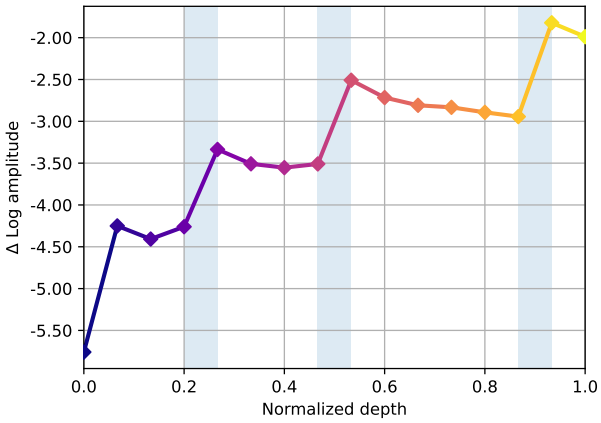
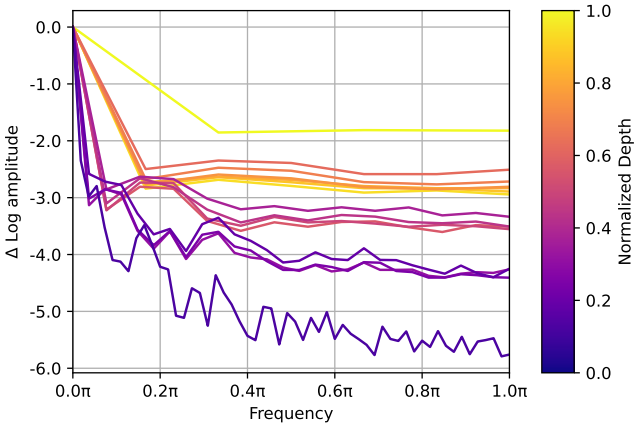
A. Appendix

A.1. Frequency Analysis Results

Reduce model size via frequency analysis As shown in Fig. 3 and Fig. 5, after the modification, the frequency spectrum is very close to the trained version of ResNet-50. The effectiveness of our approach is also illustrated in the comparison between Fig. 5 and Fig. 6, i.e., the frequency characteristics are almost the same before and after training for our modified ResNet-50.

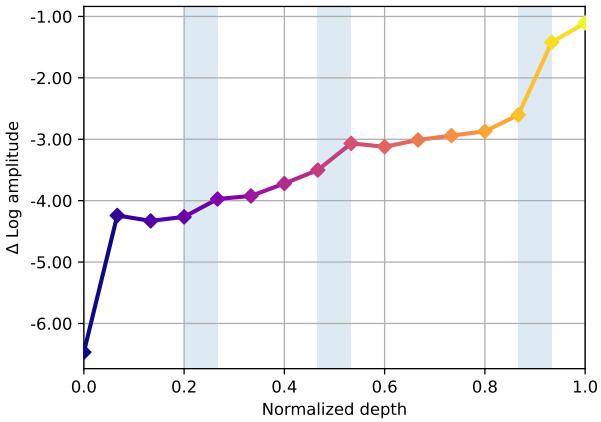


(a) Amplitude vs. Normalized Depth

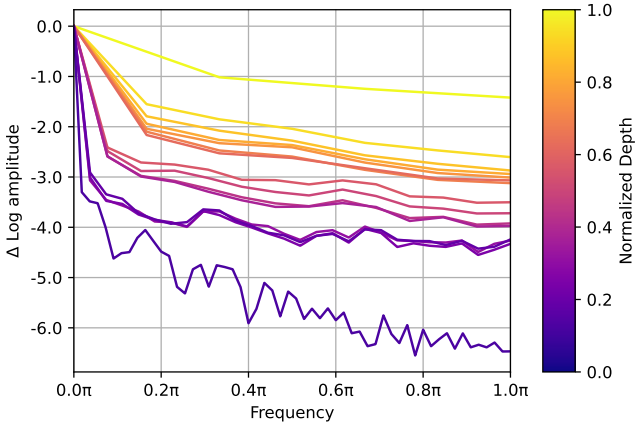


(b) Layer-wise normalized spectrum

Figure 5. Layer-wise frequency spectrum of our modified ResNet-50 before training. As shown, compared to Fig. 2, many blocks are diverging from each other; it is very close to the trained ResNet-50 (see Fig. 3). *The color-map, notations, and axes are the same as explained in Fig. 2.*



