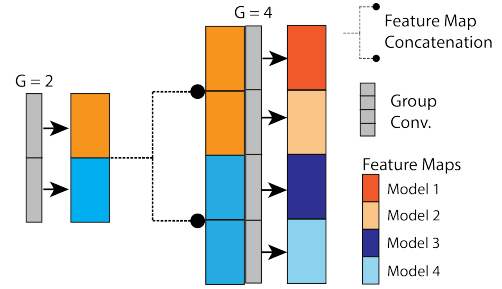

Figure 2: Efficient HNE implementation using group convolutions. Feature maps generated by different branches in the tree are stored in contiguous memory. Using group convolutions, the branch outputs can be computed in parallel. When a branch is split, the feature maps are replicated along the channel dimension and the number of groups for the next convolution is doubled.

Computational Complexity

Hierarchical vs. independent networks. We analyse the inference complexity of a HNE compared to an ensemble composed of independent networks. By assuming that functions $f^b(\cdot)$ require the same number of operations, C, for all b, the complexity of evaluating all the networks in a HNE is $T_{\text{HNE}} = (2^{B+1} - 1)C$, where $B + 1$ is the total number of blocks in each model, from root to leaf. This quantity is proportional to $2^{B+1} - 1$, which is the total number of nodes in a binary-tree of depth B. On the other hand, an ensemble composed by N networks with independent parameters has an inference cost of $T_{\text{Ind}} = (B + 1)NC$.

Considering the same number of models in the ensemble for both approaches, $N = 2^B$, the ratio between the previous time complexities is defined by:

$$R = \frac{T_{\text{Ind}}}{T_{\text{HNE}}} = \frac{B + 1}{2 - 2^{-B}}. \tag{2}$$

For $B = 0$, both independent and hierarchical ensembles reduce to a single model, and have the same computational complexity ($R = 1$). When the number of models is increased ($B > 0$), the second term in the denominator becomes negligible, and the speed-up of HNE *w.r.t* to an independent ensemble increases linearly in B, with $R \approx (B + 1)/2$. This linear speed-up is important since this is what makes larger ensembles, which enjoy improved accuracy, computationally more affordable.

Efficient HNE implementation. Despite the theoretical reduction of the inference complexity, a naive implementation where the individual network outputs are computed sequentially does not allow to fully exploit the parallelization provided by GPUs. Fortunately, the evaluation of the different networks in the HNE can be parallelized by means of group convolutions (Howard et al. 2017; Xie et al. 2017), where different sets of input channels are used to compute an independent set of outputs, see Figure 2. Compared to sequential model evaluation, this strategy allows to drastically reduce training time given that all the models can be computed with a single forward-pass.

HNE Optimization

Given a training set $D = \{(\mathbf{x}, \mathbf{y})_{1:M}\}$ composed of M sample and label pairs, HNE parameters are optimized by minimizing a loss function for each individual network as

$$\mathcal{L}^{\mathrm{I}}(\Theta) = \sum_{n=1}^{N} \sum_{m=1}^{M} \ell(F(\mathbf{x}_m; \theta_n), \mathbf{y}_m), \qquad (3)$$

where $\ell(\cdot, \cdot)$ is the cross-entropy loss comparing ground-truth labels with network outputs.

A drawback of the loss in Eq. (3) is that it is symmetric among the different models in the ensemble. Notably, it ignores the hierarchical structure of the sub-trees that are used to compose smaller sub-ensembles for adaptive inference complexity. To address this limitation, we can optimize a loss that measures the accuracy of the different sub-trees corresponding to the evaluation of an increasing number of networks in the ensemble:

$$\mathcal{L}^{\mathrm{S}}(\Theta) = \sum_{b=0}^{B} \sum_{m=1}^{M} \ell(\mathbf{y}_m^b, \mathbf{y}_m), \qquad (4)$$

where $\mathbf{y}_m^b$ is defined in Eq. (1). Despite the apparent advantages of replacing Eq. (3) by Eq. (4) during learning, we empirically show that this strategy generally produces worse results. The reason is that Eq. (4) impedes the branches to behave as an ensemble of independent networks. Instead, the different models tend to co-adapt in order to minimize the training error. As a consequence, averaging their outputs does not reduce the variance over test data predictions. To effectively exploit the hierarchical structure of HNE outputs during learning, we propose an alternative approach below.

