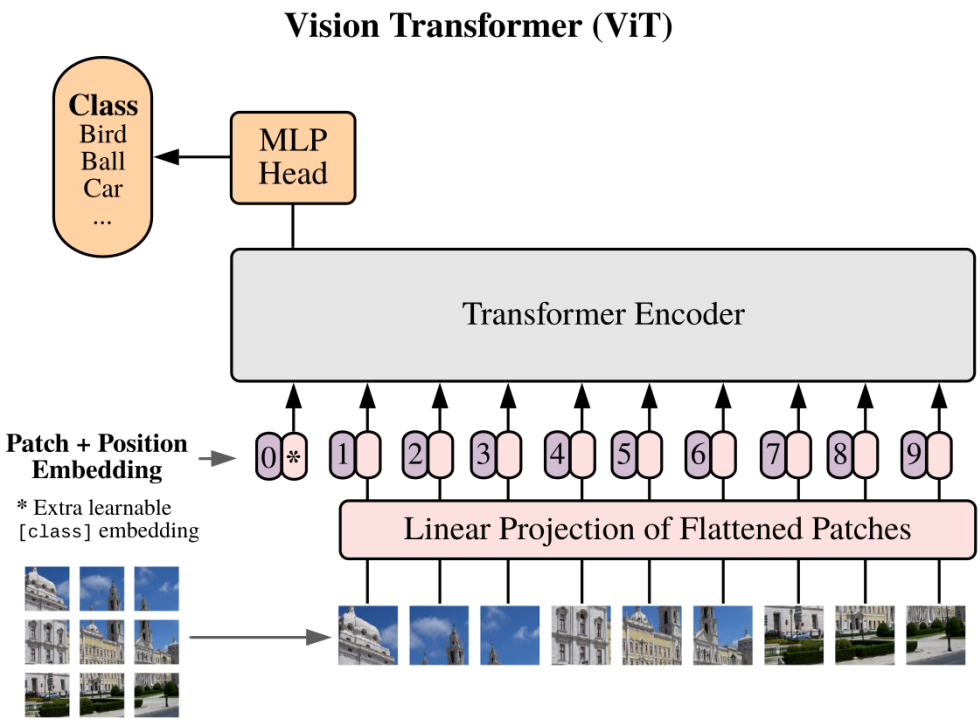
https://github.com/FrancescoSaverioZuppichini/ViT

```
import torch
import torch.nn.functional as F
import matplotlib.pyplot as plt

from torch import nn
from torch import Tensor
from PIL import Image
from torchvision.transforms import Compose, Resize, ToTensor
from einops import rearrange, reduce, repeat
from einops.layers.torch import Rearrange, Reduce
from torchsummary import summary
```

ViT Architecture



입력 이미지를 P * P 사이즈의 패치로 나눠 flatten 해 1차원 벡터 형태로 Transformer Encoder 에 입력 (Embbeding) Multi Head Attention 후 Feed Forward Layer 까지의 인코더 과정을 거침

Patch Embedding

이미지를 입력받아 패치사이즈로 나누고 1차원 벡터로 projection 시킨 후 class token 과 positional encoding

배치8 채널3 크기224*224 랜덤텐서 생성 Kernal과 Stride를 Patch_size로 갖는 Conv2D를 이용해 flatten

```
x = torch.randn(8, 3, 224, 224)
# x = torch.Size([8, 196, 768])  {batch 8 / channel 3 / 224*224}
patch_size = 16
in_channels = 3
emb_size = 768
projection = nn.Sequential(nn.Conv2d(in_channels, emb_size, kernel_size=patch_size, stride=pat
            Rearrange('b e (h) (w) -> b (h w) e'),
        )
# projection = torch.Size([8, 196, 768])

emb_size = 768
img_size = 224
patch_size = 16
```

