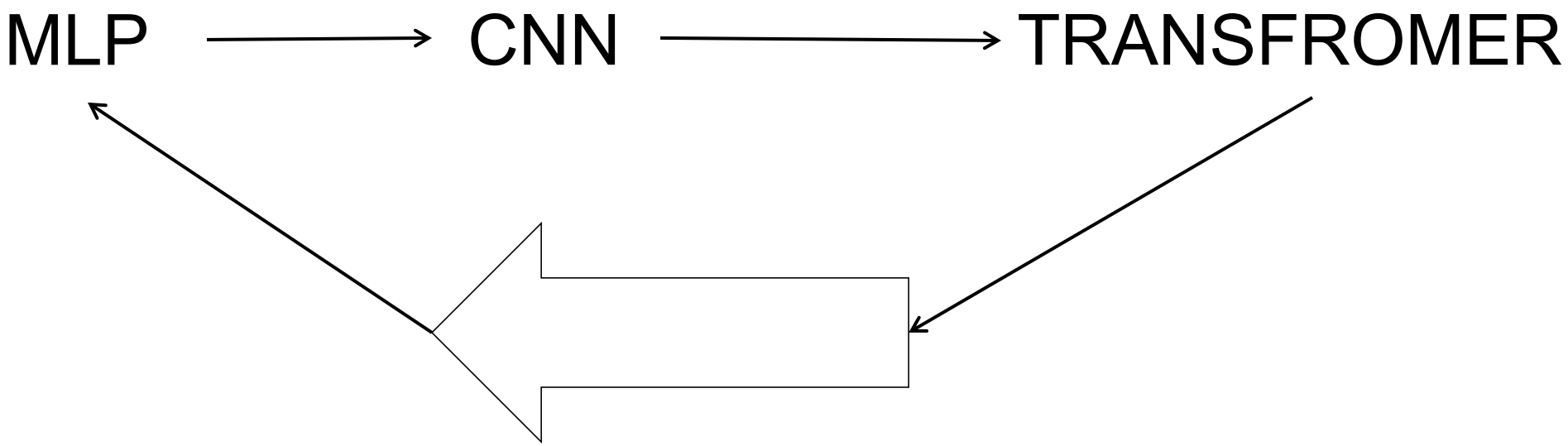
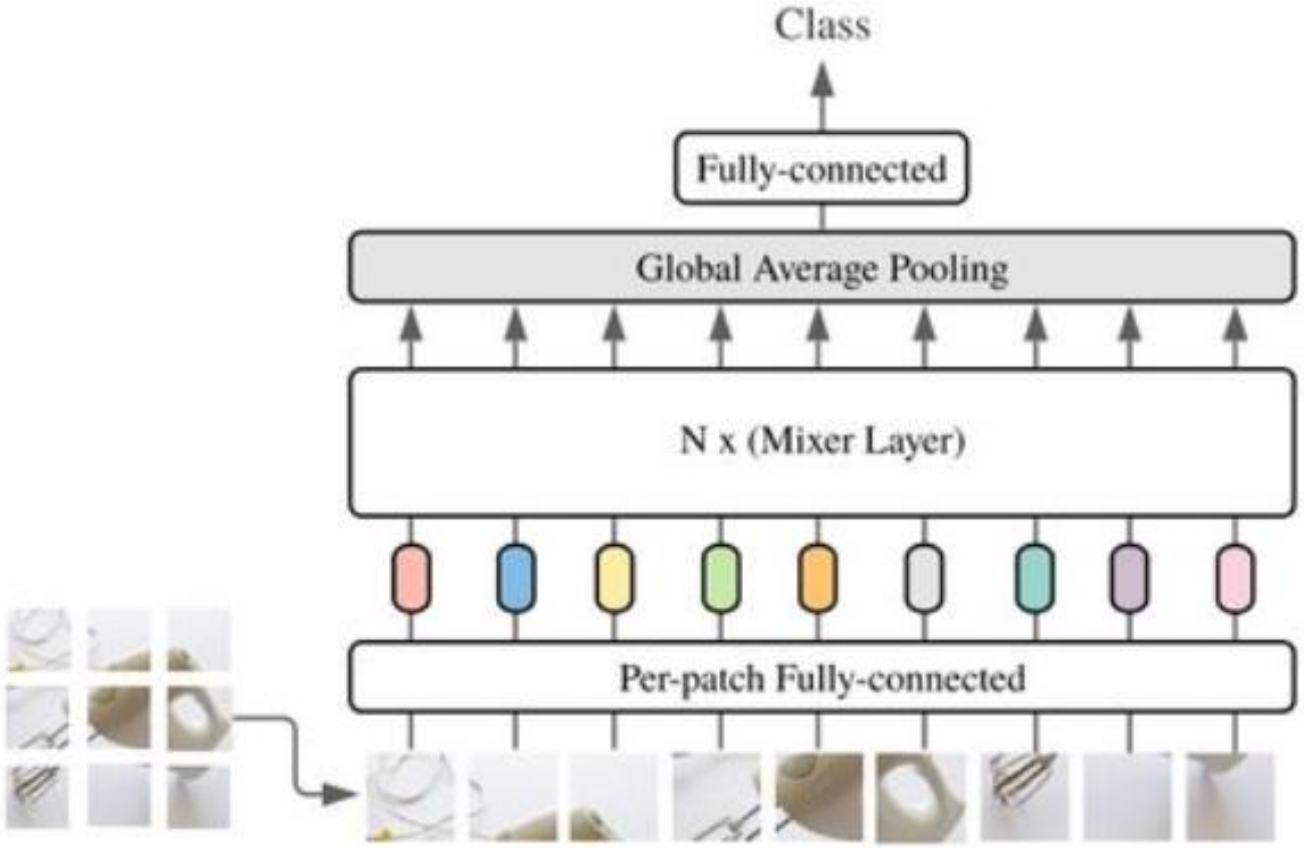
MLP-Mixer：An all-MLP Architecture for Vision

