WEIGHT SELECTION FOR MODEL INITIALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Weight initialization plays an important role in neural network training. Widely used initialization methods are proposed and evaluated for networks that are trained from scratch. However, the growing number of pretrained models now offers new opportunities for tackling this classical problem of weight initialization. In this work, we introduce weight selection, a method for initializing smaller models by selecting a subset of weights from a pretrained larger model. This enables the transfer of knowledge from pretrained weights to smaller models. Our experiments demonstrate that weight selection can significantly enhance the performance of small models and reduce their training time. Notably, it can also be used together with knowledge distillation. Weight selection offers a new approach to leverage the power of pretrained models in resource-constrained settings, and we hope it can be a useful tool for training small models in the large-model era.

1 INTRODUCTION

The initialization of neural network weights is crucial for their optimization. Proper initialization aids in model convergence and prevents issues like gradient vanishing. Two prominent initialization techniques, Xavier initialization (Glorot & Bengio, 2010a) and Kaiming initialization (He et al., 2015), have played substantial roles in neural network training. They remain the default methods in modern deep learning libraries like PyTorch (Paszke et al., 2019).

These methods were developed for training neural network from random initialization. At that time, it was the common practice. However, the landscape has changed. A variety of pretrained models are now readily available, thanks to collective efforts from the community (Wolf et al., 2019; Wightman, 2019). These models are trained on large datasets like ImageNet-21K (Deng et al., 2009) and LAION-5B (Schuhmann et al., 2022) and are often optimized by experts. As a result, fine-tuning from these pretrained models (Kolesnikov et al., 2020; Hu et al., 2021) is usually considered a preferred option today, rather than training models from scratch.

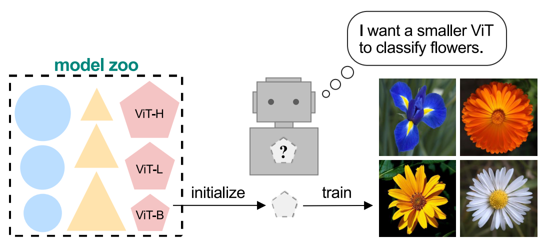


Figure 1: Large pretrained models offer new opportunities for initializing small models.

However, these pretrained large models can be prohibitive in their resource demand, preventing their wide adoption for resource-constrained settings, e.g., on mobile devices. For many pretrained model families, even the smallest model instance can be considered extremely large in certain context. For example, masked autoencoders (MAE) (He et al., 2021) and CLIP (Radford et al., 2021) both provide ViT-Base (Dosovitskiy et al., 2021), a 80M-parameter architecture, as their smallest pretrained Transformer model. This is already too large for applications on edge devices, and the smallest LLaMA (Touvron et al., 2023) model is even another 100 times larger, with 7B paramemters. With few small pretrained models available, developers would have to train them from scratch on target datasets to suit their needs. This approach misses the opportunity to utilize large pretrained models, whose knowledge is learned from extensive training on large data.

In this work, we tackle this issue by introducing a weight initialization method that uses large pretrained models to train small models. Specifically, we introduce *weight selection*, where a subset of weights from a pretrained large model is selected to initialize a smaller model. This allows for knowledge learned by the large model to transfer to the small model through its weights. Thanks to the modular design of modern neural networks, weight selection involves only three simple steps: layer selection, component mapping, and element selection. This method can be applied to any smaller model within the same model family as the large model. Using weight selection for initializing a small model is straightforward and adds no extra computational cost compared to training from scratch. It could also be useful even for large model training, e.g., initializing a LLaMA-7B with trained weights from LLaMA-30B.

We apply weight selection to train small models on image classification datasets of different scales. We observe significant improvement on accuracy across datasets and models compared with baselines. Weight selection also substantially reduces the training time required to reach the same level of accuracy. Additionally, it can work alongside another popular method for knowledge transfer from large models – knowledge distillation (Hinton et al., 2015). We believe weight selection can be a general technique for training small models. Our work also encourage further research on utilizing pretrained models for efficient deployment.

2 Related Work

Weight initialization. Weight initialization is a crucial aspect of model training. Glorot & Bengio (2010a) maintains constant variance by setting the initial values of the weights using a normal distribution, aiming to prevent gradient vanishing or explosion. Later, He et al. (2015) modified it to adapt to ReLU activations (Nair & Hinton, 2010). Mishkin & Matas (2016) crafted the orthogonality in weight matrices to keep gradient from vanishing or exploding. Saxe et al. (2013) and Vorontsov et al. (2017) put soft constraints on weight matrices to ensure orthogonality.