Class Token / Positional Encoding 추가

이미지 x 를 fetch로 나누고 flatten

```
projected_x = projection(x)
# projected_x = torch.Size([8, 196, 768]
```

cls_token = 무작위로 초기화되는 torch 매개변수, forward에서 batch로 복사되고 torch.cat을 사용해 패치 앞에 추가됨 cls_token 생성, Positional Encoding Parameter 정의

```
cls_token = nn.Parameter(torch.randn(1,1, emb_size))
# cls_token = torch.Size([1, 1, 768])
positions = nn.Parameter(torch.randn((img_size // patch_size) **2 + 1, emb_size))
# positions = torch.Size([197, 768])
```

cls_token 을 반복해 배치와 크기를 맞춤

```
batch_size = 8
cls_tokens = repeat(cls_token, '() n e -> b n e', b=batch_size)
# cls_token = torch.Size([8, 1, 768])
```

cls_token 과 X 를 1차원 연결

```
cat_x = torch.cat([cls_tokens, projected_x], dim=1)
```

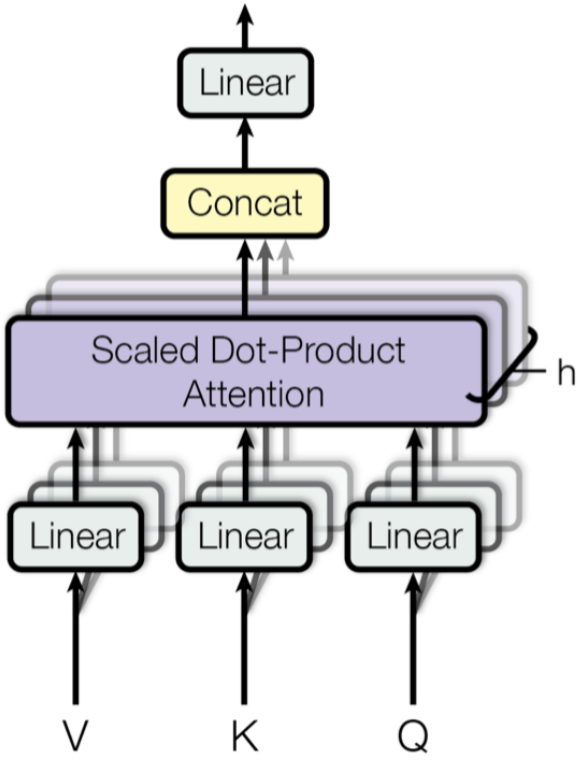
모든 배치에 position encoding 을 더해줌

```
cat_x += positions
# cat_x = torch.Size([8, 197, 768])
```

PatchEmbedding 을 클래스 형태로 구현

```
class PatchEmbedding(nn.Module):
    def __init__(self, in_channels: int = 3, patch_size: int = 16, emb_size: int = 768, img_si
        self.patch_size = patch_size
        super().__init__()
        self.projection = nn.Sequential(
            # using a conv layer instead of a linear one -> performance gains
            nn.Conv2d(in_channels, emb_size, kernel_size=patch_size, stride=patch_size),
            Rearrange('b e (h) (w) -> b (h w) e'),
        )
        self.cls_token = nn.Parameter(torch.randn(1,1, emb_size))
        self.positions = nn.Parameter(torch.randn((img_size // patch_size) **2 + 1, emb_size))

    def forward(self, x: Tensor) -> Tensor:
        b, _, _, _ = x.shape
        x = self.projection(x)
        cls_tokens = repeat(self.cls_token, '() n e -> b n e', b=b)
        # prepend the cls token to the input
        x = torch.cat([cls_tokens, x], dim=1)
        # add position embedding
        x += self.positions

        return x
```

Multy Head Attention (MHA)



3개의 Linear Projection 을 통해 임베딩된 후, 여러개의 head로 나눠져 각각 Scaled Dot-Product Attention 진행

ViT에서 Querie Key Value는 같은 텐서로 입력됨

Linear Projection

embedding 된 입력 텐서를 임베딩 사이즈로 Linear Projection K/Q/V 레이어 생성, 각 레이어는 모델 훈련과정에 학습됨

```
emb_size = 768
num_heads = 8

keys = nn.Linear(emb_size, emb_size)
queries = nn.Linear(emb_size, emb_size)
values = nn.Linear(emb_size, emb_size)

# K,Q,V = in_features=768, out_features=768, bias=True
```