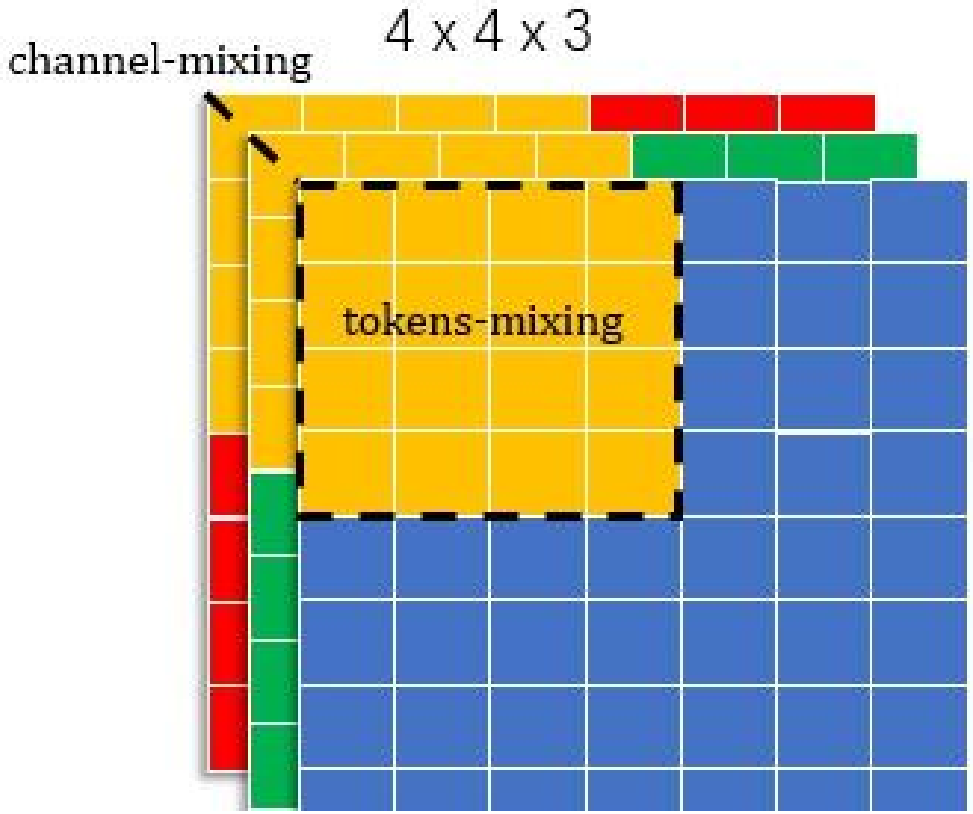
MLP-Mixer：An all-MLP Architecture for Vision