There are methods that use external sources of knowledge like data distribution or unsupervised training to initialize neural networks. A data-dependent initialization can be obtained from performing K-means clustering or PCA (Krähenbühl et al., 2015; Tang et al., 2017) on the training dataset. Larochelle et al. (2009), Masci et al. (2011), Trinh et al. (2019), and Gani et al. (2022) show training on unsupervised objectives can provide a better initialization for supervised training.

Utilizing pretrained models. Transfer learning (Zhuang et al., 2020) is a common framework for using model weights pretrained from large-scale data. Model architecture is maintained and the model is fine-tuned on specific downstream tasks (Kolesnikov et al., 2020). Knowledge distillation involves training usually a smaller student model to approximate the output of a teacher model (Hinton et al., 2015; Tian et al., 2019; Beyer et al., 2022). This allows the student model to maintain the performance of a teacher while being computationally efficient. Another alternative approach for using pretrained models is through weight pruning (LeCun et al., 1990; Han et al., 2015; Li et al., 2017b; Liu et al., 2018). It involves removing less significant weights from the model, making it more efficient without significantly compromising performance.

Lin et al. (2020) transforms parameters of a large network to an analogous smaller one through learnable linear layers using knowledge distillation to match block outputs. Sanh et al. (2019) and Shleifer & Rush (2020) creates smaller models by initializing with a subset of layers from a pretrained BERT (Devlin et al., 2018). This method requires the smaller model to have the same width as teacher's. Trockman et al. (2022) initializes convolutional layers with Gaussian distribution according to pretrained model's covariance. Similarly, Trockman & Kolter (2023) initializes self-attention layers according to observed diagonal patterns from pretrained ViTs. These two methods use statistics from, but do not directly utilize pretrained parameters. Weight selection, in contrast, directly utilizes pretrained parameters, does not require extra training, and is suitable for initializing any smaller variants of the pretrained model.

3 Approach

Given a pretrained model, our goal is to seek a good weight initialization for a smaller-size model within the same model family. Borrowing terminology from knowledge distillation, we refer to the pretrained model as *teacher* and the model we aim to initialize as *student*.

3.1 Weight Selection

Modern neural network architectures often follow a modular approach: design a layer and replicate it to build the model (He et al., 2016; Vaswani et al., 2017; Tolstikhin et al., 2021; Dosovitskiy et al., 2021; Liu et al., 2022). This design ethos promotes scalability: models can be widened by increasing the embedding dimension or the number of channels in each block, and deepened by stacking more layers. It also enables us to perform weight selection following three steps: selecting layers, mapping components within one layer, and selecting elements.

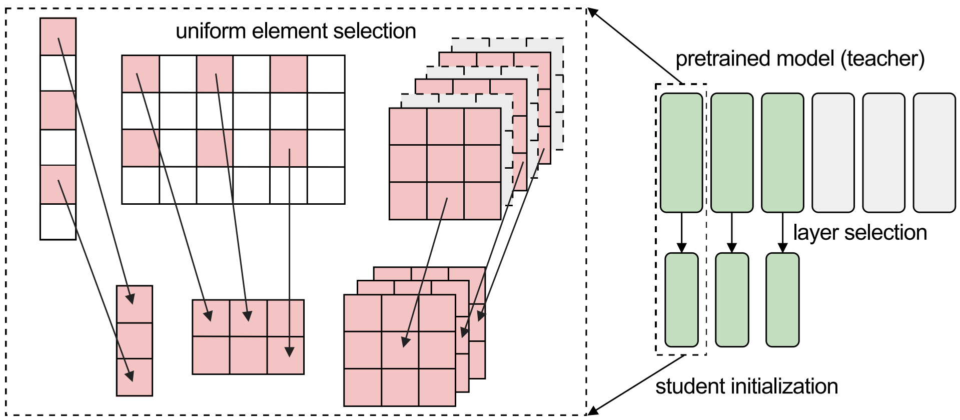


Figure 2: **Weight selection.** To initialize a smaller variant of a pretrained model, we uniformly select parameters from the corresponding component of the pretrained model.

Layer selection. Selecting layers from teacher is the first step. The procedure for layer selection is slightly different for *isotropic* architectures and *hierarchical* architectures. An isotropic architecture refers to the neural network where each layer exhibits a consistent and uniform design and behavior throughout the model. ViT (Dosovitskiy et al., 2021) and MLP-mixer (Tolstikhin et al., 2021) belong to isotropic architectures. A hierarchical architecture is characterized by multi-scale representations and a hierarchy in embedding dimensions. Hierarchical architectures typically have stages with varying scales and embedding dimensions. For example, classic convolutional networks like VGG (Simonyan & Zisserman, 2014) progressively decrease spatial dimensions while increasing channel dimensions, capturing multi-scale features. Modern architectures like Swin-Transformer (Liu et al., 2021) and ConvNeXt (Liu et al., 2022) also employ this hierarchical design.

