`HydraViT`: Stacking Heads for a Scalable ViT

Janek Haberer[*], **Ali Hojjat**[*], **Olaf Landsiedel**
Kiel University, Germany
[*]*Equal contribution*
{janek.haberer,ali.hojjat,olaf.landsiedel}@cs.uni-kiel.de

Abstract

The architecture of Vision Transformers (ViTs), particularly the Multi-head Attention (MHA) mechanism, imposes substantial hardware demands. Deploying ViTs on devices with varying constraints, such as mobile phones, requires multiple models of different sizes. However, this approach has limitations, such as training and storing each required model separately. This paper introduces `HydraViT`, a novel approach that addresses these limitations by stacking attention heads to achieve a scalable ViT. By repeatedly changing the size of the embedded dimensions throughout each layer and their corresponding number of attention heads in MHA during training, `HydraViT` induces multiple subnetworks. Thereby, `HydraViT` achieves adaptability across a wide spectrum of hardware environments while maintaining performance. Our experimental results demonstrate the efficacy of `HydraViT` in achieving a scalable ViT with up to 10 subnetworks, covering a wide range of resource constraints. `HydraViT` achieves up to 5 p.p. more accuracy with the same GMACs and up to 7 p.p. more accuracy with the same throughput on ImageNet-1K compared to the baselines, making it an effective solution for scenarios where hardware availability is diverse or varies over time. The source code is available at https://github.com/ds-kiel/HydraViT.

(a) GMACs vs. Accuracy   (b) Throughput vs. Accuracy
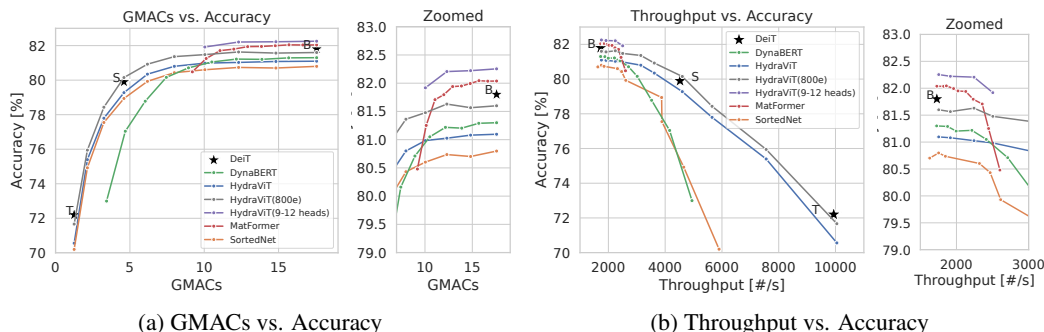
Figure 1: Performance comparison of `HydraViT` and baselines on ImageNet-1K in terms of GMACs (a) and throughput (b) evaluated on NVIDIA A100 80GB PCIe. `HydraViT` trained on 3-12 heads demonstrates superior performance over DynaBERT (Hou et al., 2020) and SortedNet (Valipour et al., 2023). While MatFormer (Kudugunta et al., 2023) shows higher performance than `HydraViT` within its limited scalability range, but when we train on a narrower scalability range (9-12 heads), `HydraViT` surpasses MatFormer. We also show that training `HydraViT` for more epochs can further improve accuracy. Note that each line corresponds to one model, and changing the number of heads in the vanilla DeiT models significantly drops their accuracy to less than 30%.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

1 Introduction

Motivation Following the breakthrough of Transformers (Vaswani et al., 2017), Dosovitskiy et al. (2021) established the Vision Transformer (ViT) as the base transformer architecture for computer vision tasks. As such, numerous studies build on top of ViTs as their base (Liu et al., 2021; Tolstikhin et al., 2021; Yu et al., 2022). In this architecture, Multi-head Attention (MHA) plays an important part, capturing global relations between different parts of the input image. However, ViTs have a much higher hardware demand due to the size of the attention matrices in MHA, which makes it challenging to find a configuration that fits heterogeneous devices.

Table 1: ViT Configurations

	ViT-Ti	ViT-S	ViT-B
# Layers	12	12	12
Dim	192	384	768
# Heads	3	6	12
Dim per Head	64	64	64
# Params	5.7 M	22 M	86 M

To accommodate devices with various constraints, ViTs offer multiple independently trained models with different sizes and hardware requirements, such as the number of parameters, FLOPS, MACs, and hardware settings such as latency and RAM, with sizes typically increasing nearly at a logarithmic scale (Kudugunta et al., 2023), see Table 1. Overall, in the configurations of ViTs, the number of heads and their corresponding embedded dimension in MHA emerges as the key hyperparameter that distinguishes them.

While being a reasonable solution for hardware adaptability, this approach has two primary disadvantages: (1) Despite larger models, e.g., ViT-S and ViT-B, not having a significant accuracy difference, each of these models needs to be individually trained, tuned, and stored, which is not suitable for downstream scenarios where the hardware availability changes over time. (2) Although the configuration range covers different hardware requirements, the granularity is usually limited to a small selection of models and cannot cover all device constraints.

Observation By investigating the architecture of these configurations, we notice that ViT-Ti, ViT-S, and ViT-B share the same architecture, except they differ in the size of the embeddings and the corresponding number of attention heads they employ, having 3, 6, and 12 heads, respectively. In essence, this can be expressed as $ViT_T \subseteq ViT_S \subseteq ViT_B$, see Table 1.

Research question In this paper, we address the following question: Can we train a universal ViT model with H attention heads and embedding dimension E, such that by increasing the embedded dimension from e_1 to e_2, where $e_1 < e_2 \leq E$, and its corresponding number of heads from h_1 to h_2, where $h_1 < h_2 \leq H$, the model's accuracy gracefully improves?

Approach In this paper, we propose `HydraViT`, a stochastic training approach that extracts subsets of embeddings and their corresponding heads within MHA across a universal ViT architecture and jointly trains them. Specifically, during training, we utilize a uniform distribution to pick a value k, where $k \leq H$. Subsequently, we extract the embedded dimension ($[0 : k \times HeadDim]$), where $HeadDim$ is the size of each head, and its corresponding first k heads ($[0 : k]$) and only include these in both the backpropagation and forward paths of the training process. To enable the extraction of such subnetworks, we reimplement all components of the ViT including MHA, Multilayer Perceptron (MLP), and Normalization Layer (NORM), see Fig. 2. By using this stochastic approach, the heads will be stacked based on their importance, such that the first heads capture the most significant features and the last heads the least significant ones from the input image.

Figure 2: Architecture of `HydraViT`

After the training phase is completed, during inference, `HydraViT` can dynamically select the number of heads based on the hardware demands. For example, if only $p\%$ of the hardware is available, `HydraViT` extracts a subnetwork with the embedded size of $\lceil p \times H \rceil \times HeadDim$ and the first $\lceil p \times H \rceil$ heads and runs the inference. This flexibility is particularly advantageous in scenarios such as processing a sequence of input images, like a video stream, where latency is critical, especially on constrained devices such as mobile devices. In such environments, where various tasks are running simultaneously, and hardware availability dynamically fluctuates, or we need to meet a

2

Figure 3: In this figure, we illustrate an example of how we extract a subnetwork with 4 heads in MHA with a total number of 6 heads. In `HydraViT`, with the stochastic dropout training, we order the attention heads in MHA and consequently their corresponding embedding vectors based on their importance.

deadline, the ability to adapt the model's configuration without loading a new model offers significant benefits.