Multi-Head

Linear Projection 된 Q/K/V 를 8개의 Multi Head 로 나눔

```
queries = rearrange(queries(x), "b n (h d) -> b h n d", h=num_heads)
keys = rearrange(keys(x), "b n (h d) -> b h n d", h=num_heads)
values  = rearrange(values(x), "b n (h d) -> b h n d", h=num_heads)
```

```
# Q,K,V = torch.Size([8, 8, 197, 96])
```

Scaled Dot Product Attention

Q * K 및 내적

```
energy = torch.einsum('bhqd, bhkd -> bhqk', queries, keys)

# energy = torch.Size([8, 8, 197, 197])
```

스케일링 후 Attention Score 와 V 를 내적 , embedding size 로 크기변환

```
# Get Attention Score
scaling = emb_size ** (1/2)
att = F.softmax(energy, dim=-1) / scaling
# att = torch.Size([8, 8, 197, 197])

# Attention Score * values
out = torch.einsum('bhal, bhlv -> bhav ', att, values)
# out = torch.Size([8, 8, 197, 96])

out = rearrange(out, "b h n d -> b n (h d)")
# out = torch.Size([8, 197, 768])
```

Multi Head Attention을 클래스 형태로 구현

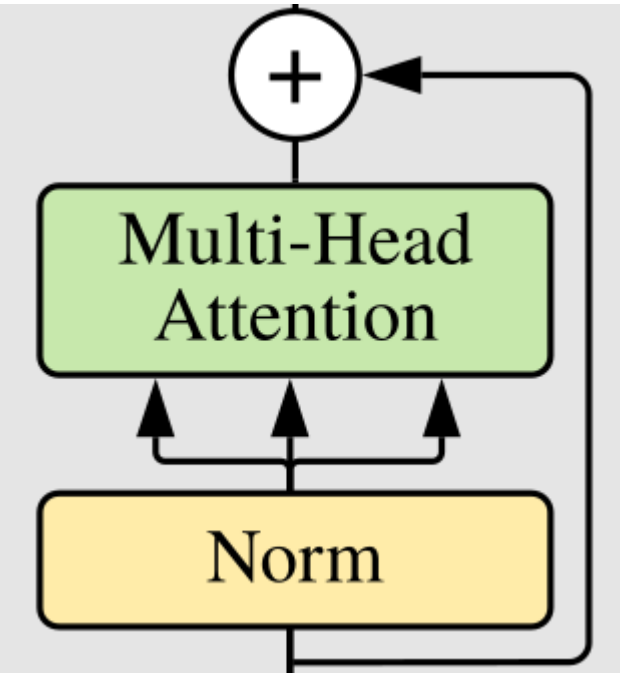
Q/K/V 각각 1개씩의 Linear 설정 대신 emb_size * 3으로 설정 후 연산시에 분배

```
# Class 로 구현한 MHA
class MultiHeadAttention(nn.Module):
    def __init__(self, emb_size: int = 768, num_heads: int = 8, dropout: float = 0):
        super().__init__()
        self.emb_size = emb_size
        self.num_heads = num_heads
        # fuse the queries, keys and values in one matrix
        self.qkv = nn.Linear(emb_size, emb_size * 3)
        self.att_drop = nn.Dropout(dropout)
        self.projection = nn.Linear(emb_size, emb_size)

    def forward(self, x : Tensor, mask: Tensor = None) -> Tensor:
        # split keys, queries and values in num_heads
        qkv = rearrange(self.qkv(x), "b n (h d qkv) -> (qkv) b h n d", h=self.num_heads, qkv=3
        queries, keys, values = qkv[0], qkv[1], qkv[2]
        # sum up over the last axis
        energy = torch.einsum('bhqd, bhkd -> bhqk', queries, keys) # batch, num_heads, query_l
        if mask is not None:
            fill value = torch.finfo(torch.float32).min
```

```
            energy.mask_fill(~mask, fill_value)

        scaling = self.emb_size ** (1/2)
        att = F.softmax(energy, dim=-1) / scaling
        att = self.att_drop(att)
        # sum up over the third axis
        out = torch.einsum('bhal, bhlv -> bhav ', att, values)
        out = rearrange(out, "b h n d -> b n (h d)")
        out = self.projection(out)
        return out
```