Hierarchical Distillation

Previous work on network ensembles have explored the use of co-distillation (Anil et al. 2018; Song and Chai 2018). These methods attempt to transfer the ensemble knowledge to the individual models by introducing an auxiliary distillation loss for each network:

$$\mathcal{L}^{\mathrm{D}}(\Theta) = \sum_{n=1}^{N} \sum_{m=1}^{M} \ell(F_n(\mathbf{x}_m; \theta_n), \mathbf{y}_m^E), \qquad (5)$$

where $\mathbf{y}^E = \frac{1}{N} \sum_{n=1}^{N} F_n(\mathbf{x}_m; \theta_n)$ is the ensemble output for sample $\mathbf{x}_m$. The cross-entropy loss $\ell(\cdot, \cdot)$ compares the network outputs with the soft-labels generated by using a soft-max function over $\mathbf{y}_m^E$. During training, the distillation loss is combined with the cross-entropy loss of Eq. (3) as $(1 - \alpha)\mathcal{L}^{\mathrm{I}} + \alpha\mathcal{L}^{\mathrm{D}}$, where α is an hyper-parameter controlling the trade-off between both terms. The gradients of $\mathbf{y}_m^E$ w.r.t Θ parameters are not back-propagated during optimization.

Whereas this distillation approach boosts the performance of individual models, it has a critical drawback in the context of ensemble learning for anytime inference. In particular, co-distillation encourage all the predictions to be similar to their average. As a consequence, the variance between model predictions decreases, limiting the improvement given by combining multiple models in an ensemble. To address this limitation, we propose a novel distillation scheme which we

refer to it as "hierarchical distillation". The core idea is to transfer the knowledge from the full ensemble to the smaller sub-ensembles used for anytime inference in HNE. In particular, we minimize:

$$\mathcal{L}^{\mathrm{HD}}(\Theta) = \sum_{b=0}^{B-1} \sum_{m=1}^{M} \ell(\mathbf{y}_m^b, \mathbf{y}_m^E). \qquad (6)$$

Different from $\mathcal{L}^{\mathrm{S}}(\cdot)$, our hierarchical distillation loss distills the predictions for each sub-tree towards the full ensemble outputs. Additionally, $\mathcal{L}^{\mathrm{HD}}(\cdot)$ does not force all the independent outputs to be similar to the full ensemble prediction as in standard distillation. Given that the first evaluated model in the tree is fixed, the different subensembles are always composed by the same subset of networks. In contrast, our hierarchical distillation loss encourages the ensemble prediction obtained from a subset of models to match the full ensemble prediction. Therefore, the outputs between the individual models in this subset can be diverse and still minimize the distillation loss, preserving the model diversity and retaining the advantages of averaging multiple networks. As empirically shown in our experiments, the proposed distillation loss slightly reduces the model diversity compared to the case where no distillation is used. However, the accuracy of individual models tends to be much higher and thus, the ensembles performance is significantly improved.

Experiments

Datasets. We experiment with the CIFAR-10/100 (Krizhevsky 2009) and ImageNet (Russakovsky et al. 2015) datasets. CIFAR-10/100 contain 50k train and 10k test images from 10 and 100 classes, respectively. Following standard protocols (He et al. 2016), we pre-process the images by normalizing their mean and standard-deviation for each color channel. Additionally, during training we use a data augmentation process where we extract random crops of 32×32 after applying a 4-pixel zero padding to the original image or its horizontal flip. Imagenet is composed by 1.2M and 50k high-resolution images for training and validation, respectively, labelled across 1,000 different categories. We use the standard protocol during evaluation resizing the image and extracting a center crop of 224×224 (He et al. 2016). For training, we apply the same data augmentation process as in (Yang et al. 2020; Huang et al. 2018a). We report classification accuracy.

Base architectures. We implement HNE with commonly used architectures. For CIFAR-10/100, we use a variant of ResNet (He et al. 2016), composed of a sequence of residual convolutional layers with bottlenecks. We employ depth-wise instead of regular convolutions to reduce computational complexity. We generate a HNE with a total five blocks embedding an ensemble of $N = 16$ CNNs. We report results for the base ResNet architecture, as well as a version where for all layers we divided the number of feature channels by two (HNE$_{small}$). This provides a more complete evaluation by adding a regime where inference is extremely efficient.