Contributions:

1. We introduce `HydraViT`, a stochastic training method that extracts and jointly trains subnetworks inside the standard ViT architecture for scalable inference.
2. In a standard ViT architecture with H attention heads, `HydraViT` can induce H submodels within a universal model.
3. `HydraViT` outperforms its scalable baselines with up to 7 p.p. more accuracy at the same throughput and performance comparable to the respective standard models DeiT-tiny, DeiT-small, and DeiT-base, see Figure 1 for details.

2 Related Work

The original Vision Transformer (ViT) (Dosovitskiy et al., 2021) has become the default architecture of Transformer-based vision models. While many works improve upon the original implementation by changing the architecture or training process (Liu et al., 2022; Touvron et al., 2022; Wang et al., 2021b), none of these works yield a scalable architecture and need multiple separate sets of weights to be able to deploy an efficient model on devices with various constraints.

For Convolutional Neural Networks (CNNs), Fang et al. (2018) create a scalable network by pruning unimportant filters and then repeatedly freezing the entire model, adding new filters, and fine-tuning. Thereby, they achieve a network that can be run with a flexible number of filters. Yu et al. (2018) achieve the same, but instead of freezing, they train a network for different layer widths at once. For Transformers, Chavan et al. (2022) use sparsity to efficiently search for a subnetwork but then require fine-tuning for every extracted subnetwork to acquire good accuracy.

Beyer et al. (2023) introduce a small change in the training process by feeding differently sized patches to the network. Thereby, they can reduce or increase the number of patches, affecting the speed and accuracy during inference. Other works use the importance of each patch to prune the least important patches during inference to achieve a dynamic ViT (Yin et al., 2022; Rao et al., 2021; Tang et al., 2022; Wang et al., 2021a).

Matryoshka Representation Learning (Kusupati et al., 2022) and Ordered Representations with Nested Dropout (Rippel et al., 2014; Hojjat et al., 2023) are techniques to make the embedding dimension of Transformers flexible, i.e., create a Transformer, which can also run partially. Kudugunta et al. (2023) use Matryoshka Learning to make the hidden layer of the MLP in each Transformer block flexible. Hou et al. (2020) change the hidden layer of the MLP and the MHA but still use the original dimension between Transformer blocks and also between MHA and MLP. Salehi et al. (2023) make the entire embedding dimension in a Transformer block flexible. However, they rely on a few non-flexible blocks followed by a router that determines the embedding dimension for the flexible blocks, which adds complexity and hinders the ability to choose with which network width to run.

3

(a) MLP	(b) NORM
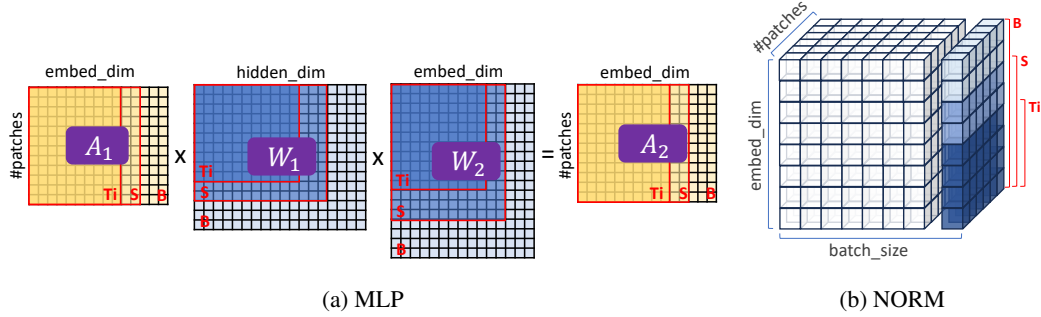
Figure 4: An illustration of subnetwork extraction within MLP and NORM layers, introduced in HydraViT. Fig. 4a demonstrates how HydraViT slices activations, denoted as A_1 and A_2, along with their respective weight matrices, denoted as W_1 and W_2, based on the number of utilized heads. Also, Fig. 4b shows how HydraViT applies normalization on the activation corresponding to the used heads. For simplicity, only subnetworks with 3, 6, and 12 heads, corresponding to ViT-Ti, ViT-S, and ViT-B respectively, are presented.

Valipour et al. (2023) propose SortedNet that trains networks to be flexible in depth and width. However, they mainly focus on evaluating with CNNs on CIFAR10 (Krizhevsky et al., 2009) and Transformers on Natural Language Processing (NLP) tasks in contrast to us. Additionally, they keep the number of heads in MHA fixed at 12, whereas we show that reducing the number of heads coupled to the embedding dimension, i.e., the first 64 values of the embedding always belong to the first head in MHA, the second 64 values belong to the second head, and so on, removes inconsistencies in the scaling of the MHA and improves performance.

Motivated by these previous works, in HydraViT, we propose a flexible ViT in which we, unlike previous works, adjust every single layer, and except for one initial training run, there is no further fine-tuning required. Additionally, we show that reducing the number of heads coupled to the embedding dimension, a weighted subnetwork sampling distribution, and adding separate classifier heads improves the performance of subnetworks.

3 HydraViT

In this section, we introduce HydraViT, which builds upon the ViT architecture. We start by detailing how general ViTs function. Next, we explain how HydraViT can extract subnetworks within the MHA, NORM, and MLP layers. Finally, we describe the stochastic training regime used in HydraViT to simultaneously train a universal ViT architecture and all of its subnetworks.

Vision Transformer HydraViT is based on the ViT architecture (Dosovitskiy et al., 2021). We start by taking the input image x and breaking it down into P patches. Each patch is then embedded into a vector of size E using patch embedding, denoted as $\mathcal{E}^E$. Positional encoding is subsequently applied to the embeddings. Following these preprocessing steps, it passes the embeddings through L blocks consisting of MHA with H heads denoted as $\mathcal{A}^H$, NORM layer denoted as $\mathcal{N}^P$, MLP denoted as $\mathcal{M}^{E \times M \times E}$ to predict the class of the input image x, where M is the dimension of the hidden layer of the MLP. With the model parameters θ, we can formulate this architecture as follows:

$$V_\theta(x; \mathcal{E}^E; \mathcal{A}^H; \mathcal{M}^{E \times M \times E}; \mathcal{N}^P) \tag{1}$$

HydraViT HydraViT is able to induce any subnetwork with $k \leq H$ heads within the standard architecture of ViT. To do so, HydraViT extracts the first k heads denoted as $\mathcal{A}^{[0:k]}$, and the embeddings corresponding to these heads in MHA and NORM layers. Additionally, it extracts the initial $[\frac{E}{H} \times k]$ neurons from the first and last layers of the MLP, and the first $[\frac{M}{H} \times k]$ neurons from the hidden layer of MLP. Therefore, we can formulate the subnetwork extracted from Eq. 1 as follows:

$$V_{\theta_k}(x; \mathcal{E}^{[0:(\frac{E}{H} \times k)]}; \mathcal{A}^{[0:k]}; \mathcal{M}^{[0:(\frac{E}{H} \times k)] \times [0:(\frac{M}{H} \times k)] \times [0:(\frac{E}{H} \times k)]}; \mathcal{N}^{[0:(\frac{E}{H} \times k)]}); k \in \{1, 2, \ldots, H\} \tag{2}$$