ViT Encoder

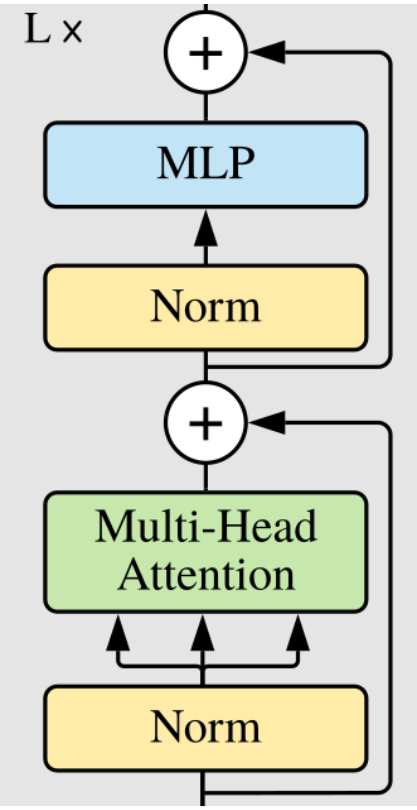
Residual Connection



fn을 입력받아 사용 fn을 forward 후 res를 더함

```
class ResidualAdd(nn.Module):
    def __init__(self, fn):
        super().__init__()
        self.fn = fn

    def forward(self, x, **kwargs):
        res = x
        x = self.fn(x, **kwargs)
        x += res
        return x
```

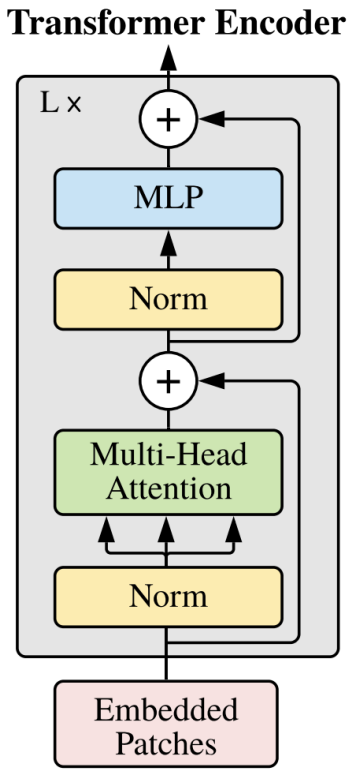
Free Forward MLP



Attention 의 output 이 FCL 로 전달됨

```
class FeedForwardBlock(nn.Sequential):
    def __init__(self, emb_size: int, expansion: int = 4, drop_p: float = 0.):
        super().__init__(
            nn.Linear(emb_size, expansion * emb_size),
            nn.GELU(),
            nn.Dropout(drop_p),
            nn.Linear(expansion * emb_size, emb_size),
        )
```

Transformer Encoder 구현



구현한 클래스들을 결합해 Transformer encoder blcok을 구현하고 블록을 depth만큼 중첩시켜 Transformer Encoder 구현

```
class TransformerEncoderBlock(nn.Sequential):
    def __init__(self,
                 emb_size: int = 768,
                 drop_p: float = 0.,
                 forward_expansion: int = 4,
                 forward_drop_p: float = 0.,
                 ** kwargs):
        super().__init__(
            ResidualAdd(nn.Sequential(
                nn.LayerNorm(emb_size),
                MultiHeadAttention(emb_size, **kwargs),
                nn.Dropout(drop_p)
            )),
            ResidualAdd(nn.Sequential(
                nn.LayerNorm(emb_size),
                FeedForwardBlock(
                    emb_size, expansion=forward_expansion, drop_p=forward_drop_p),
                nn.Dropout(drop_p)
            )
            ))

class TransformerEncoder(nn.Sequential):
    def __init__(self, depth: int = 12, **kwargs):
        super().__init__(*[TransformerEncoderBlock(**kwargs) for _ in range(depth)])
```