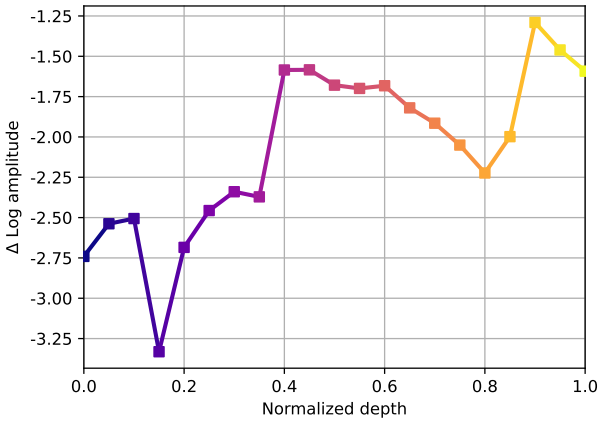
(a) Amplitude vs. Normalized Depth



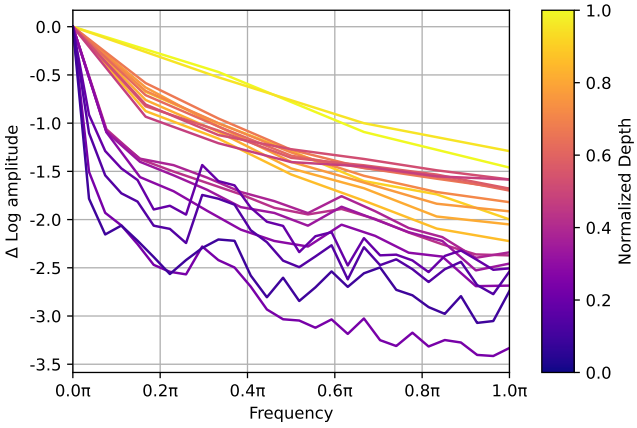
(b) Layer-wise normalized spectrum

Figure 6. Layer-wise frequency spectrum of our modified ResNet-50 after training. *The color-map, notations, and axes are the same as explained in Fig. 2.*

Why ConvNeXt works well? In our analysis, we observe that ConvNeXt shows a unique characteristic in its frequency domain representations, which is notably similar to that of Vision Transformers (ViTs). This similarity lies in how both architectures manage to capture a wide range of frequencies effectively (see Fig. 7 and Fig. 8). ConvNeXt, although fundamentally a CNN, incorporates strategies that allow it to handle global contexts and long-range dependencies, akin to the mechanisms in ViTs, especially in terms of scale in the normalized energy of the highest frequency. This is reflected in the frequency analysis where ConvNeXt demonstrates enhanced capabilities in processing both high and low-frequency features compared to traditional CNNs.