Figure 4 illustrates how HydraViT extracts subnetworks within NORM and MLP layers. For simplicity, we demonstrate subnetworks with 3, 6, and 12 heads, representing configurations for

4

Algorithm 1: Stochastic dropout training
___
Data: HydraViT: V_{θ_k},
Number of batches: N_{batch},
Number of the heads of the universal model: H,
Uniform distribution: $\mathcal{U}$.
for $1 \le e_i \le N_{epoch}$ **do**
    for $1 \le b_i \le N_{batch}$ **do**
        * sample a subnetwork *\
        $V_\theta \xrightarrow{k \sim \mathcal{U}(k)} V_{\theta_k}, k \in \{1, 2, \ldots H\}$;
        * calculate single-objective loss *\
        $\mathcal{L}(V_{\theta_k}(x_{b_i}), y)$;
        Back-propagation through subnetwork V_{θ_k};
    end
end
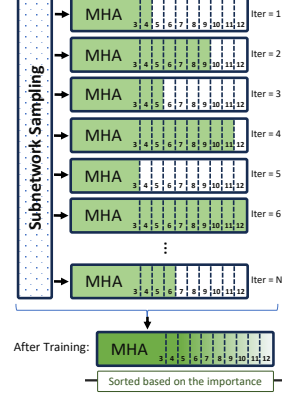
Figure 5: Stochastic tail-drop training.

ViT-Ti, ViT-S, and ViT-B, respectively. Additionally, in Figure 3, we present an example of how `HydraViT` extracts a subset of heads and their corresponding embeddings in MHA layers. By designing `HydraViT` this way, we can deploy only a subnetwork, e.g., `HydraViT` with 6 heads, and still have the option at runtime to run with even fewer heads. It is not necessary to deploy `HydraViT` with all weights, which is necessary for deployment on more constrained devices.

Stochastic dropout training Ideally, to achieve a truly scalable model, we need to extract all the possible subnetworks, calculate their loss, sum them up, and minimize it. This yields the following multi-objective optimization problem:

$$\min_{[\theta_1 \ldots \theta_H]} \sum_{i=1}^{N} \sum_{h=1}^{H} \mathcal{L}(V_{\theta_h}(x_i), y_i) \tag{3}$$

where N is the number of samples, x_i is the input and y_i is the ground truth. However, optimizing this multi-objective loss function has a complexity of $\mathcal{O}(N \times H)$ and requires at least H times more RAM compared to a single-objective loss function to store the gradient graphs, a demand that exceeds the capacity of a current GPU. To address this issue, we suggest employing stochastic training: On each iteration, instead of extracting all of the H possible subnetworks and optimizing a multi-objective loss function, we sample a value $k \in \{1, 2, \ldots, H\}$ based on a uniform discrete probability distribution $\mathcal{U}(k)$. Then we extract its respective subnetwork V_{θ_k}, and minimize only this loss function, see Alg. 1. This approach decreases the complexity of Eq. 3 to $\mathcal{O}(N)$. In this training regime, the first parts of embeddings and their corresponding attention heads become more involved in the training process, while the later parts are less engaged. After training, due to this asymmetric training, which can also be seen as an order-aware biased version of dropout, the embedding values and their respective attention heads are ordered based on importance, see Fig. 5. Note that despite the similarity to dropout we do not need scaling during training as our training and inference phases are identical. Thereby, we can simplify the Eq. 3 as follows:

$$k \sim \mathcal{U}(k); \quad \min_{\theta_k} \sum_{i=1}^{N} \mathcal{L}(V_{\theta_k}(x_i), y_i) \tag{4}$$

Separate classifiers We implement a mechanism to train separate classifier heads for each subnetwork. This adds a few parameters to the model, but only during training or when running the model in a dynamic mode, i.e., having the ability to freely choose for each input with how many heads to run the model. The advantage is that we do not need to find a shared classifier that can deal with the different amounts of features each subnetwork provides. However, if we fix the number of epochs, each classifier gets fewer gradient updates than the shared one, which is why we only use this when training `HydraViT` with 3 subnetworks.

5

Table 2: The accuracy of `HydraViT` with our different design choices. "3 Heads" corresponds to a subnetwork that has the same architecture as DeiT-tiny, "6 Heads" corresponds to DeiT-small, and "12 Heads" corresponds to DeiT-base.

Weighted Sampling?	Separate Classifiers?	Epochs	Acc [%] 3 Heads	Acc [%] 6 Heads	Acc [%] 12 Heads
✗	✗	300	72.56	79.35	80.63
✗	✗	400	73.16	79.63	80.90
✗	✗	500	73.54	80.09	81.30
✓	✗	300	72.02	79.35	80.98
✓	✗	400	72.45	79.85	81.49
✓	✗	500	72.50	79.89	81.63
✗	✓	300	72.78	79.44	80.52
✗	✓	400	73.24	79.88	81.13
✗	✓	500	73.42	80.12	81.13
✓	✓	300	72.13	79.45	81.18
✓	✓	400	72.46	79.93	81.58
✓	✓	500	72.65	80.08	81.77
DeiT-tiny/small/base			72.2	79.9	81.8

Subnetwork sampling function   When trying to train a single set of weights containing multiple subnetworks, we expect an accuracy drop compared to if each subnetwork had its own set of weights. While we mention that we use a uniform discrete probability distribution to sample subnetworks, we can also use a weighted distribution function. With weighted subnetwork sampling, we can guide the model to focus on certain submodels more than others. This is useful in a deployment scenario in which we have many devices with similar resources and want to maximize accuracy for them while maintaining good accuracy for other devices with different resources.

4   Evaluation

In this section, we evaluate the performance of `HydraViT` and compare it to the baselines introduced in Sec. 2. We assess all experiments and baselines on ImageNet-1K (Deng et al., 2009) at a resolution of 224×224. We implement on top of timm (Wightman, 2019) and train according to the procedure of Touvron et al. (2021) but without knowledge distillation. We use an NVIDIA A100 80GB PCIe to measure throughput. For RAM, we measure the model and forward pass usage with a batch size of 1. We also calculate GMACs with a batch size of 1, i.e., the GMACs needed to classify a single image.

For the experiments, we used an internal GPU cluster, and each epoch took around 15 minutes. During prototyping, we estimate that we performed an additional 50 runs with 300 epochs.

First, we show that we can attain one set of weights that achieves very similar results as the three separate DeiT models DeiT-tiny, DeiT-small, and DeiT-base (Touvron et al., 2021). Then, we look at how our design choices, i.e., changing the number of heads coupled to the embedding dimension, weighted subnetwork sampling, and adding separate classifiers for each subnetwork, impact the accuracy. Afterward, we compare `HydraViT` to the following three baselines:

- **MatFormer** Kudugunta et al. (2023) focus only on the hidden layer of the MLP to achieve a flexible Transformer and do not change the heads in MHA or the dimension of intermediate embeddings.

- **DynaBERT** Hou et al. (2020) adjust the heads in MHA in addition to the dimension of MLP and, as a result, make both flexible. However, the intermediate embedding dimension is the same as the original one in between each Transformer block and between MHA and MLP, which results in more parameters and MACs.