For ImageNet, we implement an HNE based on MobileNet v2 (Sandler et al. 2018), which uses inverted residual layers and depth-wise convolutions as main building blocks.
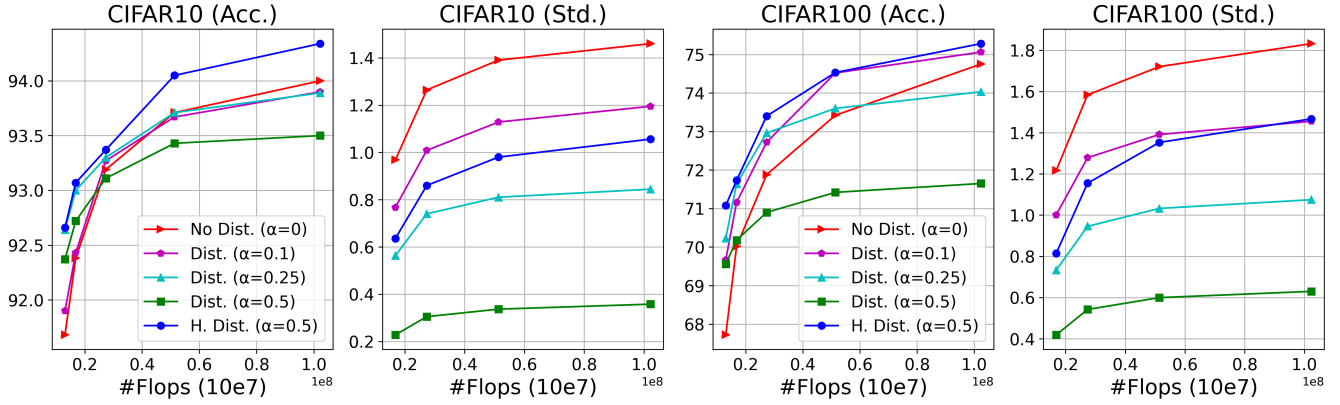
Figure 3: Accuracy and standard deviation in logits *vs* FLOPs for HNE$_{small}$ trained (i) without distillation, (ii) with distillation, and (iii) with our hierarchical distillation. Curves represent results for ensembles of size 1 up to 16.
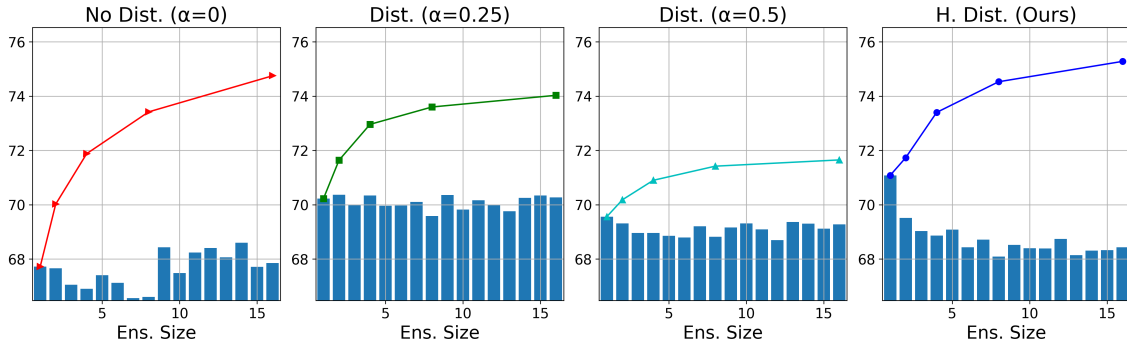

Figure 4: Results on CIFAR-100 for HNE$_{small}$ trained without distillation, standard distillation and the proposed hierarchical distillation. Curves indicate the performance of ensembles of different sizes. Bars depict the accuracy of individual models.