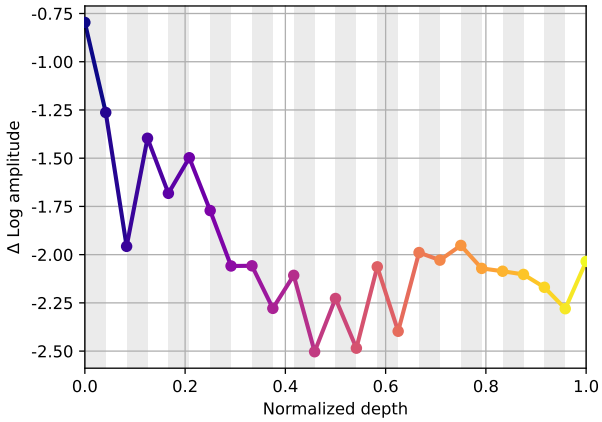


(a) Amplitude vs. Normalized Depth

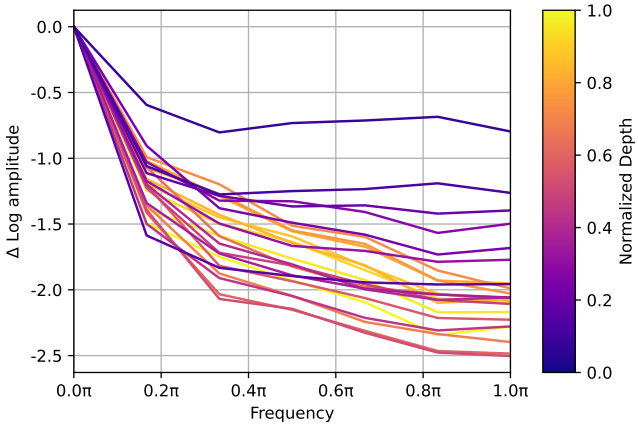


(b) Layer-wise normalized spectrum

Figure 7. Layer-wise frequency spectrum of ConvNeXt-T after training. *The color-map, notations, and axes are the same as explained in Fig. 2.*



(a) Amplitude vs. Normalized Depth

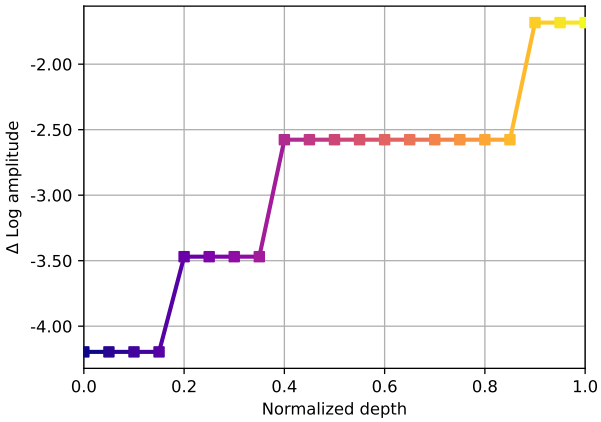


(b) Layer-wise normalized spectrum

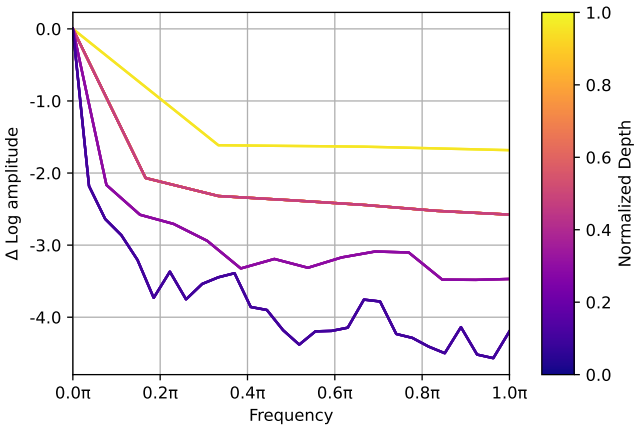
Figure 8. Layer-wise frequency spectrum of Swin-T after training. *The color-map, notations, and axes are the same as explained in Fig. 2.*

A.2. Supplementary Comparison Between ConvNeXt And Swin

In this section, we plot the frequency spectrum of ConvNeXt and Swin Transformer *before training*. As shown in Fig. 9 and Fig. 10. The frequency property of these two networks are quite different without training. However, as shown in the main paper, the frequency property is much more similar to each other after training (cf. Section A.1), especially in terms of the scale in the log-amplitude figures. This shows that these two networks are learning very similar information in frequency domain since the training dataset and task here are identical.