- **SortedNet** Valipour et al. (2023) change every single embedding, including the ones between MHA and MLP and between Transformer blocks. However, they keep the number of heads in MHA fixed, resulting in less information per head and introducing inconsistencies in the scaling of the heads in MHA.

6
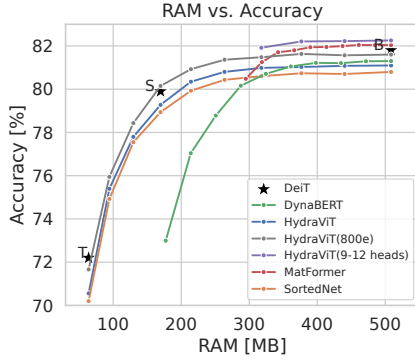
Figure 6: Performance comparison of `HydraViT` and baselines in terms of RAM usage. Similar to Figure 1, `HydraViT` achieves the best accuracy per RAM usage.
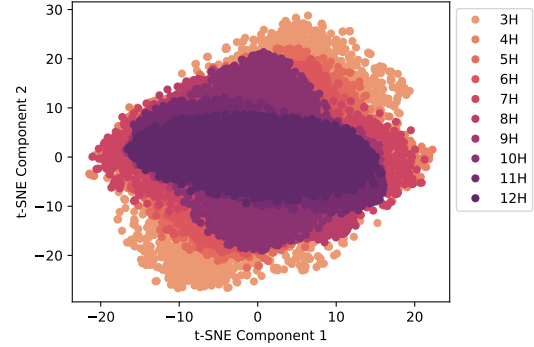
Figure 7: t-SNE representation of the last layer of `HydraViT` with different numbers of heads. As we can see, having more heads leads to more compact representations.

In contrast, instead of keeping the number of heads fixed, we change it coupled to the embedding dimension, such that each head gets the same amount of information as in the original ViT. We also evaluate adding separate classifiers and employing weighted subnetwork sampling during the training. Finally, we perform an attention analysis on our model to showcase the effect of adding heads in MHA.

4.1 One set of weights is as good as three: Tiny, Small, and Base at once

For this experiment, we train `HydraViT` for 300, 400, and 500 epochs with a pre-trained DeiT-tiny checkpoint. We show how our design choices, i.e., changing the number of heads coupled to the embedding dimension, weighted subnetwork sampling, and adding separate heads for each subnetwork, impact accuracy. Table 2 shows each subnetwork's accuracy for all the combinations of our design choices. Note that subnetworks of `HydraViT` with 3 heads result in the same architecture as DeiT-tiny, subnetworks with 6 heads result in the same as DeiT-small, and subnetworks with 12 heads result in the same as DeiT-base.

To evaluate weighted subnetwork sampling, we show in Table 2 that with 25% weight for training the subnetwork with 3 heads, 30% weight for 6 heads, and 45% weight for 12 heads, we can achieve an improvement of 0.3 to nearly 0.6 p.p. for the subnetwork with 12 heads depending on the number of epochs compared to uniform subnetwork sampling. Meanwhile, we get a change of -0.2 to +0.2 p.p. for the subnetwork with 6 heads and a decrease of 0.5 to 1.0 p.p. for the subnetwork with 3 heads compared to uniform subnetwork sampling. Therefore, we can increase accuracy at 12 heads at the cost of an overall accuracy decrease. Keep in mind that removing only one head in the vanilla DeiT-base significantly drops its accuracy to less than 30%, whereas `HydraViT` achieves more than 72% at 3 heads and 79% at 6 heads and is therefore more versatile.

To evaluate separate classifiers for each subnetwork, we show in Table 2 that it helps, in some cases, improve each subnetwork's accuracy by up to 0.2 percentage points. But it can also reduce the overall accuracy because each classifier gets fewer gradient updates than a shared classifier.

Finally, we can combine weighted subnetwork sampling and separate classifiers to achieve a high 12-head accuracy, reaching up to 81.77% accuracy at 500 epochs while maintaining a good accuracy at 3 and 6 heads. We notice that compared to only weighted subnetwork sampling, all the accuracies are up to 0.15 p.p. higher. Due to starting with a pre-trained DeiT-tiny, the classifier for 3 heads needs fewer gradient updates, and the weighted subnetwork sampling shifts the gradient bias to the larger subnetworks, which leads to overall better accuracy, see Table 2.

To summarize, we show that with `HydraViT`, we can create one set of weights that achieves, on average, the same accuracy as the three separate models DeiT-tiny, Deit-small, and DeiT-base. To attain this one set of weights, we need at least 300 fewer training epochs than are necessary to train the three different DeiT models. The subnetworks have identical RAM usage, throughput, MACs, and

7

Table 3: Comparison of `HydraViT` with the baselines MatFormer, DynaBERT, and SortedNet. The table shows for selected subnetworks the RAM usage, MACs, model parameters, throughput and their corresponding accuracy when trained from scratch (when applicable) and from the initial DeiT-tiny checkpoint. Note that DynaBERT relies on Knowledge Distillation in every block, which is why it reaches less than 1% accuracy when trained from scratch.

Method	Dim	RAM [MB]	MACs [G]	Params [M]	Throughput [# / s]	Acc [%] from scratch	Acc [%] from DeiT-tiny
MatFormer (Kudugunta et al., 2023)	768	508.45	17.56	86.6	1728.3	81.89	82.04
	384	366.08	11.99	58.2	2231.6	81.52	81.80
	192	294.9	9.2	44.1	2601.4	79.40	80.48
DynaBERT (Hou et al., 2020)	768	508.45	17.56	86.6	1725.7	-	81.30
	384	287.62	7.45	44.1	3014.6	-	80.16
	192	177.2	3.43	22.8	4944.5	-	73.00
SortedNet (Valipour et al., 2023)	768	508.45	17.56	86.6	1753.0	79.71	80.80
	384	169.6	4.6	22.1	3874.8	77.79	78.94
	192	63.87	1.25	5.7	5898.2	66.64	70.20
HydraViT	768	508.45	17.56	86.6	1754.1	80.45	81.10
	384	169.6	4.6	22.1	4603.6	78.40	79.28
	192	63.87	1.25	5.7	10047.6	67.34	70.56
HydraViT 800 Epochs	768	508.45	17.56	86.6	1754.1	81.93	81.60
	384	169.6	4.6	22.1	4603.6	79.84	80.15
	192	63.87	1.25	5.7	10047.6	68.78	71.67
HydraViT 9-12 Heads	768	508.45	17.56	86.6	1754.1	81.56	82.25
	704	440.19	14.82	72.9	1916.1	81.55	82.22
	640	376.63	12.31	60.3	2242.9	81.51	82.21
	576	317.8	10.04	49.0	2385.2	81.21	81.92

model parameters compared to the DeiT models. While in this section, we investigated `HydraViT` with only 3 subnetworks, we evaluate `HydraViT` with more subnetworks in the next section.

4.2 `HydraViT` vs. Baselines

For the next experiment, we train `HydraViT` and the baselines introduced at the beginning of this section for 300 epochs, once from scratch and once with DeiT-tiny as an initial checkpoint. While all of these baselines reduce the embedding dimension, the difference is they reduce it in different parts of the model. We choose 10 subnetworks for each model, setting the embedding dimension from 192 to 768 with steps of 64 in between. These steps correspond to having from 3 to 12 attention heads, with steps of 1 in between. While `HydraViT` supports up to 12 subnetworks, we choose to exclude the two smallest ones, as their accuracy drops too much.

