Make Vision Transformer Faster with Multi-Exit Architecture

-Deep Learning Final Project

Hyogon Ryu, KAIST AI, 20233477

1. Motivation

Despite achieving impressive performance, ViT faces challenges in inference due to its computationally expensive structure. While this may not be a significant concern for devices equipped with high computational power during inference, it becomes a critical issue when dealing with edge devices.

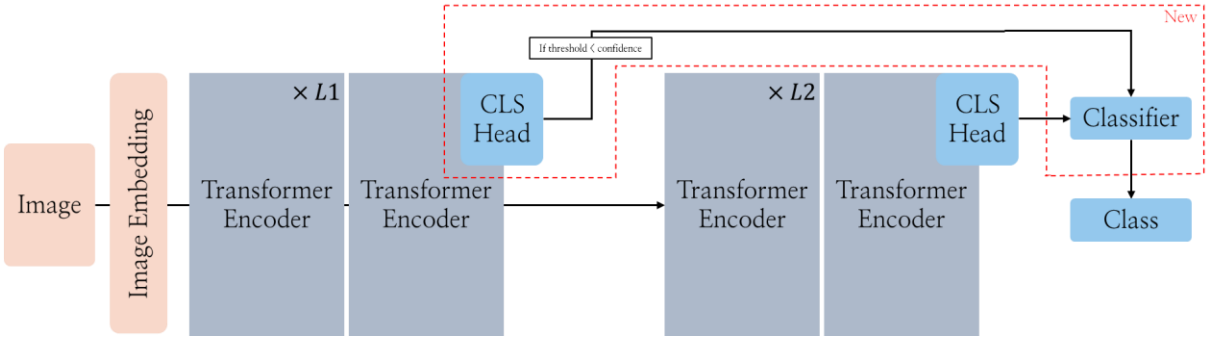
2. Goal

Improve the inference speed of the Vision Transformer by utilizing a Multi-Exit Architecture.

3. Architecture Details

Residual connection is added between the intermediate transformer block and the classifier, while keeping the rest of the architecture the same as the original ViT.

If the new residual connection produces a highly confident output, it is immediately returned as the final result without proceeding further in the inference path. This approach speeds up the inference process by classifying more images through the residual path.



[Fig1. Architecture Overview]

3.1. Implementation

```python
def forward_features(self, x):
    x = self.patch_embed(x)
    x = self._pos_embed(x)
    x = self.patch_drop(x)
    x = self.norm_pre(x)3
    if self.grad_checkpointing and not torch.jit.is_scripting():
        x = timm.models.checkpoint_seq(self.blocks, x)
    else:
        x = self.blocks[:num_blocks](x)
        output = self.forward_head(self.norm(x))
        confidence  = torch.softmax(output, dim=-1).max(dim=-1)[0].min()
        if confidence < threshold:
            x = self.blocks[num_blocks:](x)
        else:
            return output, "pre-exit"
    x = self.norm(x)
    return x

def forward(self, x):
    x = self.forward_features(x)
```

```
        if type(x) == tuple: # pre-exit
            return x[0]
        x = self.forward_head(x)
        return x
```

4. Experiments Setting

4.1. Dataset

- Imagenet-1k

Since the majority of off-the-shelf pretrained ViT models are trained on the ImageNet-1k dataset, I have also utilized the ImageNet-1k dataset in my implementation. To expedite the training process, a subset of 50,000 images from the ImageNet-1k dataset was used for training, with an additional 10,000 images allocated for validation.

4.2. Model Architecture Configuration Details

- Pretrained Model: vit_base_patch16_224 (from huggingface timm repository)
- Position of Residual Path: 3, , 9
- Classifier: Train from scratch / Fine tune / Doesn't Train
- threshold: 0.01

4.3. Metric

- Accuarcy, Inference Speed

5. Experiments Result

5.1. Inference Speed Improvement

- Objective of this Experiment: In this experiment, to assess the maximum speed improvement, I have modified the setting so that all images will be outputted at the residual connection.
- Input Data: Random Value
- Device: RTX A6000

	Original Model	Residual Path at 9th block	Residual Path at 6th block	Residual Path at 3rd block
Inference Speed(im/s)	447.10	588.89	875.14	1714.03
Speed Improvement	1.00	1.32	1.96	3.83

According to the experimental results, it was observed that passing through the residual path resulted in an improvement in inference speed. When going through the 3rd block, it was possible to expect a maximum speed improvement of up to 3.83 times.

5.2. Actual Inference Speed Improvement

- Objective of this Experiment: To evaluate the actual improvement in inference speed and potential accuracy drops
- Input Data: Imagenet-1k 10,000 images
- Device: RTX A6000

Classifier	Metric	Original Model	Residual Path at 3rd block	Residual Path at 6th block	Residual Path at 9th block
Same	Inference Speed(im/s)	439.84	438.82	439.74	440.39
	Speed Improvement	1.00	1.00	1.00	1.00
	Accuracy(%)	85.11	85.11	85.11	85.11
	Accuracy drop	0	0	0	0
Fine-Tune	Inference Speed(im/s)	440.88	440.44	440.04	440.43
	Speed Improvement	1.00	1.00	1.00	1.00
	Accuracy(%)	83.52	83.52	83.52	83.52
	Accuracy drop	0	0	0	0
Scratch	Inference Speed(im/s)	441.62	440.54	440.64	440.83
	Speed Improvement	1.00	1.00	1.00	1.00
	Accuracy(%)	81.02	80.92	80.91	80.97
	Accuracy drop	0.01	0.01	0.01	0.01

Unfortunately, contrary to our expectations, it was unable to observe performance improvements in all cases.

5.3. CPU and GPU Comparison

- Objective of this Experiment: Compare the performance on CPU and GPU
- Input Data: Imagenet-1k 10,000 images
- Device: RTX A6000

Data	Classifier	Device	Metric	Original Model	Residual Path at 3rd block	Residual Path at 6th block	Residual Path at 9th block
Imagenet-1k	Same	CPU	Inference Speed(im/s)	33.04	33.67	33.85	33.83
	Fine-Tune	CPU	Inference Speed(im/s)	28.51	28.92	32.39	31.41
			Speed Improvement	1.00	1.01	1.14	1.10
			Accuracy(%)	83.52	83.52	83.52	83.52
			Accuracy drop	0	0	0	0
		GPU	Inference Speed(im/s)	440.88	440.44	440.04	440.43
			Speed Improvement	1.00	1.00	1.00	1.00
			Accuracy(%)	83.52	83.52	83.52	83.52
			Accuracy drop	0	0	0	0
	Scratch	CPU	Inference Speed(im/s)	30.87	31.51	30.80	29.20

An interesting observation was that when conducting the experiments using a CPU, unlike when using a GPU, we were able to observe slight speed improvements in the sections marked in red. I conducted

repeated experiments, and consistently obtained the same results. Considering the characteristics of the CPU, it seems worthwhile to conduct further experiments in the future to explore this phenomenon.

6. Conclusion

This project focuses on implementing the multi-exit architecture on Vision Transformer and conducting various experiments to confirm its effectiveness. Although most of the experiments did not yield noticeable speed improvements, there were a few cases where slight improvements were observed. However, these improvements were not significant, indicating the need for further modifications to the architecture or exploration of alternative methods.