Optimising Vision Transformer Performance on Limited Datasets: A Multi-Gradient Approach - Supplementary Material

Mohsin Ali[1]*, Haider Raza[1], John Q. Gan[1], Muhammad Haris[2]
[1]School of Computer Science and Electronics Engineering, University of Essex
[2]Mohamed Bin Zayed University of Artificial Intelligence
{ma22159, h.raza, jqgan}@essex.ac.uk, muhammad.haris@mbzuai.ac.ae

1. Pseudocode for MGiT

In order to ensure the reproducibility of the proposed method (MGiT) we provide the python pseudocode on Fig 1.

```
# image_size    : Size of the image
# patch_size    : Patch size
# num_layers    : Number of ViT layers
# num_heads     : Number of heads for MHSA
# hidden_dim    : Hidden dimension
# mlp_dim       : MLP dimension
# num_classes   : Number of the classes
# epochs        : Number of epochs to train

# ω             : num_heads // ω (to reduce the MHSA heads of the auxiliary ViT)
# a             : num_layers // a (to reduce the number of layers in auxiliary ViT)
# early_epochs  : Early training epochs for auxiliary ViT

# λ             : Weight of the gradient for auxiliary ViT
# γ             : Weight decay factor


ViT=VisionTransformer()


auxiliary_ViT= VisionTransformer(
        num_layers=num_layers//a,
        num_heads=num_heads//ω)


# Training loop
for epoch in range(epochs):
    if epoch < early_epochs:
        auxiliary_ViT.train()

        # Train the auxiliary_ViT.....

        # Get the classification layer from the ViT and auxiliary_ViT
        source_layers = list(auxiliary_ViT.children())[-1]
        target_layers = list(ViT.children())[-1]

        # Transfering the weight of the auxiliary_ViT classifiaction layer
        # to ViT classifiaction layer
        for source_layer, target_layer in zip(source_layers, target_layers):
            target_dict = target_layer.state_dict()
            source_dict = source_layer.state_dict()
            for key in source_dict:
                target_layer.load_state_dict(source_layer.state_dict())

    else:

        # Check if the weight of the gradient for auxiliary ViT is greater than 0.2
        if λ > 0.2:
            # Train ViT with auxiliary .....
            ViT.train()
            auxiliary_ViT.train()

            # Train the ViT.....
            # Train the auxiliary_ViT.....

            # Get the classification layer from the ViT and auxiliary_ViT
            source_layers = list(auxiliary_ViT.children())[-1]
            target_layers = list(ViT.children())[-1]

            # Transfering the weight of the auxiliary_ViT classifiaction layer to ViT
            # using Weighted_Average mechanism
            for source_layer, target_layer in zip(source_layers, target_layers):
                source_dict = source_layer.state_dict()
                target_dict = target_layer.state_dict()
                for key in source_dict:
                    target_layer.load_state_dict(
                        Weighted_Average(source_dict[key], target_dict[key], λ)
                    )

            # Reduce the λ after some epochs

        else:
            # Continue training the ViT only with Early Stopping.....
            ViT.train()
```

Figure 1. A Python pseudocode for the implementation of MGiT

2. Plugin ViT Hyper-parameters

For training the auxiliary ViT we chose multiple hyper-parameters as detailed below:

Layer Reduction

To decrease the model's complexity, we reduced the number of layers in the auxiliary ViT using parameter α. The reduced number of layers is L/α, where L is the original ViT's number of layers. Alternative values considered for α are 1, 2, 4, 6, and 8. Notably, when $\alpha = 6$, the model exhibits a good balance between model complexity and accuracy, which provides a reduced model with 17 million parameters, 18.84 GFLOPs, while maintaining a competitive accuracy of 86.42%, as shown in Table 1.

Table 1. Effect of Layer Reduction or α on Number of Model Parameters, GFLOPs, and Accuracy

α	Parameters (Millions)	GFLOPs	Accuracy
1	88	94.44	86.45
2	45	49.08	86.38
4	24	26.4	86.13
6	17	18.84	86.42
8	10	11.28	85.07

Head Reduction

Similar to layer reduction, we decreased the number of MHSA heads in the auxiliary ViT using parameter ω, which reduces the number of heads from H to H/ω. Table 2 shows the impact of reducing the number of MHSA heads on model accuracy on CIFAR-10. Alternative values considered for ω are 1, 2, 4, and 6. When $\omega = 2$, the model achieved the highest accuracy of 86.42%.

Table 2. Impact of Head Reduction on Accuracy

ω	Accuracy
1	86.4
2	86.42
4	84.58
6	83.16

Weighted Average Mechanism

In our experiments, we explored various starting and stopping values of weight λ for the weighted average mechanism in the MGiT method for combining the weights of the auxiliary ViT and ViT/B-32. We determined the best starting weight through experimentation. Additionally, we observed that once $\lambda \leq 0.2$, the gradients from the auxiliary ViT ceased to contribute to the improvement of ViT/B-32. Consequently, we stopped the auxiliary ViT training and continued with independent training of ViT/B-32. This not only ensures the autonomy of ViT/B-32 but also aids in reducing the overall complexity of the MGiT. The starting value of λ was examined in the range of 0.5 to 0.9, as shown in Table 3. Among the trials, Trial 3 with a starting weight of 0.7 yielded the highest accuracy of 86.41%. Therefore, 0.7 was chosen as the starting weight for the MGiT method. Similarly, the stopping value of λ was examined in the range of 0.05 to 0.4. Trial 3, with a stopping weight of 0.2, achieved the highest accuracy of 86.42%. It was observed that further training with smaller λ could not further improve the performance of the MGiT, as shown in Table 4. Thus, 0.2 was chosen as the stopping weight for the MGiT method.

Table 3. Impact of Starting Weight (λ) on Accuracy

Trial	Starting Weight (λ)	Accuracy
1	0.9	85.06
2	0.8	85.12
3	0.7	86.41
4	0.6	86.21
5	0.5	85.96

Table 4. Impact of Stopping Weight (λ) on Accuracy

Trial	Stopping Weight (λ)	Accuracy
1	0.4	85.12
2	0.3	86.22
3	0.2	86.42
4	0.1	86.40
5	0.05	86.41