Table 3 shows how each baseline compares to `HydraViT` relative to their RAM usage, GMACs, model parameters, and throughput when training from scratch and when starting with a pre-trained DeiT-tiny checkpoint. Figure 1 and Figure 6 show the results of all subnetworks when starting with a pre-trained DeiT-tiny checkpoint. Besides `HydraViT`, only SortedNet can run with less than 150 MB of RAM while achieving on average 0.3 to 0.7 p.p. worse accuracy than `HydraViT`. The other baselines, which have a more limited range of subnetworks, achieve a better accuracy when running at higher embedding dimensions. The limited range, however, has the downside of not having smaller subnetworks for devices with fewer resources. And if we limit `HydraViT` to a similar range as MatFormer, training on 9 to 12 heads, we show that `HydraViT` reaches the overall highest accuracy at 82.25% compared to MatFormer's 82.04%. We also notice that `HydraViT` cannot reach the exact same performance as the three DeiT models. This is because training for 10 subnetworks with a shared classifier for only 300 epochs has its toll on the overall performance. One option is to train longer, which we demonstrated for `HydraViT` with 3 subnetworks in Section 4.1. We repeat the same here and train `HydraViT` for 800 epochs, showing that even with 10 subnetworks, we can still achieve near-similar performance as the three different DeiT models. This is while having another 7 subnetworks with similar accuracy per resource trade-off points in between. One caveat, however, is that when training from scratch, `HydraViT` struggles to get a good accuracy at 3 heads. This is most likely due to a sampling bias as the subnetworks with one and two heads are not included in training and due to training hyperparameters as they differ when training DeiT-tiny compared to DeiT-base. For detailed results, e.g., each subnetwork for every baseline, See Table 4 in Appendix A.

8
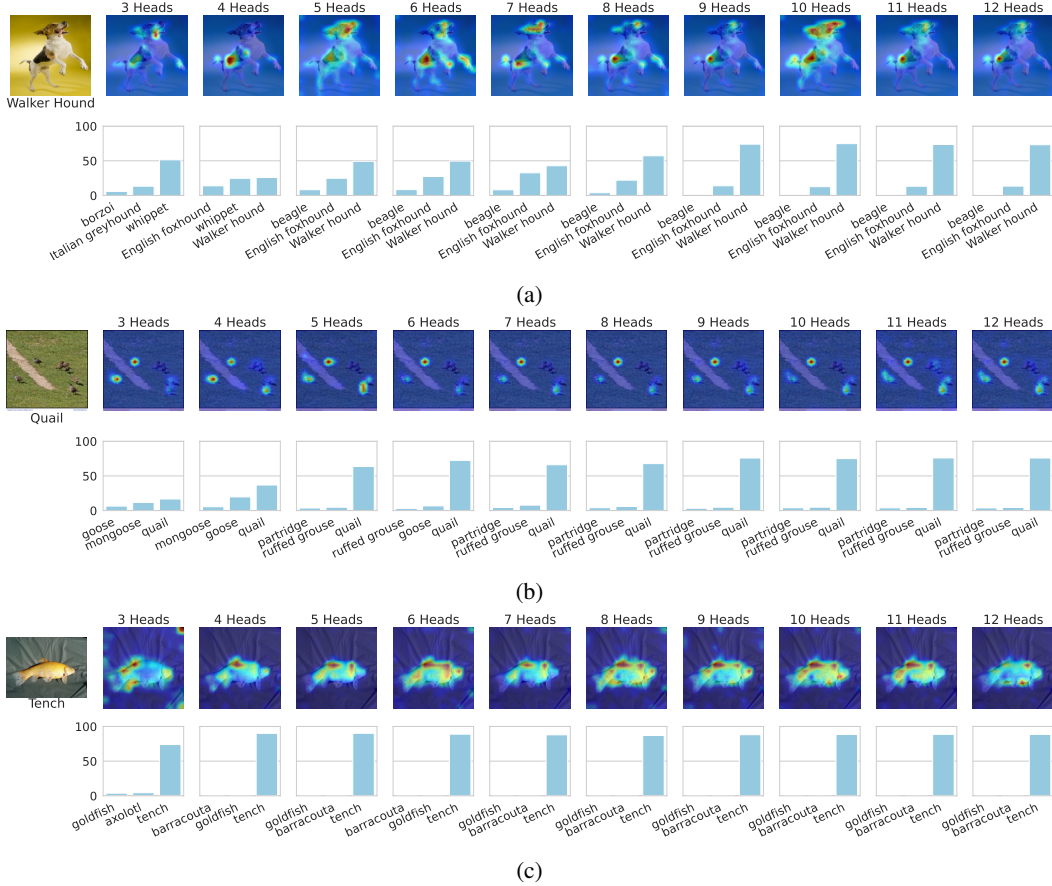
Figure 8: Attention relevance maps (Chefer et al., 2021) of 3 samples from ImageNet-1K for `HydraViT` with different number of heads. Increasing the number of heads leads to more confident classification and a more condensed attention distribution.

In summary, `HydraViT` achieves, on average, better accuracy than its baselines except for MatFormer within its limited scalability range. However, we show that training `HydraViT` on a similar scalability range outperforms MatFormer.

4.3 Analyzing `HydraViT`'s inner workings

Fig. 8 displays the attention relevance map (Chefer et al., 2021) of selected subnetworks of `HydraViT`, allowing us to visually investigate how the model's attention shifts when increasing the number of heads. Fig. 8c shows that fewer heads lead to more scattered attention, whereas increasing the number of heads makes the attention maps more compact and focused on the main object. Additionally, adding more heads enhances classification confidence. For instance, in Fig. 8a, the model misclassifies the input with 3 heads, but as we add more heads, the classification gradually shifts to the correct label and increases in confidence. We also illustrate the t-SNE visualization of the final layer for different subnetworks, see Fig. 7. The figure shows that subnetworks with more heads exhibit a denser representation, while having fewer heads results in a more sparse representation. This indicates that increasing the number of heads enhances focus on the main object, which results in less entropy and, thereby, a more compact t-SNE representation. It is worth noting that the outliers in this figure occur due to the high norm values of the embeddings (Darcet et al., 2024).

4.4 Robustness of `HydraViT`

To show the robustness of `HydraViT` we also evaluate on different ImageNet variants: ImageNet-v2 (Recht et al., 2019), ImageNet-R (Hendrycks et al., 2020), ImageNet-A (Hendrycks et al., 2019),

9
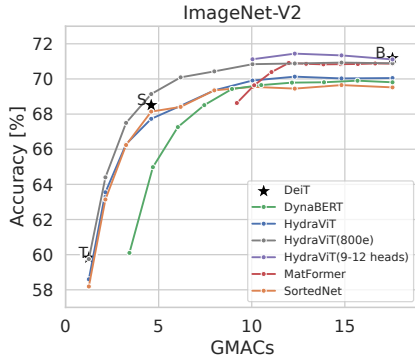
Figure 9: Performance comparison of `HydraViT` and baselines in terms of GMACs on ImageNet-v2.
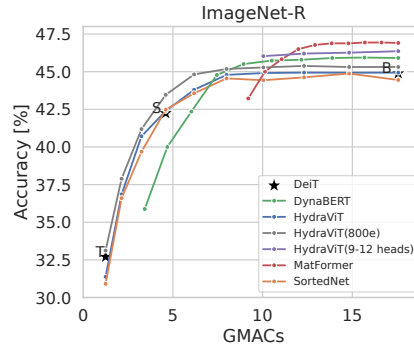
Figure 10: Performance comparison of `HydraViT` and baselines in terms of GMACs on ImageNet-R.