For isotropic architectures, we select first N layers from the teacher, where N represents the student's layer count. We denote this method of layer selection as *first-N selection*. When dealing with hierarchical structures like ConvNeXt (Liu et al., 2022), first-N selection is applied at each individual stage. As an illustrative example, consider initializing ConvNeXt-F using ConvNeXt-T. Both networks comprise four stages, with depths of [2, 2, 6, 2] and [3, 3, 9, 3], respectively. A direct application of first-N selection would involve selecting the first 2 layers from the 1st, 2nd, and 4th stages, and the first 6 layers from the 3rd stage.

Component mapping. After the previous step, we have obtained the layer mapping from teacher to student. The task is then reduced to initializing one student's layer with one teacher's layer. Thanks to the modular approach adopted by modern neural network design, layers in models of the same family have an identical set of components, only differing in their width. The process for matching the corresponding components is thus a natural one-to-one mapping.

Element selection. Upon establishing component mapping, the next step is to initialize the student's component using its larger counterpart from the teacher. The default method for element selection is *uniform selection*, where we select evenly-spaced elements from teacher's tensor as shown in figure 2. Details on *uniform selection* and other element selection variants will be introduced in the next part.

3.2 Methods for element selection

In this part, we formulate element selection and introduce different selection criteria. Consider a weight tensor W_s from the student that we seek to initialize with the teacher's weight tensor W_t. If

W_t has the shape $t_1, t_2, ..., t_n$, then W_s, which is of the same component type with W_t, will also span n dimensions. Our goal is to create W_s by selecting a subset of W_t's elements. We discuss several possible implementations of element selection in this section.

Uniform selection (default). For each dimension i of W_t, select evenly-spaced s_i slices out of t_i. For example, to initialize a linear layer W_s of shape 2×3 with a linear layer W_t of shape 4×6, we select 1st and 3rd slice along the first dimension, and 1st, 3rd, and 5th slice along the second dimension. We present pseudocode for uniform selection in Algorithm 1. The algorithm starts with a copy of teacher's weight tensor W_t and iteratively perform selection on all dimensions of W_t to reach the desired shape for student. Notably, in architectures that incorporate grouped components — such as the multi-head attention module in ViTs and the grouped convolution in ResNeXt (Xie et al., 2017) — *uniform selection* absorbs information from all groups. For example, when applied on ViTs, *uniform selection* will select parameters from all heads in the attention block, which is likely to be beneficial for inheriting knowledge from the pretrained ViTs.

Algorithm 1 Uniform element selection from teacher's weight tensor

Input: W_t ▷ teacher's weight tensor
Input: s_shape ▷ desired dimension for student's weight tensor
Output: W_s with shape s_shape

procedure UNIFORMELEMENTSELECTION(W_t, student_shape)
    $W_s \leftarrow$ Copy of W_t ▷ student's weight tensor
    $n \leftarrow$ length of W_t.shape
    for $i = 1 \rightarrow n$ **do**
        $d_t \leftarrow W_t$.shape$[i]$
        $d_s \leftarrow s\_shape[i]$
        $indices \leftarrow$ Select d_s evenly spaced numbers from 1 to d_t
        $W_s \leftarrow$ Select $indices$ along W_s's i^{th}dimension
    end for
    return W_s
end procedure

Consecutive selection. For each dimension i of W_t, select *consecutive* s_i slices out of t_i. In contrast to *uniform selection*, for architectures with grouped components, *consecutive selection* select some entire groups while omitting the contrast. For architectures without such grouped components, *consecutive selection* is equivalent to *uniform selection*.

Random selection. For all weight tensors, and for each dimension i of W_t, select the same randomly-generated set of s_i slices out of t_i. This method stems from the existence of residual connections — neurons that get added in the teacher model will have their interaction preserved in the student. Furthermore, complete neurons with their inputs and outputs are preserved since only weights with consistent positions get selected. It is worth noting that *uniform selection* and *consecutive selection* are special instances of *random selection*.

Random selection (without consistency). Along every dimension i of W_t, randomly select s_i slices out of t_i. Unlike *random selection*, this method does not require selecting the same indices for every weight tensor. We create this method for comparison to examine the importance of consistency.

We compare the performance of these element selection methods in Section 4.3. We default to *uniform selection* as our method, which shows its superiority in our empirical experiments.