| | | | \# models evaluated for inference | | | | |
			1	2	4	8	16
HNE$_{small}$ (N=16)	**C10**	$\mathcal{L}^I$	91.7	92.4	93.2	93.7	94.0
		$\mathcal{L}^S$	91.6	92.1	92.7	92.9	93.3
		$\mathcal{L}^{HD}$	**92.7**	**93.1**	**93.4**	**94.1**	**94.3**
	C100	$\mathcal{L}^I$	67.7	70.0	71.9	73.4	74.8
		$\mathcal{L}^S$	65.0	66.3	68.3	69.6	72.6
		$\mathcal{L}^{HD}$	**71.1**	**71.7**	**73.4**	**74.5**	**75.3**
HNE (N=16)	**C10**	$\mathcal{L}^I$	93.6	94.2	94.8	95.0	95.2
		$\mathcal{L}^S$	92.9	93.6	93.5	94.1	94.4
		$\mathcal{L}^{HD}$	**94.6**	**94.9**	**95.1**	**95.5**	**95.6**
	C100	$\mathcal{L}^I$	73.5	75.4	77.5	**79.0**	79.7
		$\mathcal{L}^S$	70.7	73.3	74.0	75.7	76.8
		$\mathcal{L}^{HD}$	**76.1**	**77.2**	**78.0**	**79.0**	**79.8**

Table 1: Accuracy of HNE and HNE$_{small}$ embedding 16 different networks on CIFAR. Columns correspond to the number of models evaluated during inference.

In this case, we also use five blocks generating $N = 16$ different networks. In the supplementary material we present a detailed description of our HNE implementation using ResNet and MobileNetv2 and provide all the training hyperparameters. The particular design choices for both architectures are set to produce a similar computational complexity

as previous methods to which we compare. We have released a Pytorch implementation of HNE [1].

Inference complexity metric. Following (Huang et al. 2018a; Ruiz and Verbeek 2019; Zhang, Ren, and Urtasun 2019), we evaluate the computational complexity of the models according to the number of floating-point operations (FLOPs) during inference. The advantages of this metric is that it is independent from differences in hardware and implementations, and is strongly correlated with the wall-clock inference time.

Ablation Study on CIFAR-10/100

Optimizing HNE. In order to understand the advantages of our hierarchical distillation approach, we compare three different alternative objectives to train HNE: (i) the independent loss across models, $\mathcal{L}^I$ in Eq. (3), (ii) the structured loss maximizing the accuracy of nested ensembles, $\mathcal{L}^S$ in Eq. (4), and (iii) our hierarchical distillation loss, $\mathcal{L}^{HD}$ in Eq. (6), which is combined with $\mathcal{L}_I$.

As shown in Table 1, $\mathcal{L}^I$ provides better performance as compared to training with $\mathcal{L}^S$. The reason is that $\mathcal{L}^S$ encourages individual model outputs to co-adapt in order to

---

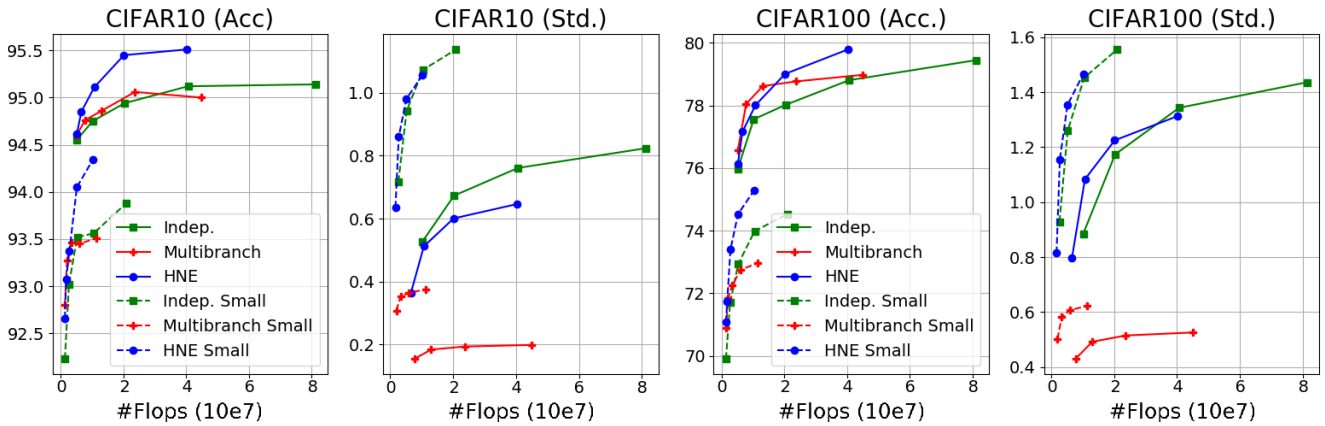[1] https://gitlab.com/adriaruizo/dhne-aaai21