ImageNet-Sketch (Wang et al., 2019), and ImageNet-ReaL (Beyer et al., 2020; Russakovsky et al., 2015). On four of these five ImageNet variants, `HydraViT` achieves the overall best results. Figure 9 shows this for ImageNet-v2. Figure 10 shows the only ImageNet variant, i.e., ImageNet-R, where `HydraViT` trained with 9 to 12 heads is not able to achieve the best results. Nevertheless, `HydraViT` reaches a competitive accuracy on these difficult variants and has on average the best results. See Appendix F for full results.

4.5 Limitations

Training complexity `HydraViT` optimizes 10 loss functions simultaneously, which increases the computational load on the optimization progress. As a result, we require more training iterations to achieve accuracy comparable to that of individually trained models such as DeiT-tiny, DeiT-small, and DeiT-base. However, by training multiple models within a unified framework, `HydraViT` ultimately requires much less total training time compared to training each of these 10 models for 300 epochs individually. See Appendix C for more details on training complexity.

Evaluation on different hardware Our main focus with `HydraViT` is on the efficiency and scalability on a single device, rather than the deployment on smaller hardware. However, metrics such as GMACs and params, are consistent across different platforms. Additionally, the skeleton of `HydraViT` is identical to DeiT, and others have evaluated the latency and performance metrics of DeiT on smaller devices. For instance, FastViT (Vasu et al., 2023) evaluates DeiT on the iPhone 12 Pro, MobileViT (Mehta and Rastegari, 2023) on the iPhone 12, SPViT (Kong et al., 2022) on the ZCU102 FPGA and Galaxy S20, and GhostNetV3 (Liu et al., 2024) on the Huawei Mate 40 Pro. These studies provide insight into the expected performance and latency of `HydraViT` on different hardware, indirectly supporting our claims about `HydraViT`'s adaptability.

Evaluation on other models While `HydraViT` has been evaluated on DeiT-tiny, DeiT-small, and DeiT-base configurations, which have the same number of layers, we have not yet applied it to larger models like DeiT-large with more layers. We plan to explore this in future works.

5 Conclusion

We introduce `HydraViT`, a novel approach for achieving a scalable ViT architecture. By dynamically stacking attention heads and adjusting embedded dimensions within the MHA layer during training, `HydraViT` induces multiple subnetworks within a single model. This enables `HydraViT` to adapt to diverse hardware environments with varying resource constraints while maintaining strong performance. Our experiments on ImageNet-1K demonstrate that `HydraViT` achieves significant accuracy improvements compared to baseline approaches, with up to 5 percentage points higher accuracy at the same computational cost and up to 7 percentage points higher accuracy at the same throughput. This makes `HydraViT` a practical solution for real-world deployments where hardware availability is diverse or changes over time.

10

Acknowledgments and Disclosure of Funding

This research has received funding from the Federal Ministry for Digital and Transport under the CAPTN-Förde 5G project grant no. 45FGU139_H and the Federal Ministry for Economic Affairs and Climate Action under the Marispace-X project grant no. 68GX21002E. It was supported in part through high-performance computing resources available at the Kiel University Computing Centre.

References

Lucas Beyer, Olivier J. Henaff, Alexander Kolesnikov, Xiaohua Zhai, and Aaron van den Oord. 2020. Are we done with ImageNet? *arXiv preprint arXiv:2002.05709* (2020).

Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. 2023. Flexivit: One model for all patch sizes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14496–14506.

Arnav Chavan, Zhiqiang Shen, Zhuang Liu, Zechun Liu, Kwang-Ting Cheng, and Eric P Xing. 2022. Vision transformer slimming: Multi-dimension searching in continuous optimization space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4931–4941.

Hila Chefer, Shir Gur, and Lior Wolf. 2021. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 782–791.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. 2024. Vision Transformers Need Registers. In *The Twelfth International Conference on Learning Representations*. `https://openreview.net/forum?id=2dnO3LLiJ1`

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. `https://doi.org/10.1109/CVPR.2009.5206848`

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. `https://openreview.net/forum?id=YicbFdNTTy`

Biyi Fang, Xiao Zeng, and Mi Zhang. 2018. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 115–127.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. 2020. The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization. *arXiv preprint arXiv:2006.16241* (2020).

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. 2019. Natural Adversarial Examples. *arXiv preprint arXiv:1907.07174* (2019).

Ali Hojjat, Janek Haberer, and Olaf Landsiedel. 2023. ProgDTD: Progressive Learned Image Compression With Double-Tail-Drop Training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 1130–1139.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems* 33 (2020), 9782–9793.

Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, et al. 2022. SPViT: Enabling Faster Vision Transformers via Latency-Aware Soft Token Pruning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*. Springer, 620–640.

11

Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, Prateek Jain, et al. 2023. MatFormer: Nested Transformer for Elastic Inference. *arXiv preprint arXiv:2310.07707* (2023).

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. 2022. Matryoshka representation learning. *Advances in Neural Information Processing Systems* 35 (2022), 30233–30249.

Zhenhua Liu, Zhiwei Hao, Kai Han, Yehui Tang, and Yunhe Wang. 2024. GhostNetV3: Exploring the Training Strategies for Compact Models. arXiv:2404.11202 [cs.CV] https://arxiv.org/abs/2404.11202

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. 2022. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12009–12019.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10012–10022.

Sachin Mehta and Mohammad Rastegari. 2023. Separable Self-attention for Mobile Vision Transformers. *Transactions on Machine Learning Research* (2023). https://openreview.net/forum?id=tBl4yBEjKi

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems* 34 (2021), 13937–13949.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019. Do ImageNet Classifiers Generalize to ImageNet?. In *International Conference on Machine Learning*. 5389–5400.

Oren Rippel, Michael Gelbart, and Ryan Adams. 2014. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*. PMLR, 1746–1754.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Mohammadreza Salehi, Sachin Mehta, Aditya Kusupati, Ali Farhadi, and Hannaneh Hajishirzi. 2023. Sharcs: Efficient transformers through routing with dynamic width sub-networks. *arXiv preprint arXiv:2310.12126* (2023).

Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. 2022. Patch slimming for efficient vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12165–12174.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. 2021. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems* 34 (2021), 24261–24272.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*. PMLR, 10347–10357.

Hugo Touvron, Matthieu Cord, and Hervé Jégou. 2022. Deit iii: Revenge of the vit. In *European conference on computer vision*. Springer, 516–533.

12

Mojtaba Valipour, Mehdi Rezagholizadeh, Hossein Rajabzadeh, Marzieh Tahaei, Boxing Chen, and Ali Ghodsi. 2023. Sortednet, a place for every network and every network in its place: Towards a generalized solution for training many-in-one neural networks. *arXiv preprint arXiv:2309.00255* (2023).

Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. 2023. FastViT: A fast hybrid vision transformer using structural reparameterization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5785–5795.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. 2019. Learning Robust Global Representations by Penalizing Local Predictive Power. In *Advances in Neural Information Processing Systems*. 10506–10518.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2021b. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*. 568–578.

Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. 2021a. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. *Advances in neural information processing systems* 34 (2021), 11960–11973.

Ross Wightman. 2019. PyTorch Image Models. https://github.com/rwightman/pytorch-image-models. https://doi.org/10.5281/zenodo.4414861

Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. 2022. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10809–10818.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. 2018. Slimmable neural networks. *arXiv preprint arXiv:1812.08928* (2018).

Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. 2022. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10819–10829.

13

A   Detailed Results of Submodels

Table 4: Detailed results of all subnetworks for each Baseline and `HydraViT`. Note that DynaBERT relies on knowledge distillation in every block, which is why it reaches less than 1% accuracy when trained from scratch.

Method	Dim	RAM [MB]	MACs [G]	Params [M]	Throughput [# / s]	Acc [%] from scratch	Acc [%] from DeiT-tiny
Matformer	768	508.45	17.56	86.6	1728.3	81.89	82.04
	704	484.73	16.63	81.8	1822.5	81.89	82.04
	640	461.0	15.7	77.1	1860.3	81.89	82.04
	576	437.27	14.78	72.4	1960.1	81.87	81.99
	512	413.55	13.85	67.7	2020.7	81.79	81.95
	448	389.81	12.92	63.0	2128.4	81.66	81.94
	384	366.08	11.99	58.2	2231.6	81.52	81.80
	320	342.36	11.06	53.5	2356.4	81.09	81.71
	256	318.63	10.13	48.8	2444.8	80.47	81.25
	192	294.9	9.2	44.1	2601.4	79.40	80.48
DynaBERT	768	508.45	17.56	86.6	1725.7	-	81.30
	704	471.65	15.68	79.5	1876.8	-	81.29
	640	434.85	13.88	72.4	1995.7	-	81.20
	576	398.03	12.16	65.3	2213.7	-	81.22
	512	361.23	10.51	58.2	2412.4	-	81.05
	448	324.43	8.94	51.2	2709.3	-	80.71
	384	287.62	7.45	44.0	3014.6	-	80.16
	320	250.81	6.03	37.0	3522.5	-	78.78
	256	214.01	4.69	30.0	4157.1	-	77.04
	192	177.2	3.43	22.8	4944.5	-	73.00
SortedNet	768	508.45	17.56	86.6	1753.0	79.71	80.80
	704	440.19	14.82	72.7	1629.5	79.79	80.70
	640	376.63	12.31	60.3	1846.6	79.82	80.74
	576	317.8	10.04	49.0	2318.3	79.69	80.60
	512	263.68	7.99	38.8	2466.7	79.28	80.43
	448	214.27	6.18	29.9	2612.1	78.88	79.93
	384	169.6	4.6	22.1	3874.8	77.79	78.94
	320	129.63	3.25	15.4	3886.0	75.85	77.55
	256	94.4	2.14	10.0	4654.6	72.55	74.92
	192	63.87	1.25	5.7	5898.2	66.64	70.20
HydraViT	768	508.45	17.56	86.6	1754.1	80.45	81.10
	704	440.19	14.82	72.7	1916.1	80.47	81.08
	640	376.63	12.31	60.3	2242.9	80.42	81.03
	576	317.8	10.04	49.0	2503.2	80.36	80.99
	512	263.68	7.99	38.8	3141.5	80.03	80.80
	448	214.27	6.18	29.9	3616.4	79.45	80.35
	384	169.6	4.6	22.1	4603.6	78.40	79.28
	320	129.63	3.25	15.4	5652.0	76.66	77.80
	256	94.4	2.14	10.0	7558.2	73.39	75.40
	192	63.87	1.25	5.7	10047.6	67.34	70.56
HydraViT 800 Epochs	768	508.45	17.56	86.6	1754.1	81.93	81.60
	704	440.19	14.82	72.7	1916.1	81.90	81.57
	640	376.63	12.31	60.3	2242.9	81.84	81.63
	576	317.8	10.04	49.0	2503.2	81.73	81.48
	512	263.68	7.99	38.8	3141.5	81.54	81.36
	448	214.27	6.18	29.9	3616.4	80.98	80.92
	384	169.6	4.6	22.1	4603.6	79.84	80.15
	320	129.63	3.25	15.4	5652.0	78.07	78.43
	256	94.4	2.14	10.0	7558.2	74.83	75.95
	192	63.87	1.25	5.7	10047.6	68.78	71.67

14
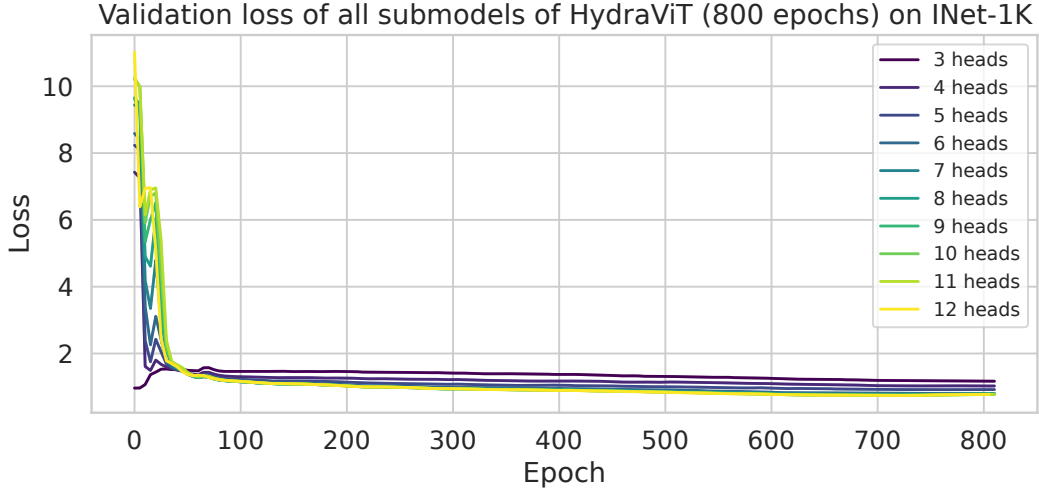
Figure 11: Training loss for all subnetworks when training `HydraViT` for 800 epochs.

Table 5: Average training throughput of `HydraViT` and DeiT models for one epoch.

Model	Throughput [#/s]
DeiT-tiny	1610
DeiT-small	821
DeiT-base	340
HydraViT	1093

B Generalization of Submodels

Figure 11 shows that we do not overfit on any subnetwork even when training for 800 epochs. Instead, in `HydraViT`, stochastic dropout training minimizes the loss more or less uniformly across all subnetworks. This is especially true when we have more subnetworks, as only one subnetwork is optimized per batch and our loss objective gets more complicated. Therefore `HydraViT` needs more epochs to reach a specific accuracy with a subnetwork than the individually trained subnetwork.

C Training Complexity

Table 5 shows that we need to take model size into account when comparing training epochs. While DeiT-tiny is by far the fastest to complete one epoch, `HydraViT` is actually on average faster to train with its 10 subnetworks than DeiT-small and DeiT-base.