Head Layer

FCL Classification 수행

```
class ClassificationHead(nn.Sequential):
    def __init__(self, emb_size: int = 768, n_classes: int = 1000):
        super().__init__(
            Reduce('b n e -> b e', reduction='mean'),
            nn.LayerNorm(emb_size),
            nn.Linear(emb_size, n_classes))


class ViT(nn.Sequential):
    def __init__(self,
                 in_channels: int = 3,
                 patch_size: int = 16,
                 emb_size: int = 768,
                 img_size: int = 224,
                 depth: int = 12,
                 n_classes: int = 1000,
                 **kwargs):
        super().__init__(
            PatchEmbedding(in_channels, patch_size, emb_size, img_size),
            TransformerEncoder(depth, emb_size=emb_size, **kwargs),
            ClassificationHead(emb_size, n_classes)
        )

ViT_model = ViT().cuda()

#summary(ViT_model, (3, 224, 224))


summary(ViT_model, (3, 224, 224))
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 768, 14, 14]         590,592
         Rearrange-2             [-1, 196, 768]               0
    PatchEmbedding-3             [-1, 197, 768]               0
         LayerNorm-4             [-1, 197, 768]           1,536
            Linear-5            [-1, 197, 2304]       1,771,776
           Dropout-6          [-1, 8, 197, 197]               0
            Linear-7             [-1, 197, 768]         590,592
MultiHeadAttention-8             [-1, 197, 768]               0
           Dropout-9             [-1, 197, 768]               0
      ResidualAdd-10             [-1, 197, 768]               0
        LayerNorm-11             [-1, 197, 768]           1,536
           Linear-12            [-1, 197, 3072]       2,362,368
             GELU-13            [-1, 197, 3072]               0
          Dropout-14            [-1, 197, 3072]               0
           Linear-15             [-1, 197, 768]       2,360,064
          Dropout-16             [-1, 197, 768]               0
      ResidualAdd-17             [-1, 197, 768]               0
```

```
         LayerNorm-18            [-1, 197, 768]           1,536
            Linear-19           [-1, 197, 2304]       1,771,776
           Dropout-20         [-1, 8, 197, 197]               0
            Linear-21            [-1, 197, 768]         590,592
  MultiHeadAttention-22            [-1, 197, 768]               0
           Dropout-23            [-1, 197, 768]               0
       ResidualAdd-24            [-1, 197, 768]               0
         LayerNorm-25            [-1, 197, 768]           1,536
            Linear-26           [-1, 197, 3072]       2,362,368
              GELU-27           [-1, 197, 3072]               0
           Dropout-28           [-1, 197, 3072]               0
            Linear-29            [-1, 197, 768]       2,360,064
           Dropout-30            [-1, 197, 768]               0
       ResidualAdd-31            [-1, 197, 768]               0
         LayerNorm-32            [-1, 197, 768]           1,536
            Linear-33           [-1, 197, 2304]       1,771,776
           Dropout-34         [-1, 8, 197, 197]               0
            Linear-35            [-1, 197, 768]         590,592
  MultiHeadAttention-36            [-1, 197, 768]               0
           Dropout-37            [-1, 197, 768]               0
       ResidualAdd-38            [-1, 197, 768]               0
         LayerNorm-39            [-1, 197, 768]           1,536
            Linear-40           [-1, 197, 3072]       2,362,368
              GELU-41           [-1, 197, 3072]               0
           Dropout-42           [-1, 197, 3072]               0
            Linear-43            [-1, 197, 768]       2,360,064
           Dropout-44            [-1, 197, 768]               0
       ResidualAdd-45            [-1, 197, 768]               0
         LayerNorm-46            [-1, 197, 768]           1,536
            Linear-47           [-1, 197, 2304]       1,771,776
           Dropout-48         [-1, 8, 197, 197]               0
            Linear-49            [-1, 197, 768]         590,592
  MultiHeadAttention-50            [-1, 197, 768]               0