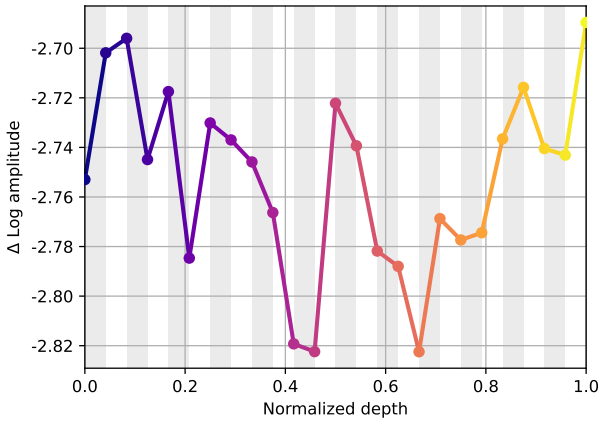


(a) Amplitude vs. Normalized Depth

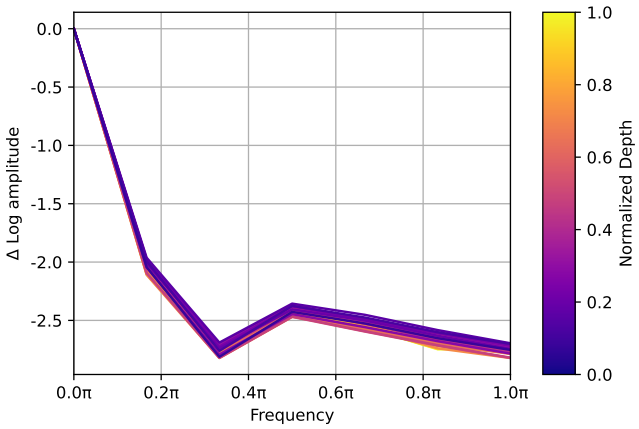


(b) Layer-wise normalized spectrum

Figure 9. Layer-wise frequency spectrum of ConvNeXt-T before training. As shown, many blocks are overlapping with each other; this is the reason why we only see four lines on the right figure.



(a) Amplitude vs. Normalized Depth

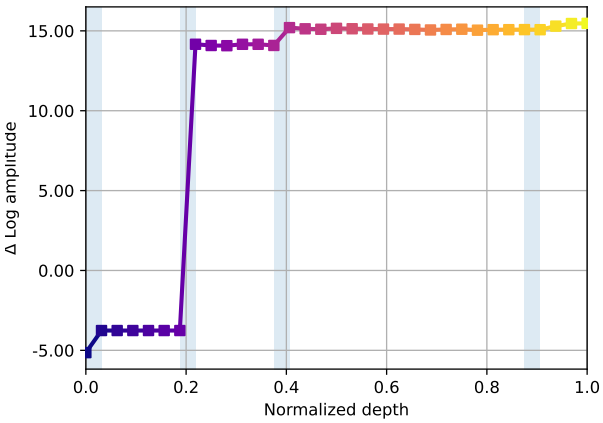


(b) Layer-wise normalized spectrum

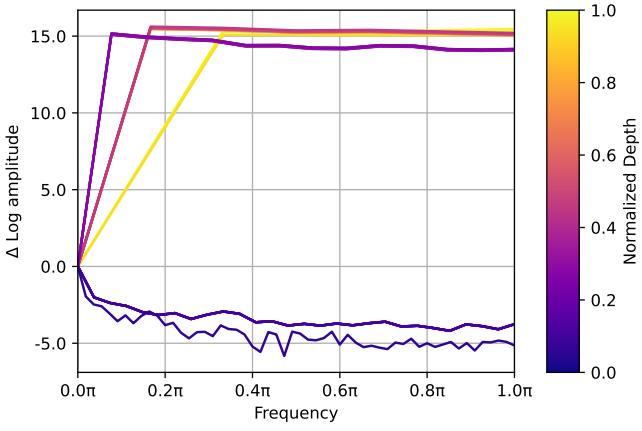
Figure 10. Layer-wise frequency spectrum of Swin-T before training. As shown, all blocks are almost overlapping with each other.

A.3. How Do CNN-GNN Models Compare?

In this section we examine the layer-wise frequency spectrum of a CNN-GNN based model, namely MobileViG-B. The layer-wise frequency spectrum of MobileViG-B before and after training is shown in Figures 11 and 12. In the layer-wise frequency spectrum we can see that after training, the frequency domain representations of MobileViG are similar to that of ViTs and ConvNeXt. MobileViG, like ConvNeXt is able to capture a wide range of frequencies. Since MobileViG is a CNN-GNN based framework it is able to handle global object interactions, similar to ViTs. Thus, MobileViG's strong performance can be attributed in part to its ability to process both high and low-frequency features better than standard CNN based models.