4 EXPERIMENTS

4.1 SETTINGS

Datasets. We evaluate weight selection on 9 image classification datasets including ImageNet-1K (Deng et al., 2009), CIFAR-10 (Krizhevsky et al., a), CIFAR-100 (Krizhevsky et al., b), Flowers (Nilsback & Zisserman, 2008), Pets (Parkhi et al., 2012), STL-10 (Coates et al., 2011), Food-101 (Bossard et al., 2014)), DTD (Cimpoi et al., 2014), SVHN (Netzer et al., 2011) and EuroSAT (Helber et al., 2019; 2018). These datasets vary in scales ranging from 5K to 1.2M training images.

dataset (scale ↓)	random init	weight selection	change	random init	weight selection	change
ImageNet-1K	73.9	75.6	↑1.6	76.1	76.4	↑0.3
SVHN	94.9	96.5	↑1.6	95.7	96.9	↑1.2
Food-101	79.6	86.9	↑7.3	86.9	89.0	↑2.1
EuroSAT	97.5	98.6	↑1.1	98.4	98.8	↑0.4
CIFAR-10	92.4	97.0	↑4.6	96.6	97.4	↑0.8
CIFAR-100	72.3	81.4	↑9.1	81.4	84.4	↑3.0
STL-10	61.5	83.4	↑21.9	81.4	92.3	↑10.9
Flowers	62.4	81.9	↑19.5	80.3	94.5	↑14.2
Pets	25.0	68.6	↑43.6	72.9	87.3	↑14.4
DTD	49.4	62.5	↑13.1	63.7	68.8	↑5.1

(a) ViT-T (b) ConvNeXt-F

Table 1: **Test accuracy on image classification datasets.** On all 9 datasets, employing weight selection for initialization leads to an improvement in test accuracy. Datasets are ordered by their image counts. Weight selection provides more benefits when evaluated on datasets with fewer images.

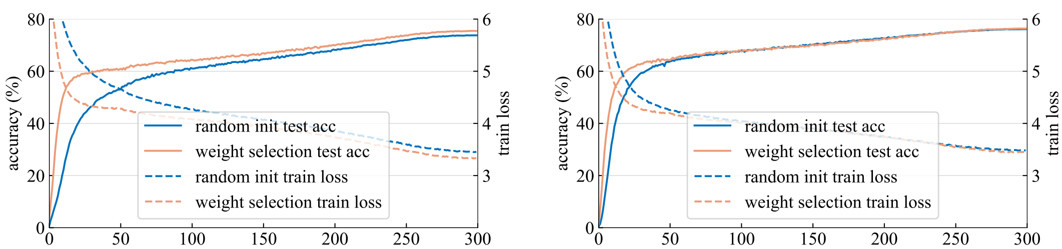
Models. We perform experiments on ViT-T/16 (Touvron et al., 2020) and ConvNeXt-F (Liu et al., 2022), with ImageNet-21K pretrained ViT-S/16 and ConvNeXt-T as their teachers respectively. We obtain weights for ImageNet-21K pretrained ViT-S/16 from Steiner et al. (2021) and ImageNet-21K pretrained ConvNeXt-T from Liu et al. (2022).

Training. We follow the training recipe from ConvNeXt (Liu et al., 2022) with adjustments to batch size, learning rate, and stochastic depth rate for different datasets. See Appendix A for details. To ensure a fair comparison, we adapt hyperparameters only for baseline training, and the same set of hyperparameters is used for training models with weight selection.

Random initialization baseline. We utilize the model-specific default initialization from the timm library (Wightman, 2019), a popular computer vision library with reliable reproducibility. Its default initialization of ViT-T and ConvNeXt-F employs a truncated normal distribution with a standard deviation of 0.02 for linear and convolution layers. The truncated normal distribution, due to its property to clip initialization values, is adopted to develop modern neural networks (Liu et al., 2022).

4.2 Results

Our experiment results are presented in Table 1. Across all 9 image classification datasets (ordered by scale in the table), weight selection consistently boosts test accuracy, especially for smaller datasets. Weight selection addresses the well-known challenge of training ViT on small datasets, which likely contributes to the significant accuracy improvement for ViT. Training curves for ImageNet-1K are shown in Figure 3. Both models benefit from weight selection early on and maintain this advantage throughout training.



(a) ViT-T (b) ConvNeXt-F

Figure 3: **Training curves on ImageNet-1K.** When initialized using weights selected from ImageNet-21K pretrained models, both ViT-T (from ViT-S) and ConvNeXt-F (from ConvNeXt-T) exhibit superior performance compared to their randomly initialized versions.

4.3 Comparisons

We compare weight selection and its variants with two widely-adopted initialization method: Xavier initialization (Glorot & Bengio, 2010b) and Kaiming initialization (He et al., 2015) as shown in Table 2. Weight selection with uniform, consecutive, and random selection yields considerably better results than all classic initialization methods. In addition, we observe a sharp drop in performance for weight selection without consistency, which proves the importance of maintaining consistency. Nonetheless, even weight selection without consistency exhibits a better performance than baseline on ViT-T and comparable performance to the best random initialization method on ConvNeXt-T.

init	ViT-T	ConvNeXt-F
timm default (trunc normal)	72.3	81.4
Xavier (Glorot & Bengio, 2010b)	72.1	82.8
Kaiming (He et al., 2015)	73	82.5
weight selection (uniform)	81.4	**84.4**
weight selection (consecutive)	**81.6**	84.1
weight selection (random)	81.2	83.8
weight selection (random w/o consistency)	77.4	82.8

Table 2: **Comparison with classic initialization methods.** Weight selection methods with consistency outperforms classic initialization methods by a large margin.