           Dropout-51            [-1, 197, 768]               0
       ResidualAdd-52            [-1, 197, 768]               0
         LayerNorm-53            [-1, 197, 768]           1,536
            Linear-54           [-1, 197, 3072]       2,362,368
              GELU-55           [-1, 197, 3072]               0
           Dropout-56           [-1, 197, 3072]               0
            Linear-57            [-1, 197, 768]       2,360,064
           Dropout-58            [-1, 197, 768]               0
       ResidualAdd-59            [-1, 197, 768]               0
         LayerNorm-60            [-1, 197, 768]           1,536
            Linear-61           [-1, 197, 2304]       1,771,776
           Dropout-62         [-1, 8, 197, 197]               0
            Linear-63            [-1, 197, 768]         590,592
  MultiHeadAttention-64            [-1, 197, 768]               0
           Dropout-65            [-1, 197, 768]               0
       ResidualAdd-66            [-1, 197, 768]               0
         LayerNorm-67            [-1, 197, 768]           1,536
            Linear-68           [-1, 197, 3072]       2,362,368
              GELU-69           [-1, 197, 3072]               0
           Dropout-70           [-1, 197, 3072]               0
```

```
            Linear-71            [-1, 197, 768]       2,360,064
           Dropout-72            [-1, 197, 768]               0
       ResidualAdd-73            [-1, 197, 768]               0
         LayerNorm-74            [-1, 197, 768]           1,536
            Linear-75           [-1, 197, 2304]       1,771,776
           Dropout-76         [-1, 8, 197, 197]               0
            Linear-77            [-1, 197, 768]         590,592
  MultiHeadAttention-78            [-1, 197, 768]               0
           Dropout-79            [-1, 197, 768]               0
       ResidualAdd-80            [-1, 197, 768]               0
         LayerNorm-81            [-1, 197, 768]           1,536
            Linear-82           [-1, 197, 3072]       2,362,368
              GELU-83           [-1, 197, 3072]               0
           Dropout-84           [-1, 197, 3072]               0
            Linear-85            [-1, 197, 768]       2,360,064
           Dropout-86            [-1, 197, 768]               0
       ResidualAdd-87            [-1, 197, 768]               0
         LayerNorm-88            [-1, 197, 768]           1,536
            Linear-89           [-1, 197, 2304]       1,771,776
           Dropout-90         [-1, 8, 197, 197]               0
            Linear-91            [-1, 197, 768]         590,592
  MultiHeadAttention-92            [-1, 197, 768]               0
           Dropout-93            [-1, 197, 768]               0
       ResidualAdd-94            [-1, 197, 768]               0
         LayerNorm-95            [-1, 197, 768]           1,536
            Linear-96           [-1, 197, 3072]       2,362,368
              GELU-97           [-1, 197, 3072]               0
           Dropout-98           [-1, 197, 3072]               0
            Linear-99            [-1, 197, 768]       2,360,064
          Dropout-100            [-1, 197, 768]               0
      ResidualAdd-101            [-1, 197, 768]               0
        LayerNorm-102            [-1, 197, 768]           1,536
           Linear-103           [-1, 197, 2304]       1,771,776
          Dropout-104         [-1, 8, 197, 197]               0
           Linear-105            [-1, 197, 768]         590,592
  MultiHeadAttention-106            [-1, 197, 768]               0
          Dropout-107            [-1, 197, 768]               0
      ResidualAdd-108            [-1, 197, 768]               0
        LayerNorm-109            [-1, 197, 768]           1,536
           Linear-110           [-1, 197, 3072]       2,362,368
             GELU-111           [-1, 197, 3072]               0
          Dropout-112           [-1, 197, 3072]               0
           Linear-113            [-1, 197, 768]       2,360,064
          Dropout-114            [-1, 197, 768]               0
      ResidualAdd-115            [-1, 197, 768]               0