Figure 5: Results on CIFAR-100 for HNE$_{small}$ using different ensemble architectures: (i) fully-independent networks, (ii) multi-branch architecture with shared backbone, (iii) our proposed HNEs.

minimize the training error. However, as the different networks are not trained independently, the variance reduction resulting from averaging multiple models in an ensemble is lost, causing a performance drop on test data. Using our hierarchical distillation loss, however, consistently outperforms the alternatives in all the evaluated ensemble sizes, both architectures, and on both datasets. This is because our approach preserves the advantages of averaging multiple independent models, at the same time that the performance of hierarchical ensembles is increased via distillation.

Comparing distillation approaches. After demonstrating the effectiveness of our distillation method to boost the performance of hierarchical ensembles, we evaluate its advantages *w.r.t* standard distillation. For this purpose, we train HNE using the loss $\mathcal{L}^D$ of Eq. (5), as (Song and Chai 2018). For standard distillation we evaluate a range of values for α to mix the distillation and cross-entropy loss in order to analyze the impact on accuracy and model diversity. The latter is measured as the std. deviation in the logits of the evaluated models, averaged across all classes and test samples. In Figure 3 we report both the accuracy and logit standard deviation on the test set. For limited space reasons, we only show results for HNE$_{small}$. The corresponding figure for the bigger HNE model can be found in supplementary material.

Consider standard distillation with a high weight on the distillation loss ($\alpha = 0.5$). As expected, the performance of small ensembles is improved *w.r.t* training without distillation. For larger ensembles, however, the accuracy tends to be significantly lower compared to not using distillation. This is due to the reduction in diversity among the models induced by the standard distillation loss. This effect can be controlled by reducing the weight α, but smaller values (less distillation) compromise the accuracy of small ensembles. In contrast, our hierarchical distillation achieves the best FLOPs-accuracy trade-offs for all ensemble sizes and datasets. For small ensemble sizes, our hierarchical distillation obtains similar or better accuracy than standard distillation. For large ensemble sizes, our approach significantly improves over standard distillation, and the accuracy is com-

parable or better than those obtained without distillation. These results clearly shows the advantages of hierarchical distillation for any-time inference. The reason is that, in this setting, the goal is not only in to optimize the accuracy for a given FLOP count, but to jointly boost the performance for all the possible ensemble sizes.

Analysis of individual network accuracies. To provide additional insight in the previous results, Figure 4 depicts the performance of HNE for different ensemble sizes used during inference (curves), and the accuracy of the individual networks in the ensemble (bars). Comparing the results without distillation (first col.) to standard distillation (second and third cols.), we make two observations. First, standard distillation significantly increases the accuracy of the individual models. This is expected because the knowledge from the complete ensemble is transferred to each network independently. Second, when using standard distillation, performance tends to be lower than HNE trained without distillation when the number of models in the ensemble is increased. Both phenomena are explained by the tendency of standard distillation to decrease the diversity between the individual models. As a consequence, the gains obtained by combining a large number of networks is reduced even though they are individually more accurate.

The results for our Hierarchical Distillation (last col.) clearly show its advantages with respect to the alternative approaches. We can observe that the accuracy of the first model is better than in HNE trained without distillation, and also significantly higher than the accuracy of the model obtained by standard distillation. The reason is that the ensemble knowledge is directly transferred to the predictions of the first network in the hierarchical structure. The performance of the other individual networks in the ensemble tends to be lower than when training with standard distillation. The improvement obtained by ensembling their predictions is, however, significantly larger. This is because hierarchical distillation preserves the diversity between networks, compensating the lower accuracy of the individual models.