4.4 Compatibility with Knowledge Distillation

Weight selection transfers knowledge from pretrained models via parameters. Another popular approach for knowledge transferring is knowledge distillation (Hinton et al., 2015), which utilizes outputs from pretrained models. Here we explore the compatibility of these two techniques.

Settings. We evaluate the performance of combining weight selection with two different approaches in knowledge distillation – logit-based distillation and feature-based distillation. Logit-based distillation uses KL-divergence as the loss function for matching student's and teacher's logits. Denote the student's output probabilities as p_s, and teacher's output probabilities as p_t, the loss for logit-based distillation can be formulated as

$$L = L_{class} + \alpha \cdot KL(p_t || p_s) \tag{1}$$

where L_{class} is supervised loss, and α is the coefficient for distillation loss. Note that matching logits requires the teacher to be trained on the same dataset as the student. For logit-based distillation, We perform ImageNet-1K training experiments on ViT-T and use an ImageNet-1K pretrained ViT-S model from DeiT (Touvron et al., 2020) as the teacher for both knowledge distillation and weight selection. We set α to 1 in this experiment.

Feature-based distillation steps in when a classification head of target dataset is not available. Denote teacher's output as O_t, and student's output as O_s. Feature-based distillation can be formulated as

$$L = L_{class} + \alpha \cdot L_1(O_t, MLP(O_s)) \tag{2}$$

An MLP is used to project student's output to teacher's embedding dimension, and L_1 loss is used to match the projected student's output and teacher's output. For feature-based distillation, we perform CIFAR-100 training experiments on ViT-T, using ImageNet-21K pretrained ViT-S as the teacher for both knowledge distillation and weight selection. We tune α on distillation trials, and use the same value for α for experiments with both distillation and weight selection.

Results. Table 3 provides results for knowledge distillation and weight selection when applied individually or together. Introduction of logit-based distillation improves the ImageNet-1K accuracy by 0.9%. Similarly, feature-based distillation on CIFAR-100 increases accuracy by 6.4%. Without incurring additional inference cost, employing weight selection only produces a better result than the vanilla logit-based distillation and feature-based distillation. More importantly, the combination of distillation with weight selection delivers the best results, boosting accuracies to 76.0% on ImageNet-1K and 83.9% on CIFAR-100. These results further confirm weight selection's usefulness as an independent technique and the compatibility between weight selection and knowledge distillation.

setting	ImageNet-1K (logit distillation)		CIFAR-100 (feature distillation)	
	test acc	change	test acc	change
baseline	73.9	-	72.3	-
distill	74.8	↑0.9	78.4	↑6.4
weight selection	**75.5**	↑1.6	**81.4**	↑9.1
distill + weight selection	**76.0**	↑2.1	**83.9**	↑11.6

Table 3: **Compatibility with knowledge distillation.** Weight selection is useful as an independent technique, and is compatible with knowledge distillation.

5 Analysis

We perform comprehensive analysis on weight selection. Unless otherwise specified, we use weight selection from the ImageNet-21K pretrained ViT-S to initialize ViT-T, and the reported test accuracy is evaluated on CIFAR-100.

Reduction in training time. We directly measure the reduction in training time by training ViT-T with weight selection for different numbers of epochs and present the results in Figure 4. Warmup epochs is modified to maintain its ratio with total epochs. With weight selection, the same performance on CIFAR-100 can be obtained with only 1/3 epochs compared to training from random initialization.

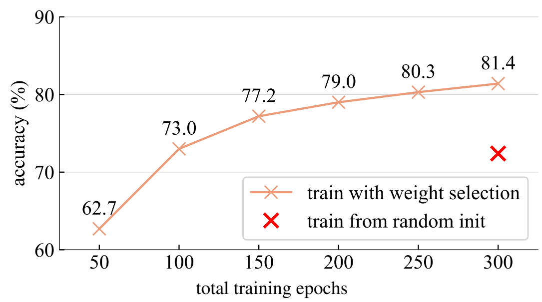


Figure 4: **Faster training.** With weight selection, ViT-T can reach the same performance on CIFAR-100 with only a third number of epochs compared to training from random initialization.

