

ViT-MNIST Report

Wry

2021 年 7 月 6 日

第一章 程序

1.1 原理介绍

Vision Transformer(ViT) 由 Dosovitskiy 等人^[1]提出, Transformer 是自然语言处理 (Natural Language Processing, NLP) 中常见的模型, 而在计算机视觉 (Computer Vision, CV) 领域, 注意力机制的应用范围仍然较小, 通常是替换卷积神经网络 (Convolutional Neural Network, CNN) 中的一部分组件, 同时保持其整体结构不变。作者等人发现对 CNN 的依赖是毫无必要的, 在图像分类任务上单纯使用 Transformer 能够带来更好的效果。

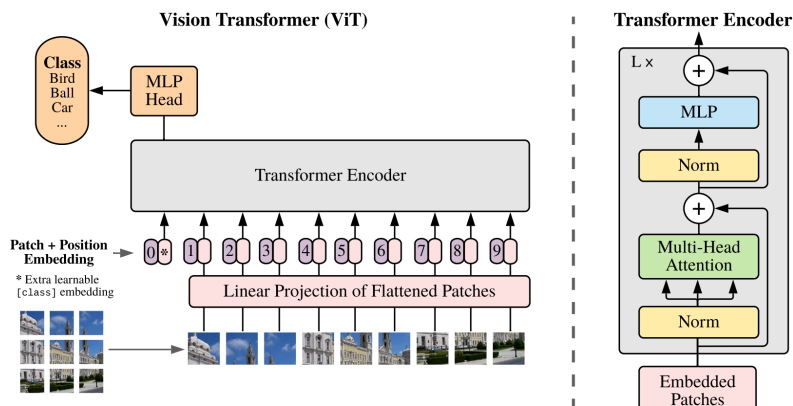


图 1.1: 模型结构

模型结构如 1.1所示, 首先目标图片被划分为几个大小相同的 patch, 线性组合后输入到一个标准的 Transformer Encoder 中, 在输入到一个多层感知器 (Multi-Layer Perceptron, MLP) 中得到分类的结果。

本实验中, ViT 的实现依赖了开源项目 `vit-pytorch`^[2]。

1.2 实现细节

如图 1.2所示是 ViT 模型中 `__init__` 方法的代码，定义了模型的结构，与 1.1所示结构一致。

```

1 def __init__(self, *, image_size, patch_size, num_classes, dim, depth, heads, mlp_dim, pool = 'cls', channels =
2   3, dim_head = 64, dropout = 0., emb_dropout = 0.):
3     super().__init__()
4     image_height, image_width = pair(image_size)
5     patch_height, patch_width = pair(patch_size)
6     assert image_height % patch_height == 0 and image_width % patch_width == 0, 'Image dimensions must be
7     divisible by the patch size.'
8     num_patches = (image_height // patch_height) * (image_width // patch_width)
9     patch_dim = channels * patch_height * patch_width
10    assert pool in {'cls', 'mean'}, 'pool type must be either cls (cls token) or mean (mean pooling)'
11
12    self.to_patch_embedding = nn.Sequential(
13        Rearrange('b c (h p1) (w p2) -> b (h w) (p1 p2 c)', p1 = patch_height, p2 = patch_width),
14        nn.Linear(patch_dim, dim),
15    )
16
17    self.pos_embedding = nn.Parameter(torch.randn(1, num_patches + 1, dim))
18    self.cls_token = nn.Parameter(torch.randn(1, 1, dim))
19    self.dropout = nn.Dropout(emb_dropout)
20
21    self.transformer = Transformer(dim, depth, heads, dim_head, mlp_dim, dropout)
22
23    self.pool = pool
24    self.to_latent = nn.Identity()
25
26    self.mlp_head = nn.Sequential(
27        nn.LayerNorm(dim),
28        nn.Linear(dim, num_classes)
29    )

```

图 1.2: `__init__` 代码

如图 1.3所示是 ViT 模型中 `forward` 方法的代码，描述了模型正向传播的过程。

如图 1.4所示是训练一个 `epoch` 的代码。

```
1 def forward(self, img):
2     x = self.to_patch_embedding(img)
3     b, n, _ = x.shape
4
5     cls_tokens = repeat(self.cls_token, '() n d -> b n d', b = b)
6     x = torch.cat((cls_tokens, x), dim=1)
7     x += self.pos_embedding[:, :(n + 1)]
8     x = self.dropout(x)
9
10    x = self.transformer(x)
11
12    x = x.mean(dim = 1) if self.pool == 'mean' else x[:, 0]
13
14    x = self.to_latent(x)
15    return self.mlp_head(x)
```