        LayerNorm-116            [-1, 197, 768]           1,536
           Linear-117           [-1, 197, 2304]       1,771,776
          Dropout-118         [-1, 8, 197, 197]               0
           Linear-119            [-1, 197, 768]         590,592
  MultiHeadAttention-120            [-1, 197, 768]               0
          Dropout-121            [-1, 197, 768]               0
      ResidualAdd-122            [-1, 197, 768]               0
        LayerNorm-123            [-1, 197, 768]           1,536
```

```
            Linear-124            [-1, 197, 3072]       2,362,368
              GELU-125            [-1, 197, 3072]               0
           Dropout-126            [-1, 197, 3072]               0
            Linear-127             [-1, 197, 768]       2,360,064
           Dropout-128             [-1, 197, 768]               0
      ResidualAdd-129             [-1, 197, 768]               0
        LayerNorm-130             [-1, 197, 768]           1,536
           Linear-131            [-1, 197, 2304]       1,771,776
           Dropout-132         [-1, 8, 197, 197]               0
           Linear-133             [-1, 197, 768]         590,592
  MultiHeadAttention-134              [-1, 197, 768]               0
           Dropout-135             [-1, 197, 768]               0
      ResidualAdd-136             [-1, 197, 768]               0
        LayerNorm-137             [-1, 197, 768]           1,536
           Linear-138            [-1, 197, 3072]       2,362,368
             GELU-139            [-1, 197, 3072]               0
           Dropout-140            [-1, 197, 3072]               0
           Linear-141             [-1, 197, 768]       2,360,064
           Dropout-142             [-1, 197, 768]               0
      ResidualAdd-143             [-1, 197, 768]               0
        LayerNorm-144             [-1, 197, 768]           1,536
           Linear-145            [-1, 197, 2304]       1,771,776
           Dropout-146         [-1, 8, 197, 197]               0
           Linear-147             [-1, 197, 768]         590,592
  MultiHeadAttention-148              [-1, 197, 768]               0
           Dropout-149             [-1, 197, 768]               0
      ResidualAdd-150             [-1, 197, 768]               0
        LayerNorm-151             [-1, 197, 768]           1,536
           Linear-152            [-1, 197, 3072]       2,362,368
             GELU-153            [-1, 197, 3072]               0
           Dropout-154            [-1, 197, 3072]               0
           Linear-155             [-1, 197, 768]       2,360,064
           Dropout-156             [-1, 197, 768]               0
      ResidualAdd-157             [-1, 197, 768]               0
        LayerNorm-158             [-1, 197, 768]           1,536
           Linear-159            [-1, 197, 2304]       1,771,776
           Dropout-160         [-1, 8, 197, 197]               0
           Linear-161             [-1, 197, 768]         590,592
  MultiHeadAttention-162              [-1, 197, 768]               0
           Dropout-163             [-1, 197, 768]               0
      ResidualAdd-164             [-1, 197, 768]               0
        LayerNorm-165             [-1, 197, 768]           1,536
           Linear-166            [-1, 197, 3072]       2,362,368
             GELU-167            [-1, 197, 3072]               0
           Dropout-168            [-1, 197, 3072]               0
           Linear-169             [-1, 197, 768]       2,360,064
           Dropout-170             [-1, 197, 768]               0
      ResidualAdd-171             [-1, 197, 768]               0
           Reduce-172                  [-1, 768]               0
        LayerNorm-173                  [-1, 768]           1,536
           Linear-174                 [-1, 1000]         769,000
  ----------------------------------------------------------------
  Total params: 86,415,592
```

```
Trainable params: 86,415,592
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 364.33
Params size (MB): 329.65
Estimated Total Size (MB): 694.56
----------------------------------------------------------------
```