Pretrained models. We study the effect of using different pretraining models. In addition to supervised learning, vision transformers can be used as the backbone for vision-language foundation models and self-supervised learning. We evaluate the performance of ViT-B as teacher under different pretraining regimes: CLIP (Radford et al., 2021), MAE (He et al., 2021), and DINO (Caron et al., 2021). Table 4 presents the results. Across all datasets, initializing with pretrained weights consistently outperforms the random initialization. The teacher model, when pretrained with supervised learning on ImageNet-21K, provides the most effective initialization. Note that for this experiment, we use ViT-B as the teacher, since it's the smallest model that MAE and CLIP provide.

Pretrained models	CIFAR-10	CIFAR-100	STL-10
supervised (ImageNet-21K)	95.1	**77.6**	**73.1**
CLIP (Radford et al., 2021)	94.9	77.3	66.0
MAE (He et al., 2021)	**95.9**	77.2	71.0
DINO (Caron et al., 2021)	95.0	75.7	69.4

Table 4: **Different pretrained models.** Supervised teacher has the best performance.

Layer selection. Shleifer & Rush (2020) selects maximally spaced layers from BERT to initialize small models. In this part, we compare two layer selection methods: maximally spaced layer selection and first-N layer selection. To evaluate different layer selection methods, We create ViT-A which has half the depth of ViT-S. In this experiment, we use ViT-A and ConvNeXt-F as student, and ImageNet-21K pretrained ViT-S and ConvNeXt-T as their teacher for weight selection. From Table 5, we find that first-N selection performs consistently better than evenly-spaced selection on both ViT-Atto and ConvNeXt-Femto. Presumably, since layers initialized by first-N selection are naturally contiguous, they offer a more effective initialization for smaller models.

setting	ViT-A	ConvNeXt-F
random init	69.6	81.3
first-N layer selection	**77.6**	**84.4**
maximally-spaced layer selection	76.7	83.2

Table 5: **Layer selection.** First-N layer selection performs better than maximally-spaced layer selection on both ViT and ConvNeXt.

Comparison with pruning. We test the existing structured and unstructured pruning methods on reducing pretrained ViT-S to ViT-T. An important thing to note is that our setting is different from neural network pruning. Structured pruning (Li et al., 2017a) typically only prunes within residual blocks for networks with residual connections, and unstructured pruning (Han et al., 2015) prune weights by setting weights to zero instead of removing it. Despite that these pruning methods are not designed for our setting, we can extend structured and unstructured pruning methods to be applied here. Specifically, we can adopt L_1 pruning and magnitude pruning for element selection. For magnitude pruning, we squeeze the parameters into smaller matrices as the initialization for ViT-T.

We present results in Table 6. L_1 *pruning* yields better results compared to random initialization baseline. Its gap with weight selection can be explained by absence of consistency on residual connections. *Magnitude pruning* only produces marginally better results over random initialization, presumably because of the squeezing operation, which does not preserve the original structure.

setting	ViT-T	ConvNeXt-F
random init	72.3	81.4
weight selection	**81.4**	**84.4**
L_1 pruning	79.5	82.8
magnitude pruning	73.8	81.9

Table 6: **Comparison with pruning.** L_1 and *magnitude* pruning performs worse than weight selection.

teacher	params	test acc
ViT-S	22M	**81.4**
ViT-B	86M	77.6
ViT-L	307M	76.9

Table 7: **Teacher's size.** Smaller teacher provides better initialization.

Teacher's size. The size of the teacher model can be a crucial factor for the performance of weight selection. A larger teacher means a higher percentage of parameters will be discarded, which may affect the knowledge transferring process of weight selection. We present the result for using ViT-S, ViT-B, and ViT-L as the teacher in weight selection to initialize ViT-T in Table 7. Initializing from a teacher of closer size produces better results. Interestingly, even selecting 5M parameters from t301M parameters in ViT-L is effective, outperforming the random initialization baseline by 4.5%.

Linear probing. We use linear probing to directly measure the raw model's ability as a feature extractor, which can be a good indicator of the initialization quality. Linear probing is a technique used to assess a pretrained model's representations by training a linear classifier on top of the fixed features extracted from the model.

Following the recipe in He et al. (2021), we apply linear probing on CIFAR-100 to evaluate ViT-T and ConvNeXt-F initialized with weight selection from their ImageNet-21K pretrained teachers, ViT-S and ConvNeXt-T respectively. We compare the result between random initialization and weight selection with different element selection variants. As shown in Table 8, even without any training, the model from weight selection performs significantly better than random initialization as a feature extractor. Moreover, uniform, first, and random selection methods perform significantly better than inconsistent random selection, demonstrating the importance of consistency in weight selection.

setting	ViT-T	ConvNeXt-F
random init	13.5	7.1
uniform	**28.2**	**23.6**
first	26.1	22.3
random	27.2	20.7
random (w/o consistency)	23.6	13.4

Table 8: **Linear probing on CIFAR-100.** Weight selection directly produces a better feature extractor.