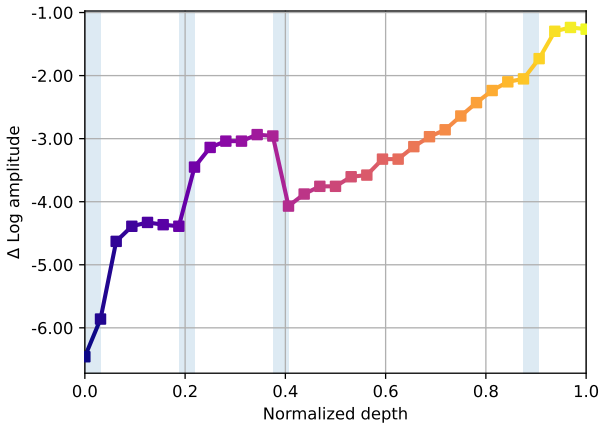


(a) Amplitude vs. Normalized Depth

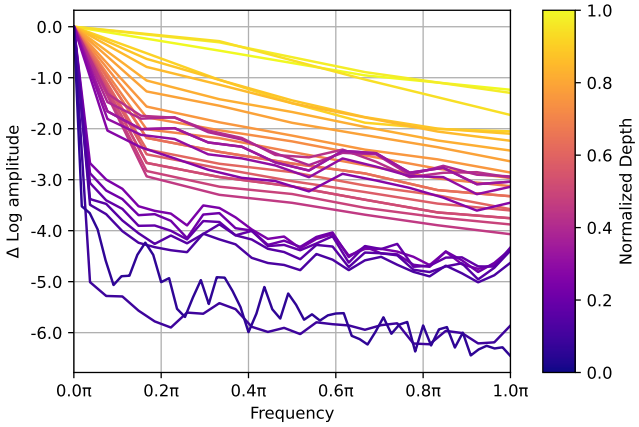


(b) Layer-wise normalized spectrum

Figure 11. Layer-wise frequency spectrum of MobileViG-B before training.



(a) Amplitude vs. Normalized Depth



(b) Layer-wise normalized spectrum

Figure 12. Layer-wise frequency spectrum of MobileViG-B after training.

A.4. Relating SNR to Log Loss

Suppose that z is the set of true labels and $\hat{z}$ the predicted probability of the true label being one determined by some optimized process. We define the Log Loss function as follows in Equation 20.

$$Log\ Loss = -\frac{1}{N}\sum_{i=1}^{N} z_i log(\hat{z}_i) + (1-z_i) log(1-\hat{z}_i) \quad (20)$$

Where z_i is the ith image's true label and $\hat{z}_i$ is the ith image's predicted probability. Given a single image z_i, its loss can be expressed as a piecewise function.

$$Log\ Loss_{z_i} = \begin{cases} -log(1-\hat{z}_i) & z_i = 0 \\ -log(\hat{z}_i) & z_i = 1 \end{cases} \quad (21)$$

Furthermore, we define the error or noise between the true label and the predicted probability as follows:

$$\epsilon = |z_i - \hat{z}_i| \quad (22)$$

Which can be expanded into Equation 23 as $\hat{z}_i \in [0, 1]$

$$\epsilon = \begin{cases} \hat{z}_i & z_i = 0 \\ 1-\hat{z}_i & z_i = 1 \end{cases} \quad (23)$$

Then, the SNR of the predicted probability $\hat{z}_i$ of a single image can be expressed in Equation 24.

$$SNR_{\hat{z}_i} = \begin{cases} \frac{E[(1-z_i)^2]}{E[\epsilon^2]} & z_i = 0 \\ \\ \frac{E[z_i^2]}{E[\epsilon^2]} & z_i = 1 \end{cases} \quad (24)$$

Which can be simplified as

$$SNR_{\hat{z}_i} = \frac{1}{E[\epsilon^2]} \quad (25)$$

Therefore, when ϵ is large, SNR is small and vice versa. Next, we substitute ϵ from Equation 23 into Equation 21 to demonstrate the relationship between SNR and Log Loss. Note that, for both cases $z_i = 0, 1$ the equation is identical.

$$Log\ Loss_{z_i} = -log(1-\epsilon) \quad (26)$$

When ϵ is large, the log loss diverges toward infinity. Conversely, when ϵ is 0, the log loss is zero.

From Equation 25, 26 it is clear that SNR and Log Loss are inversely, but not directly, correlated. That is, a high SNR indicates a low Log Loss and vice versa. While we only prove the relationship between SNR and Binary Log Loss, these steps can be easily used to generalize on multi-class log loss models with $N > 2$ number of classes.

