Three things everyone should know about ViTs

– Supplemental material –

A Baselines

<table>
<thead>
<tr>
<th>↓ Training procedure</th>
<th>#epochs</th>
<th>ViT-Ti</th>
<th>ViT-S</th>
<th>ViT-B</th>
<th>ViT-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeiT [62]</td>
<td>300</td>
<td>72.2</td>
<td>79.8</td>
<td>81.8</td>
<td>–</td>
</tr>
<tr>
<td>Steiner et al. [58]</td>
<td>300</td>
<td>69.6</td>
<td>76.0</td>
<td>78.7</td>
<td>74.0</td>
</tr>
<tr>
<td>He et al. [24]</td>
<td>300</td>
<td>–</td>
<td>–</td>
<td>82.1</td>
<td>81.5†</td>
</tr>
<tr>
<td>He et al. [24] with EMA</td>
<td>300</td>
<td>–</td>
<td>–</td>
<td>82.3</td>
<td>82.6†</td>
</tr>
<tr>
<td>Our baseline</td>
<td>300</td>
<td>72.7</td>
<td>79.7</td>
<td>82.2±0.06</td>
<td>83.0</td>
</tr>
<tr>
<td>Our baseline with LayerScale [64]</td>
<td>400</td>
<td>73.5</td>
<td>80.7</td>
<td>82.7</td>
<td>84.0</td>
</tr>
</tbody>
</table>

Table 8. Comparison our baseline with previous training procedures. We only include results that correspond to the vanilla ViT introduced by Dosovitskiy et al. [16] for Vit-B, Vit-L and Touvron et al. [62] for Vit-Ti and ViT-S. All models are trained on ImageNet-1k at resolution 224 × 224 without distillation. †200 epochs.

B Transfer Learning Datasets

Table 9. Datasets used in transfer experiments and corresponding references.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train size</th>
<th>Test size</th>
<th>#classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet [56]</td>
<td>1,281,167</td>
<td>50,000</td>
<td>1000</td>
</tr>
<tr>
<td>iNaturalist 2018 [28]</td>
<td>437,513</td>
<td>24,426</td>
<td>8,142</td>
</tr>
<tr>
<td>iNaturalist 2019 [28]</td>
<td>265,240</td>
<td>3,003</td>
<td>1,010</td>
</tr>
<tr>
<td>Flowers-102 [48]</td>
<td>2,040</td>
<td>6,149</td>
<td>102</td>
</tr>
<tr>
<td>Stanford Cars [34]</td>
<td>8,144</td>
<td>8,041</td>
<td>196</td>
</tr>
<tr>
<td>CIFAR-100 [36]</td>
<td>50,000</td>
<td>10,000</td>
<td>100</td>
</tr>
<tr>
<td>CIFAR-10 [36]</td>
<td>50,000</td>
<td>10,000</td>
<td>10</td>
</tr>
</tbody>
</table>
import torch
import torch.nn as nn
class hMLP_stem(nn.Module):
    """ Image to Patch Embedding """
    def __init__(self, img_size=(224,224), patch_size=(16,16), in_chans=3, embed_dim=768):
        super().__init__()
        num_patches = (img_size[1] // patch_size[1]) * (img_size[0] // patch_size[0])
        self.img_size = img_size
        self.patch_size = patch_size
        self.num_patches = num_patches
        self.proj = torch.nn.Sequential(*[
n            nn.Conv2d(in_chans, embed_dim//4, kernel_size=4, stride=4),
            nn.SyncBatchNorm(embed_dim//4),
            nn.GELU(),
            nn.Conv2d(embed_dim//4, embed_dim//4, kernel_size=2, stride=2),
            nn.SyncBatchNorm(embed_dim//4),
            nn.GELU(),
            nn.Conv2d(embed_dim//4, embed_dim, kernel_size=2, stride=2),
            nn.SyncBatchNorm(embed_dim),
        ])

    def forward(self, x):
        B, C, H, W = x.shape
        x = self.proj(x).flatten(2).transpose(1, 2)
        return x

Algorithm 1 Pseudocode of hMLP stem in a PyTorch-like style.