Longer training on ImageNet-1K. To assess if our initialization method remains beneficial for extended training durations, we use the improved training recipe from Liu et al. (2023). Specifically, the total epochs are extended to 600, and mixup / cutmix are reduced to 0.3. The results, as displayed in Table 9, affirm that our method continues to provide an advantage even under extended training durations. Both ViT-T and ConvNeXt-F, when initialized using weight selection, consistently surpass models initialized randomly. This confirms that our method does not compromise the model's capacity to benefit from longer training.

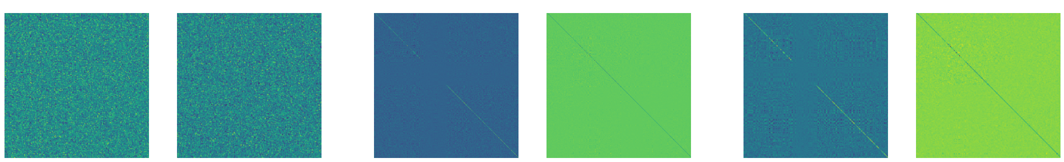
setting	ViT-T		ConvNeXt-F	
	test acc	change	test acc	change
random init	73.9	-	76.1	-
weight selection	**75.5**	↑1.6	**76.4**	↑0.3
random init (longer training)	76.3	-	77.5	-
weight selection (longer training)	**77.4**	↑1.1	**77.7**	↑0.2

Table 9: **Longer training.** Weight selection's improvement is robust under stronger recipe.

Mimetic initialization. Mimetic initialization (Trockman & Kolter, 2023) uses the diagonal properties of trained self-attention layer's weights to initialize ViTs. We present results for mimetic initialization in Table 10. Mimetic initialization improves upon random initialization baseline. By directly utilizing pretrained parameters, weight selection outperforms mimetic initialization by a large margin. In addition, we visualize the product of $W_q W_k^T$ and VW_{proj} the first head in the first attention block of ViT-T with random initialization, pretrained ViT-S, and ViT-T with weight selection. As shown in Figure 5, weight selection enables small models to inherit the desirable diagonal properties in its self-attention layers, which only exists in pretrained models.

setting	CIFAR-10	CIFAR-100	STL-10
random init	92.4	72.3	61.5
mimetic init	93.3	74.7	67.5
weight selection	**97.0**	**81.4**	**83.4**

Table 10: **Comparison with mimetic initialization.** Weight selection significantly outperforms mimetic initialization by directly utilizing pretrained parameters



(a) random init (b) pretrained ViT-S (c) weight selection

Figure 5: **Visualization of self-attention layers.** Visualization of $W_q W_k^T$ (left) and VW_{proj} (right) for ViT-T with random initialization, pretrained ViT-S, and ViT-T with weight selection. Weight selection can inherit the diagonal property of self-attention layers that only exists in pretrained ViTs.

6 Conclusion

We propose weight selection, a novel initialization method that utilizes large pretrained models. With no extra cost, it is effective for improving the accuracy of a small model and reducing its training time needed to reach a certain accuracy level. We extensively analyze its properties and compare it with alternative methods. We hope our research can inspire further exploration into algorithms for training small neural networks.

Reproducibility Statement: We provide our training recipe and hyperparameters in detail in Appendix A. Our code is available at this anonymous GitHub link: https://github.com/anonymous-wivwauyg/weight-selection.

REFERENCES

Hangbo Bao, Li Dong, and Furu Wei. BEiT: BERT pre-training of image transformers. *arXiv:2106.08254*, 2021.

Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. Knowledge distillation: A good teacher is patient and consistent. In *CVPR*, 2022.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *AISTATS*, volume 15 of *Proceedings of Machine Learning Research*, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/coates11a.html.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR Workshops*, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. doi: 10.1109/CVPR.2009.5206848.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Hanan Gani, Muzammal Naseer, and Mohammad Yaqub. How to train vision transformer on small-scale datasets? *arXiv preprint arXiv:2210.07240*, 2022.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010a.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010b.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv:2111.06377*, 2021.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 204–207. IEEE, 2018.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *ECCV*. Springer, 2020.

Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). a. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). b. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(1), 2009.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *NeurIPS*, 1990.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *ICLR*, 2017a.

Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR*, 2017b.

Ye Lin, Yanyang Li, Ziyang Wang, Bei Li, Quan Du, Tong Xiao, and Jingbo Zhu. Weight distillation: Transferring the knowledge in neural network parameters. *arXiv preprint arXiv:2009.09152*, 2020.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. 2021.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv:1810.05270*, 2018.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022.