A.5. Cifar-100 Image Classification Results

We conduct image classification experiments on the Cifar-100 [19] dataset, training from scratch for 200 epochs. We report the top-1 accuracy on the test set. We implement all models using PyTorch [39] and the Timm library [53]. We use the AdamW [28] optimizer with a cosine annealing schedule.

When testing on Cifar-100 (Table 3), ConvNeXt-T modified sees a gain of 0.9% in accuracy when the channel width is cut and the depth is increased to maintain a similar number of parameters. For Swin-T [26], we only see an increase in accuracy of 0.1% when cutting the channel width and increasing the depth to maintain a similar number of parameters. For MobileViG-Ti, when we cut the channel width and increase the depth, we see an increase in accuracy of 0.8% on the Cifar-100 dataset as well.

Table 3. Results of ConvNeXt, Swin Transformer, ResNet-50, and MobileViG on Cifar-100 image classification task. *Type* indicates whether the model is CNN-based, ViT-based, or CNN-GNN-based. *Params* lists the number of model parameters in millions. Note that the number of parameters are slightly different for Cifar-100 as the classification head is for 100 classes, not 1000 classes like in ImageNet-1k.

Model	**Modifications**	**Type**	**Params**	**Top-1 (%)**
ResNet-50 [13]	None	CNN	23.7	80.9
ResNet-50 Modified [13]	Channel Width Decreased	CNN	21.6	80.8
ConvNeXt-T [27]	None	CNN	28	82.5
ConvNeXt-T Modified Small [27]	Channel Width Decreased	CNN	12.5	81.8
ConvNeXt-T Modified [27]	Width Decreased, Depth Increased	CNN	28	83.4
MobileViG-Ti [32]	None	CNN-GNN	4.3	80.2
MobileViG-Ti Modified [32]	Width Decreased, Depth Increased	CNN-GNN	4.6	81.0
GreedyViG-S [33]	None	CNN-GNN	10.5	83.8
GreedyViG-S Modified [33]	Width Decreased, Depth Increased	CNN-GNN	11.1	84.2
Swin-T [26]	None	ViT	28	74.9
Swin-T Modified Small [26]	Channel Width Decreased	ViT	12.3	72.1
Swin-T Modified [26]	Width Decreased, Depth Increased	ViT	28	75.0

A.6. Cifar-10 Image Classification Results

We conduct image classification experiments on the Cifar-10 [19] dataset, training from scratch for 200 epochs. The Cifar-10 dataset consists of 10 object classes with 50K training images and 10K test images. We report the top-1 accuracy on the test set. We implement all of the models using PyTorch [39] and the Timm library [53]. We use the AdamW [28] optimizer with a cosine annealing schedule.

When testing on Cifar-10 in Table 4, we see an improvement in accuracy of 0.6% on MobileViG-Ti [32] when decreasing the channel width and increasing the depth of the network. When we decrease the channel width of ConvNeXt-T [27], we see a significant decrease in parameters from 28 million to 12.5 million, but a much smaller decrease in accuracy of only 0.3%. For the modified Swin-T [26], we also see a large decrease in parameters, but a small decrease in accuracy of 0.5%. Overall the results on Cifar-10 also show we can decrease model size while maintaining accuracy or improve accuracy while maintaining a similar number of parameters.

Table 4. Results of ConvNeXt, Swin Transformer, ResNet-50, and MobileViG on Cifar-10 image classification task. *Type* indicates whether the model is CNN-based, ViT-based, or CNN-GNN-based. *Params* lists the number of model parameters in millions. Note that parameter counts are slightly different for Cifar-10 as the classification head is for 10 classes, not 1000 classes like in ImageNet-1k.