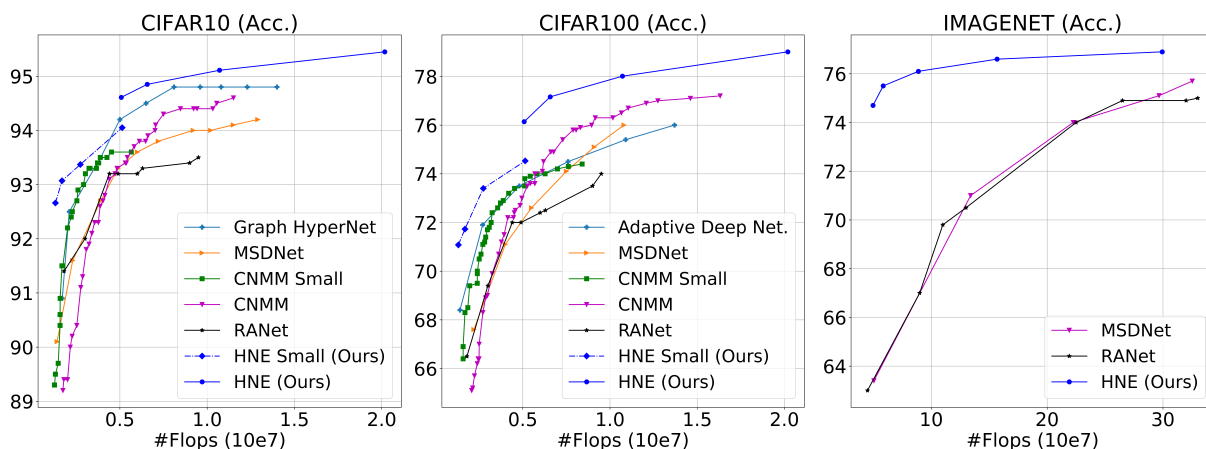**Hierarchical parameter sharing.** In this experiment, we

Figure 6: Comparison of HNE with the state of the art. Each curve corresponds to a single model with anytime inference.

compare the performance obtained by HNE and an ensemble of independent networks, using the same base architecture. We also compare to multi-branch architectures (Lan, Zhu, and Gong 2018; Lee et al. 2015). In the latter case, the complexity is reduced by sharing the same "backbone" for the first blocks for all models, and then bifurcate in one step to N independent branches for the subsequent blocks. For HNE we use five different blocks with $N = 16$. To achieve architectures in a similar FLOP range, we use the first three blocks as the backbone and implement 16 independent branches for the last two blocks. We use our hierarchical distillation loss for all three architectures.

In Figure 5 we report both the accuracy (Acc) and the standard deviation of logits (Std). We observe that HNE obtains significantly better results than the independent networks. Moreover, the hierarchical structure allows to significantly reduce the computational cost for large ensemble sizes. For small ensembles on CIFAR-100 the multi-branch models obtain slightly better accuracy than HNE. In all other settings, HNE achieve similar or better accuracy, especially for larger ensembles. The results can again be understood by observing the diversity across models. HNE and independent models have similar diversity, while in the case of the multi-branch ensemble diversity is significantly lower. This shows the importance of using different parameters in early-blocks to achieve diversity across models.

Comparison with the State of the Art

CIFAR-10/100. We compare the performance of HNE trained with hierarchical distillation with state-of-the-art approaches for anytime inference: Multi-scale DenseNets (Huang et al. 2018a), Resolution Adaptive Networks (Yang et al. 2020), Graph Hyper-Networks (Zhang, Ren, and Urtasun 2019), Deep Adaptive Networks (Li et al. 2019), and Convolutional Neural Mixture Models (Ruiz and Verbeek 2019). We report HNE results for ensembles of sizes up to eight, in order to provide a maximum FLOP count similar to the compared methods (<250M). Results in Figure 6 show that for both datasets HNE significantly outperforms previous approaches across all the FLOP range.

ImageNet. We compare our method with Multi-Scale Densenets (Huang et al. 2018a), and Resolution Adaptive Networks (Yang et al. 2020). To the best of our knowledge these works have reported state-of-the-art performance on ImageNet for anytime inference. The results in Figure 6 show that our HNE achieves better accuracy than the compared methods across all inference complexities. Compared to the best baseline, our method achieves an accuracy improvement across the different FLOP ranges between 1.5% and 11%. This is a significant performance boost given the difficulty and large-scale nature of ImageNet. Additionally, note that the minimum FLOP count for HNE and the compared models are similar. Whereas HNE needs a full pass over a single base model to provide an initial output, the compared approaches based on intermediate classifiers also require to compute all the intermediate network activations up to the first classifier, which can be considered as the base model for these approaches.