Zhuang Liu, Zhiqiu Xu, Joseph Jin, Zhiqiang Shen, and Trevor Darrell. Dropout reduces underfitting. In *ICML*, 2023.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*, pp. 52–59. Springer, 2011.

Dmytro Mishkin and Jiri Matas. All you need is a good init. In *ICLR*, 2016.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. pp. 722–729, 12 2008. doi: 10.1109/ICVGIP.2008.47.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*. PMLR, 2021.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *NeurIPS*, 2022.

Sam Shleifer and Alexander M Rush. Pre-trained summarization distillation. *arXiv preprint arXiv:2010.13002*, 2020.

Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NeurIPS*, 2014.

Andreas Steiner, Alexander Kolesnikov, , Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

JingLei Tang, Dong Wang, ZhiGuang Zhang, LiJun He, Jing Xin, and Yang Xu. Weed identification based on k-means feature learning combined with convolutional neural network. *Computers and electronics in agriculture*, 135:63–70, 2017.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*, 2019.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. In *NeurIPS*, 2021.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv:2012.12877*, 2020.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *ICCV*, 2021.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Trieu H Trinh, Minh-Thang Luong, and Quoc V Le. Selfie: Self-supervised pretraining for image embedding. *arXiv preprint arXiv:1906.02940*, 2019.

Asher Trockman and J Zico Kolter. Mimetic initialization of self-attention layers. *arXiv preprint arXiv:2305.09828*, 2023.

Asher Trockman, Devin Willmott, and J Zico Kolter. Understanding the covariance structure of convolutional filters. *arXiv preprint arXiv:2210.03651*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. In *ICML*. PMLR, 2017.

Ross Wightman. GitHub repository: Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

APPENDIX

A TRAINING SETTINGS

Training recipe. We provide our training recipe with configurations in Table 11. The recipe is adapted from ConvNeXt (Liu et al., 2022).

Training Setting	Configuration
optimizer	AdamW
base learning rate	4e-3
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.999
batch size	4096
training epochs	300
learning rate schedule	cosine decay
warmup epochs	50
warmup schedule	linear
layer-wise lr decay (Clark et al., 2020; Bao et al., 2021)	None
randaugment (Cubuk et al., 2020)	(9, 0.5)
mixup (Zhang et al., 2018)	0.8
cutmix (Yun et al., 2019)	1.0
random erasing (Zhong et al., 2020)	0.25
label smoothing (Szegedy et al., 2016)	0.1
layer scale (Touvron et al., 2021)	1e-6
head init scale (Touvron et al., 2021)	None
gradient clip	None

Table 11: **Our basic recipe.**

Hyper-parameters. Table 12 and Table 13 record batch size, warmup epochs, and training epochs of ConvNeXt-F and Vit-T, respectively, for each dataset. The batch size of each dataset is chosen proportional to its total size. The warmup epochs are set as around one-fifth of the total training epochs. Base learning rates for ConvNeXt-F and ViT-T are 4e-3 and 2e-3 respectively.

	C-10	C-100	Pets	Flowers	STL-10	Food101	DTD	SVHN	EuroSAT	IN1k
batch size	1024	1024	128	128	128	1024	128	1024	512	4096
warmup epochs	50	50	100	100	50	50	100	10	50	50
training epochs	300	300	600	600	300	300	600	50	300	300
drop path rate	0.1	0.1	0.1	0.1	0	0.1	0.2	0.1	0.1	0

Table 12: **Hyper-parameter setting on ConvNeXt-F.**

	C-10	C-100	Pets	Flowers	STL-10	Food101	DTD	SVHN	EuroSAT	IN1k
batch size	512	512	512	512	512	512	512	512	512	4096
warmup epochs	50	50	100	100	50	50	100	10	50	50
training epochs	300	300	600	600	300	300	600	50	300	300

Table 13: **Hyper-parameter setting on ViT-T.**

B WEIGHT COMPONENTS

We conduct ablation studies on ViT-T to understand the influence of distinct model components on performance. In particular, we evaluate the performance of weight selection without one of the following particular type of layers: patch embedding, position embedding, attention block, normalization layer, or MLP layer. As illustrated in Table 14, excluding component from initialization leads to substantial drops in accuracy for all datasets. The results confirm that initializing with all components from pretrained models is necessary.

Setting	CIFAR-10	CIFAR-100	STL-10
random init	92.4	72.3	61.5
weight selection	**97.0**	**81.4**	**83.4**
w/o patch embed	96.8	79.5	77.1
w/o pos embed	95.6	78.4	80.2
w/o attention	96.2	77.3	80.5
w/o normalization	96.2	79.0	79.8
w/o mlp	95.6	78.8	74.2

Table 14: **ViT component ablation.** Using all components from pretrained models is the best.