AMixer: Adaptive Weight Mixing for Self-Attention Free Vision Transformers *Supplementary Materials*

A Implementation Details

Implementation of AMixer. We provide the pseudo code of Adaptive Weight Mixing in Algorithm 1. We apply the adaptive weight mixing to Swin Transformers [11] to construct our hierarchical models including AMixer-T, AMixer-S, and AMixer-B. The detailed configurations of our architecture variants can be found in Table 1.

Image classification. We train all our image classification models for 300 epochs using the AdamW optimizer [12] on ImageNet [4]. We set the initial learning rate as 0.001 and decay the learning rate to $1e^{-5}$ using the cosine learning rate scheduler. We set the batch size to 1024 for all models. We use a linear warm-up learning rate in the first 5 epochs following DeiT [18] and Swin [11]. For ViT-style models, we follow the regularization strategies suggested by [18], where Mixup [22] (0.8), CutMix [21] (1.0), random erasing [23] (0.25), label smoothing (0.1), RandAugment [3] (9, 0.5), repeated augmentation [6] and EMA model [14] are used during training. For hierarchical models, we do not use repeated augmentation and EMA model following the original implementation of Swin. We use the absolute positional embeddings for ViT-style models and do not add absolute/relative positional embeddings for Swin-based models. We remove the class token for all our models and use the global average pooling over all tokens to obtain the global representation of the input as in Swin. We set the stochastic depth coefficient [7] to 0.1, 0.2, 0.35 and 0.5 for AMixer-DeiT-S, AMixer-T, AMixer-S and AMixer-B, respectively. We insert the LayerScale operation [19] to the end of each block before the residual connection to ease the training of deeper models. Note that we do not add any extra convolutional layers to our models to fairly compare with DeiT and Swin, and do not use the extra regularization tricks like knowledge distillation [18] and token labeling [8] although it is useful to further improve the performance of our models.

Transfer learning. We evaluate generality of AMixer and the learned representation on several commonly used transfer learning benchmark datasets including CIFAR-10 [10], CIFAR-100 [10], Stanford Cars [9] and Flowers-102 [13]. We follow the setting of previous works [15, 5, 18, 17], where the model is initialized by the ImageNet pre-trained weights and finetuned on the new datasets. During finetuning, we use the AdamW [12] optimizer and set the weight decay to $1e^{-4}$. We use batch size 512 and a smaller initial learning rate of 0.0001 with cosine decay. We set the warm-up epoch to 5 and maximal norm for gradient clipping to

Algorithm 1 Pseudocode of Adaptive Weight Mixing.

```
class AdaptiveWeightMixing(nn.Module):
    def __init__(self, dim, h, w, n_heads, n_bank):
        self.weight_bank = nn.Parameter(torch.randn(n_bank, (2 * h - 1) * (2 * w - 1)))
        self.v_proj = nn.Linear(dim, dim)
        self.c_proj = nn.Linear(dim, dim)

        self.adapter = nn.Sequential(
            nn.Linear(dim, dim // 4),
            nn.GELU(),
            nn.Linear(dim // 4, n_bank * n_heads)
        )

    def forward(self, x):
        B, N, C = x.shape
        # weight generation
        mix_policy = self.adapter(x).view(B, N, n_bank, n_heads)
        weight_bank = self.weight_bank[:, self.rel_idx]
        weight = einsum('bnkh,knm->bnmh', mix_policy, weight_bank)
        weight = softmax(weight, dim=1)

        # spatial mixing
        x = self.v_proj(x.view(B, N, n_heads, -1))
        x = einsum('bnhc,bnmh->bmhc', x, weight)
        x = self.c_proj(x.view(B, N, C))
        return x
```

Table 1: Detailed configurations of different variants of AMixer. Our models is built based on the Swin Transformers [11] architecture. We provide the number of blocks, channels, and heads in each stage. The FLOPs are calculated with 224×224 input.

Model	#Blocks	#Channels	#Heads	#Weight Bank	Params	FLOPs
AMixer-T	[2, 2, 9, 2]	[96, 192, 384, 768]	[6, 12, 24, 48]	[9, 18, 36, 72]	26M	4.5G
AMixer-S	[2, 2, 24, 2]	[96, 192, 384, 768]	[6, 12, 24, 48]	[9, 18, 36, 72]	46M	9.0G
AMixer-B	[2, 2, 24, 2]	[128, 256, 512, 1024]	[8, 16, 32, 64]	[12, 24, 64, 96]	83M	16.0G

1 to stabilize the training and preserve the knowledge learned from pre-training. We keep the regularization methods used in ImageNet pre-training unchanged. For CIFAR-10 and CIFAR-100 with relatively more samples, we train the model for 200 epochs. For other datasets, the model is trained for 1000 epoch. Our models are trained and evaluated on commonly used splits following [15].

Semantic segmentation. We conduct the semantic segmentation experiments using MMSegmentation toolbox [2]. We follow the experiment settings in PVT [20] and Twins [1]. We train our model for 80K steps with a batch size of 16 where we use 8 GPUs with 2 images on each GPU. We set the stochastic depth coefficient to 0.2 and 0.3 for AMixer-T and AMixer-S respectively.

B More Analysis and Visualization

Effects of the choice of activation function. Compared to previous all-MLP models like MLP-Mixer [16] and ResMLP [17], one key difference of our framework is the self-attention style activation function. Therefore, we thoroughly study the effects of different activation functions in Table 2. Although

Table 2: **Effects of the choice of activation function.** We investigate the effects of different activation functions to normalize the spatial mixing weights based on our ViT-style AMixer model.