D Model Loading Time

Table 6 shows that the loading time, i.e., the time it takes to load a model into RAM, is for many applications low enough to enable model switching during runtime.

Table 6: HydraViT loading times, each model was loaded six times.

Model	Heads	Dim	Latency [ms]
HydraViT	3	192	74.1 ± 1.4
HydraViT	6	384	133.3 ± 6.1
HydraViT	12	768	138.6 ± 5.8

15

Table 7: The accuracy of `HydraViT` when initialized with DeiT-tiny vs DeiT-base. While the accuracy at the 12 heads is higher with DeiT-base initialization the average accuracy is lower than with DeiT-tiny initialization.

Initialization	Epochs	Acc [%] 3 Heads	Acc [%] 6 Heads	Acc [%] 12 Heads
DeiT-tiny	400	73.16	79.63	80.90
DeiT-base	400	70.83	79.62	81.84

E   Initializing `HydraViT` with DeiT-base instead of DeiT-tiny

In `HydraViT`, the attention heads are treated as stacks, with each head built on top of the previous ones. The smallest submodel, with 3 heads, is equivalent to DeiT-tiny. Initializing with DeiT-tiny positions this 3-head submodel at its optimal starting point and ensures that larger submodels, which also contain these 3 heads, also start from a good local optimum. By employing the stochastical dropout training introduced in `HydraViT`, additional heads (4th, 5th, etc.) are trained iteratively on top of the initial three heads, creating a layered stack of attention heads. DeiT-base initialization helps `HydraViT` at 12 heads but does not yield strong smaller submodels, as Table 7 shows.

F   Results on ImageNet variants

Figure 12 shows all results of `HydraViT` and baselines in terms of GMACs and throughput on all ImageNet variants. Note that Figure 9 and Figure 10 contain the results for accuracy per GMACs on ImageNet-v2 and ImageNet-R.

16
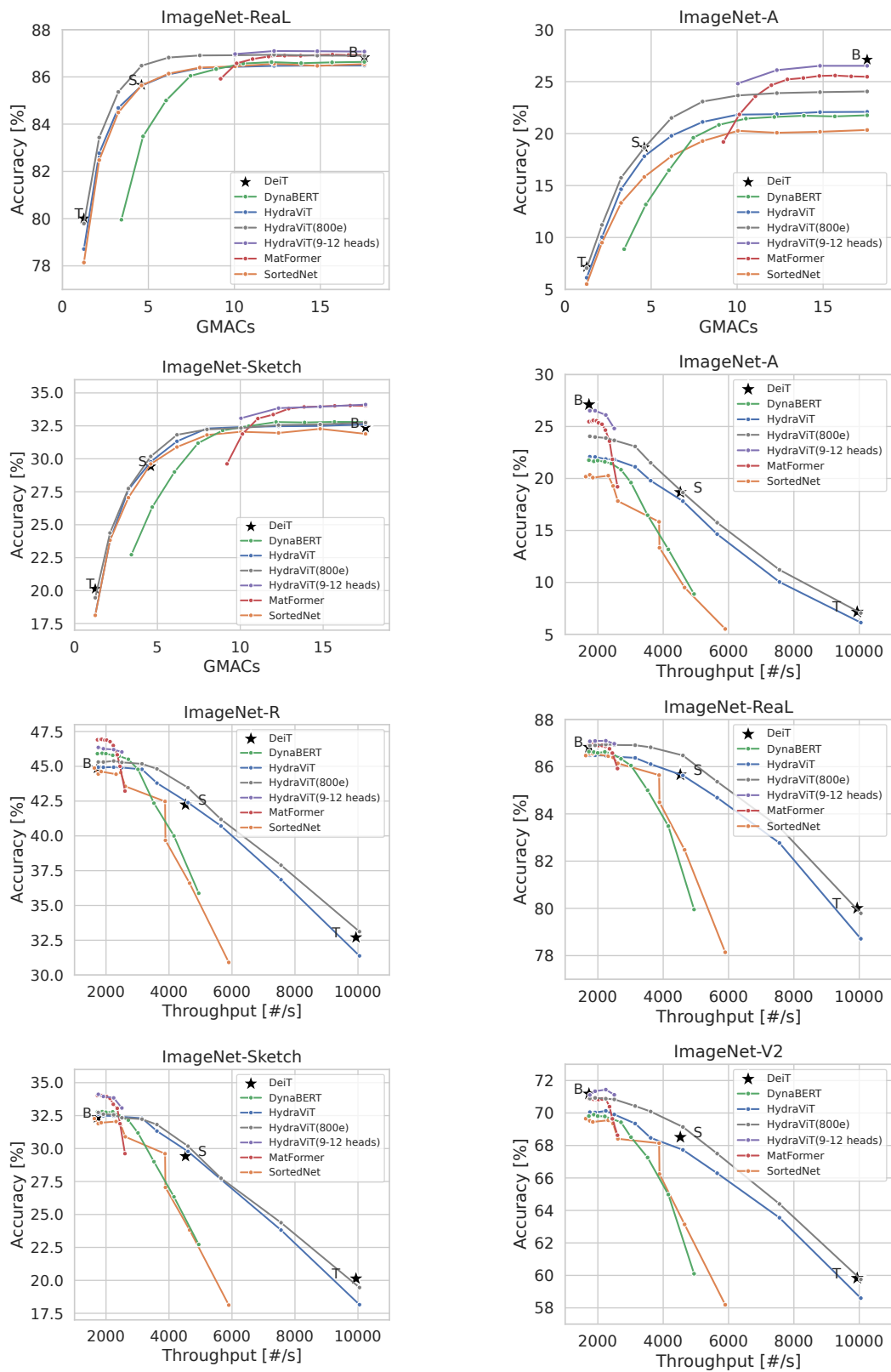
Figure 12: Full results of `HydraViT` and baselines in terms of GMACs and throughput on ImageNet variants.

17

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claim that we are introducing a scalable ViT, inducing multiple subnetworks into one model, which is what we design and then verify in the evaluation.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We mention limitations throughout the evaluation when discussing results and also include a separate section, see Section 4.5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

18

Answer: [NA]

Justification: We include no proofs and only provided a formal definition for the problem.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We disclose all the information needed to reproduce the main experimental results. We submit the code to verify the basic results as supplementary material and also open-source it at `https://github.com/ds-kiel/HydraViT`.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

19

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We open-source the code at `https://github.com/ds-kiel/HydraViT` for reproducing the results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The main experimental settings are in the paper, while we provide the code for detailed experimental settings and details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Error bars are not reported because it would be too computationally expensive.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

20

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We list the compute resources in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Our research does not involve human subjects or participants, and we adhere to licenses to prevent data-related concerns.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: HydraViT improves scalability of transformers, which makes deploying them on devices with various constraints easier. We do not believe there is a societal impact of that.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

21

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We used the ImageNet-1K dataset in the computer vision domain and believe there is no risk for misuse of the trained models.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We build on top of the pytorch-image-models repository, properly credit them, and follow their license. We also include a citation for the repository and the datasets used.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

22

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We introduce a new model and training scripts to facilitate the training, as described in Section 3. We include everything necessary in the code release, with respective documentation and licensing.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

23

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

24