Conclusions

In this paper we proposed Hierarchical Neural Ensembles (HNE), a framework to design deep models with anytime inference. In addition, we introduced a novel hierarchical distillation approach adapted to the structure of HNE. Compared to previous deep models with anytime inference, we have reported state-of-the-art compute-accuracy trade-offs on CIFAR-10/100 and ImageNet. While we have demonstrated the effectiveness of our framework in the context of CNNs for image classification, our approach is generic and can be used to build ensembles of other types of deep networks for different tasks and domains. In particular, HNE can be applied to any base model and network branching. This property allows to design anytime models adapted to different computational constraints such as the maximum and minimum FLOP count or the number of desired operating points. This flexibility allows our framework to be combined with other approaches for efficient inference such as network compression or neural architecture search.

Acknowledgments

References

Anil, R.; Pereyra, G.; Passos, A.; Ormandi, R.; Dahl, G. E.; and Hinton, G. E. 2018. Large scale distributed neural network training through online distillation. *ICLR* .

Ba, J.; and Caruana, R. 2014. Do deep nets really need to be deep? In *NIPS*.

Bhardwaj, K.; Lin, C.-Y.; Sartor, A.; and Marculescu, R. 2019. Memory-and communication-aware model compression for distributed deep learning inference on iot. *ACM Transactions on Embedded Computing Systems (TECS)* 18(5s): 1–22.

Bolukbasi, T.; Wang, J.; Dekel, O.; and Saligrama, V. 2017. Adaptive neural networks for efficient inference. In *ICML*.

Breiman, L. 1996. Bagging predictors. *Machine learning* .

Cai, H.; Gan, C.; Wang, T.; Zhang, Z.; and Han, S. 2020. Once for All: Train One Network and Specialize it for Efficient Deployment. In *ICLR*.

Chen, H.; and Yao, X. 2009. Regularized negative correlation learning for neural network ensembles. *IEEE Transactions on Neural Networks* .

Elbayad, M.; Gu, J.; Grave, E.; and Auli, M. 2020. Depth-Adaptive Transformer. In *ICLR*.

Furlanello, T.; Lipton, Z. C.; Tschannen, M.; Itti, L.; and Anandkumar, A. 2018. Born again neural networks. In *ICML*.

Geiger, M.; Jacot, A.; Spigler, S.; Gabriel, F.; Sagun, L.; d'Ascoli, S.; Biroli, G.; Hongler, C.; and Wyart, M. 2020. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment* 2020(2).

Hansen, L.; and Salamon, P. 1990. Neural Network Ensembles. *PAMI* 12(10).

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* .

Huang, G.; Chen, D.; Li, T.; Wu, F.; van der Maaten, L.; and Weinberger, K. Q. 2018a. Multi-scale dense networks for resource efficient image classification. *ICLR* .

Huang, G.; Li, Y.; Pleiss, G.; Liu, Z.; Hopcroft, J.; and Weinberger, K. 2017. Snapshot ensembles: Train 1, get m for free. *ICLR* .

Huang, G.; Liu, S.; van der Maaten, L.; and Weinberger, K. 2018b. CondenseNet: An efficient densenet using learned group convolutions. In *CVPR*.

Ilg, E.; Cicek, O.; Galesso, S.; Klein, A.; Makansi, O.; Hutter, F.; and Brox, T. 2018. Uncertainty estimates and multi-hypotheses networks for optical flow. In *ECCV*.

Kim, J.; Park, Y.; Kim, G.; and Hwang, S. J. 2017. SplitNet: Learning to semantically split deep networks for parameter reduction and model parallelization. In *ICML*.

Krizhevsky, A. 2009. *Learning Multiple Layers of Features from Tiny Images*. Master's thesis, University of Toronto.

Krogh, A.; and Vedelsby, J. 1995. Neural network ensembles, cross validation, and active learning. In *NIPS*.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*.

Lan, X.; Zhu, X.; and Gong, S. 2018. Knowledge distillation by on-the-fly native ensemble. In *NIPS*.

Lee, J.; and Chung, S.-Y. 2020. Robust Training with Ensemble Consensus. In *ICLR*.

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

Li, H.; Zhang, H.; Qi, X.; Yang, R.; and Huang, G. 2019. Improved Techniques for Training Adaptive Deep Networks. In *ICCV*.