Model	**Type**	**Params (M)**	**Top-1 Acc (%)**
ConvNeXt-T [27]	CNN	28	97.1
ConvNeXt-T Modified [27]	CNN	12.5	96.8
MobileViG-Ti [32]	CNN-GNN	4.3	95.6
MobileViG-Ti Modified [32]	CNN-GNN	4.6	96.2
GreedyViG-S [33]	CNN-GNN	10.5	96.7
GreedyViG-S Modified [33]	CNN-GNN	11.0	96.9
Swin-T [26]	ViT	28	91.1
Swin-T Modified [26]	ViT	12.3	90.6

A.7. Medical Image Classification Results

We perform experiments in medical image classification on two datasets, OrganSMNIST and DermaMNIST, which are part of the MedMNIST benchmark [55]. OrganSMNIST consists of 11 organs with 13,932 training and 2,452 validation abdominal CT images. DermaMNIST consists of 7 skin lesion classes with 7,007 training and 1,003 validation Dermatoscope images. We implement all of the models using Pytorch and Timm library. We train all the models from scratch for 200 epochs using AdamW optimizer with a cosine annealing schedule. We report the average top-1 accuracy on validation images.

When testing on medical image classification on OrganSMNIST [55] in Table 5, we see an improvement in accuracy of 0.4% when we modify the channel width of ResNet-50. When we modify ConvNeXt-T [27], we see a significant decrease in parameters from 28 million to 12.5 million and an accuracy increase of 0.6%. For Swin-T [26], we also see a large decrease in parameters and an increase in accuracy of 0.5%. Overall the results on OrganSMNIST also show we can decrease model size while maintaining accuracy or improve accuracy while maintaining a similar number of parameters.

Table 5. Results of ConvNeXt, Swin Transformer, MobileViG, and ResNet-50 on OrganSMNIST [55] medical image classification task. *Type* indicates whether the model is CNN-based, ViT-based, or CNN-GNN-based. *Params* lists the number of model parameters in millions. Note that parameter counts are slightly different for OrganSMNIST as the classification head is for 11 classes, not 1000 classes like in ImageNet-1k.

Model	Type	Params (M)	Top-1 Acc (%)
ResNet-50 [13]	CNN	23.7	92.6
ResNet-50 Modified [13]	CNN	21.6	93.0
ConvNeXt-T [27]	CNN	28	91.6
ConvNeXt-T Modified [27]	CNN	12.5	92.2
MobileViG-Ti [32]	CNN-GNN	4.3	90.7
MobileViG-Ti Modified [32]	CNN-GNN	4.6	91.4
GreedyViG-S [33]	CNN-GNN	10.5	91.6
GreedyViG-S Modified [33]	CNN-GNN	11.0	92.4
Swin-T [26]	ViT	28	91.7
Swin-T Modified [26]	ViT	12.3	92.2

When testing on medical image classification on DermaMNIST [55] in Table 6, we see an improvement in accuracy of 1.0% when we modify the channel width of ResNet-50. When we modify ConvNeXt-T [27], we see a significant decrease in parameters from 28 million to 12.5 million and an accuracy increase of 1.1%. For Swin-T [26], we also see a large decrease in the number of parameters by 15.7 million, but a decrease in accuracy of 0.7%. Overall the results on DermaMNIST show we can decrease model size while maintaining accuracy or improve accuracy while maintaining a similar number of parameters. The results on DermaMNIST, like OrganSMNIST, further exemplify that some of the larger networks overfit compared to the networks with their channel width reduced. The results on OrganSMNIST and DermaMNIST further demonstrate the benefits of altering network width and depth based on frequency analysis.

Table 6. Results of ConvNeXt, Swin Transformer, MobileViG, and ResNet-50 on DermaMNIST [55] medical image classification task. *Type* indicates whether the model is CNN-based, ViT-based, or CNN-GNN-based. *Params* lists the number of model parameters in millions. Note that parameter counts are slightly different for DermaMNIST as the classification head is for 7 classes, not 1000 classes like in ImageNet-1k.

Model	Type	Params (M)	Top-1 Acc (%)
ResNet-50 [13]	CNN	23.7	76.1
ResNet-50 Modified [13]	CNN	21.6	77.1
ConvNeXt-T [27]	CNN	28	78.1
ConvNeXt-T Modified [27]	CNN	12.5	79.2
MobileViG-Ti [32]	CNN-GNN	4.3	75.4
MobileViG-Ti Modified [32]	CNN-GNN	4.6	75.9
GreedyViG-S [33]	CNN-GNN	10.5	76.0
GreedyViG-S Modified [33]	CNN-GNN	11	76.3
Swin-T [26]	ViT	28	80.0
Swin-T Modified [26]	ViT	12.3	79.3