图 1.3: forward 代码

```
1 def train_epoch(model, optimizer, data_loader, loss_history):
2     total_samples = len(data_loader.dataset)
3     model.train()
4
5     for i, (data, target) in enumerate(data_loader):
6         optimizer.zero_grad()
7         output = F.log_softmax(model(data), dim=1)
8         loss = F.nll_loss(output, target)
9         loss.backward()
10        optimizer.step()
11
12        if i % 100 == 0:
13            print('[ ' + '{:5}'.format(i * len(data)) + ' / ' + '{:5}'.format(total_samples) +
14                  ' ( ' + '{:3.0f}'.format(100 * i / len(data_loader)) + '% ) ] Loss: ' +
15                  '{:6.4f}'.format(loss.item()))
16            loss_history.append(loss.item())
```

图 1.4: train_epoch 代码

第二章 结果

经过 25 个 epoch 的训练后效果如图 2.1所示。

train loss 如图 2.2所示。

test loss 如图 2.3所示。

```
Epoch: 25
[   0/60000 (  0%)] Loss: 0.0047
[10000/60000 ( 17%)] Loss: 0.0870
[20000/60000 ( 33%)] Loss: 0.0544
[30000/60000 ( 50%)] Loss: 0.0096
[40000/60000 ( 67%)] Loss: 0.0059
[50000/60000 ( 83%)] Loss: 0.0026

Average test loss: 0.0545 Accuracy: 9854/10000 (98.54%)

Execution time: 4361.09 seconds
```

图 2.1: result

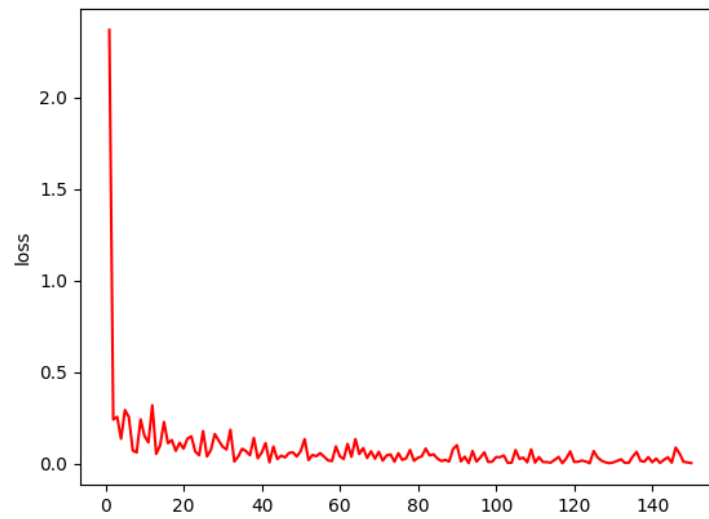


图 2.2: train loss

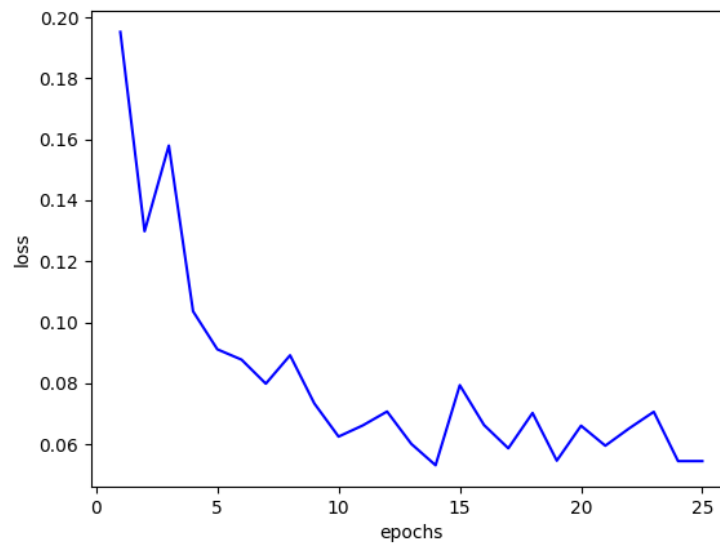


图 2.3: test loss

参考文献

- [1] DOSOVITSKIY A, BEYER L, KOLESNIKOV A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[C]//International Conference on Learning Representations. 2021.
- [2] LUCIDRAINS. vit-pytorch[EB/OL]. 2020. <https://github.com/lucidrains/vit-pytorch>.