Identity	Softmax	ℓ_2-Normalize	Sigmoid	LayerNorm
80.1	**80.3**	80.2	80.1	80.0

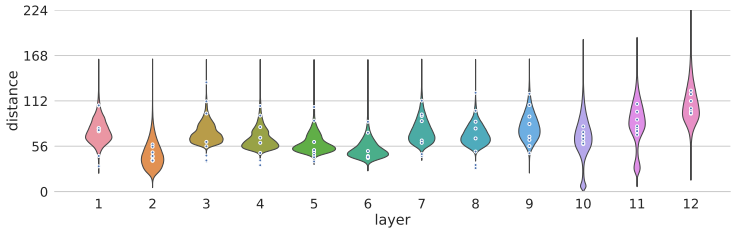

Fig. 1: **The distributions of mean attention distances in different layers.** We measure the distribution of the mean attention distances of our ViT-style AMixer model. The dots show the mean attention distances of each head.

using Softmax can achieve slightly better performance compared to no activation function, the role of activation function is much less critical in our framework.

Attention distance. We measure the distribution of the mean attention distances in different layers of our ViT-style AMixer model in Figure 1. The mean attention distance is defined by $d_{\text{mean}} = \frac{1}{HW}\sum_i \sum_j \mathbf{M}_{i,j} d_{\ell_2}(i,j)$. We see the mean attention distances gradually increase when the model becomes deeper. Our model has diverse attention patterns in different layers.

More Visualization. To have an intuitive understanding of our model, we visualize the weight bank in different layers in Figure 2, where the weights are stored based on our relative attention weight format. We show the first 6 weights of the weight bank from shallow, medium and deep layers as well as the average of all weights in the layer. We see the shallow layers usually focus on the local information while the deeper layers capture more long-range dependencies. It is interesting that although we do not impose any symmetric regularization on the weight bank, the learned weights are usually symmetric.

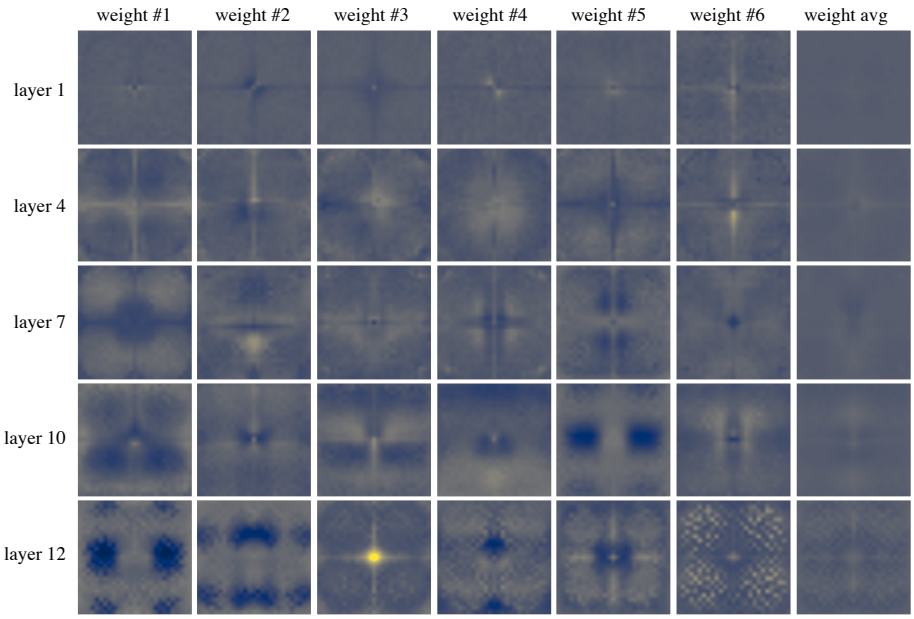

Fig. 2: **Visualization of the weight bank in different layers.** We visualize the weight bank in different layers, where the weights are stored based on our relative attention weight format. We see the shallow layers usually focus on the local information while the deeper layers capture more long-range dependencies.

References

1. Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., Xia, H., Shen, C.: Twins: Revisiting the design of spatial attention in vision transformers. In: NeurIPS 2021 (2021)
2. Contributors, M.: MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation (2020)
3. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: CVPRW. pp. 702–703 (2020)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009)
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
6. Hoffer, E., Ben-Nun, T., Hubara, I., Giladi, N., Hoefler, T., Soudry, D.: Augment your batch: Improving generalization through instance repetition. In: CVPR. pp. 8129–8138 (2020)
7. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: ECCV. pp. 646–661 (2016)
8. Jiang, Z., Hou, Q., Yuan, L., Zhou, D., Jin, X., Wang, A., Feng, J.: Token labeling: Training a 85.5% top-1 accuracy vision transformer with 56m parameters on imagenet. arXiv preprint arXiv:2104.10858 (2021)
9. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: ICCVW. pp. 554–561 (2013)
10. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
11. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030 (2021)
12. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
13. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: ICVGIP. pp. 722–729 (2008)
14. Polyak, B.T., Juditsky, A.B.: Acceleration of stochastic approximation by averaging. SIAM journal on control and optimization **30**(4), 838–855 (1992)
15. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: ICML. pp. 6105–6114. PMLR (2019)
16. Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., et al.: Mlp-mixer: An all-mlp architecture for vision. arXiv preprint arXiv:2105.01601 (2021)
17. Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Joulin, A., Synnaeve, G., Verbeek, J., Jégou, H.: Resmlp: Feedforward networks for image classification with data-efficient training. arXiv preprint arXiv:2105.03404 (2021)
18. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. arXiv preprint arXiv:2012.12877 (2020)
19. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper with image transformers. arXiv preprint arXiv:2103.17239 (2021)

20. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions (2021)
21. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: ICCV. pp. 6023–6032 (2019)
22. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)
23. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: AAAI. vol. 34, pp. 13001–13008 (2020)