Liu, Y.; Stehouwer, J.; Jourabloo, A.; and Liu, X. 2019. Deep tree learning for zero-shot face anti-spoofing. In *CVPR*.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning efficient convolutional networks through network slimming. In *ICCV*.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*.

Ma, N.; Zhang, X.; Zheng, H.-T.; and Sun, J. 2018. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In *ECCV*.

Malinin, A.; Mlodozeniec, B.; and Gales, M. 2020. Ensemble Distribution Distillation. In *ICLR*.

Minetto, R.; Segundo, M.; and Sarkar, S. 2019. Hydra: an ensemble of convolutional neural networks for geospatial land classification. *IEEE Transactions on Geoscience and Remote Sensing* .

Naftaly, U.; Intrator, N.; and Horn, D. 1997. Optimal ensemble averaging of neural networks. *Network: Computation in Neural Systems* .

Neal, B.; Mittal, S.; Baratin, A.; Tantia, V.; Scicluna, M.; Lacoste-Julien, S.; and Mitliagkas, I. 2018. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591* .

Rokach, L. 2010. Ensemble-based classifiers. *Artificial Intelligence Review* 33(1-2): 1–39.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2015. Fitnets: Hints for thin deep nets. *ICLR* .

Roy, D.; Panda, P.; and Roy, K. 2020. Tree-CNN: a hierarchical deep convolutional neural network for incremental learning. *Neural Networks* .

Ruiz, A.; and Verbeek, J. 2019. Adaptative Inference Cost With Convolutional Neural Mixture Models. In *ICCV*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV* .

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*.

Song, G.; and Chai, W. 2018. Collaborative learning for deep neural networks. In *NIPS*.

Sun, S.; Cheng, Y.; Gan, Z.; and Liu, J. 2019. Patient Knowledge Distillation for BERT Model Compression. In *EMNLP*.

Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; and Le, Q. 2019. MnasNet: Platform-aware neural architecture search for mobile. In *CVPR*.

Tanno, R.; Arulkumaran, K.; Alexander, D.; Criminisi, A.; and Nori, A. 2019. Adaptive neural trees. *ICML* .

Ueda, N.; and Nakano, R. 1996. Generalization error of ensemble estimators. In *Proceedings of International Conference on Neural Networks*. IEEE.

Veit, A.; and Belongie, S. 2018. Convolutional networks with adaptive inference graphs. In *ECCV*.

Wang, X.; Yu, F.; Dou, Z.-Y.; Darrell, T.; and Gonzalez, J. 2018. SkipNet: Learning dynamic routing in convolutional networks. In *ECCV*.

Wu, B.; Dai, X.; Zhang, P.; Wang, Y.; Sun, F.; Wu, Y.; Tian, Y.; Vajda, P.; Jia, Y.; and Keutzer, K. 2019. FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In *CVPR*.

Wu, Z.; Nagarajan, T.; Kumar, A.; Rennie, S.; Davis, L.; Grauman, K.; and Feris, R. 2018. BlockDrop: Dynamic inference paths in residual networks. In *CVPR*.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *CVPR*.

Yang, L.; Han, Y.; Chen, X.; Song, S.; Dai, J.; and Huang, G. 2020. Resolution Adaptive Networks for Efficient Inference. In *CVPR*.

Yu, J.; and Huang, T. 2019. Universally slimmable networks and improved training techniques. *ICCV* .

Yu, J.; Jin, P.; Liu, H.; Bender, G.; Kindermans, P.-J.; Tan, M.; Huang, T.; Song, X.; Pang, R.; and Le, Q. 2020. Bignas: Scaling up neural architecture search with big single-stage models. *ECCV* .

Yu, J.; Yang, L.; Xu, N.; Yang, J.; and Huang, T. 2018. Slimmable neural networks. *ICLR* .

Zhang, C.; Ren, M.; and Urtasun, R. 2019. Graph hypernetworks for neural architecture search. *ICLR* .

Zhang, Y.; Xiang, T.; Hospedales, T.; and Lu, H. 2018. Deep mutual learning. In *CVPR*.

Zhou, Z.-H.; Wu, J.; and Tang, W. 2002. Ensembling neural networks: many could be better than all. *Artificial intelligence* .