MLP-Mixer：An all-MLP Architecture for Vision

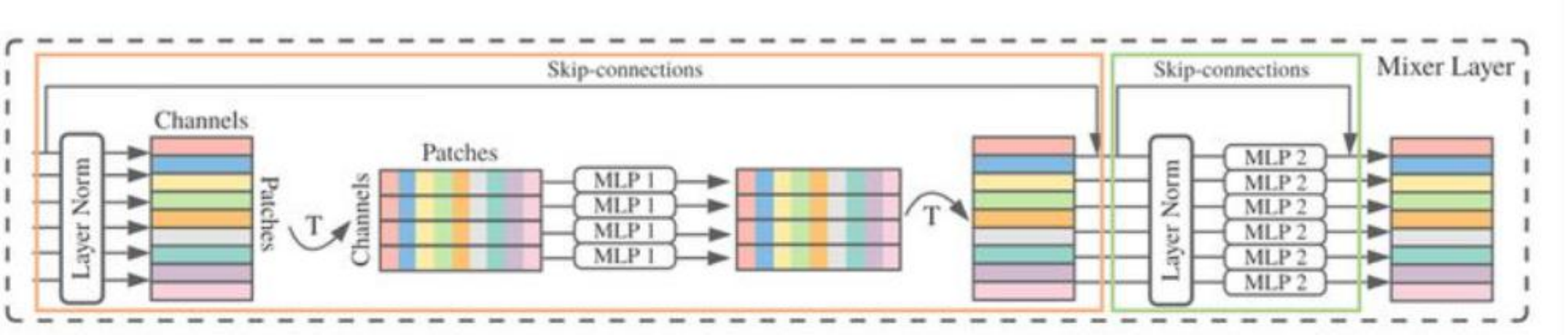


MLP-Mixer：An all-MLP Architecture for Vision



MLP-Mixer：An all-MLP Architecture for Vision

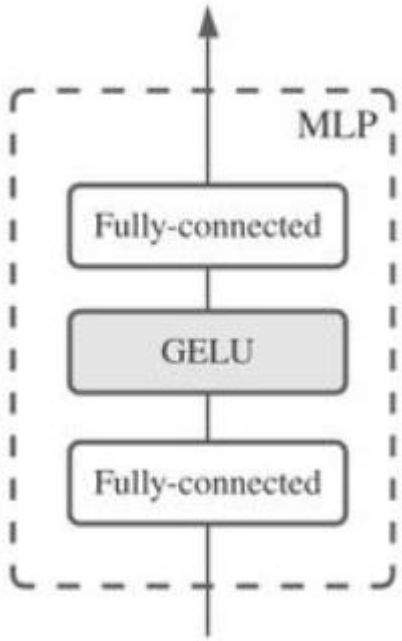
Mixer Architecture



token-mixing

channel-mixing MLP

MLP-Mixer：An all-MLP Architecture for Vision



$$U_{*,i} = X_{*,i} + W_2\sigma(W_1 LayerNorm(X)_{*,i}),\ \text{for } i = 1, \cdots, C$$
$$Y_{j,*} = U_{j,*} + W_4\sigma(W_3 LayerNorm(U)_{j,*}),\ \text{for } j = 1, \cdots, S$$

逐元非线性激活函数（GELU）

MLP-Mixer：An all-MLP Architecture for Vision

$$E[mx] = xE[m]$$

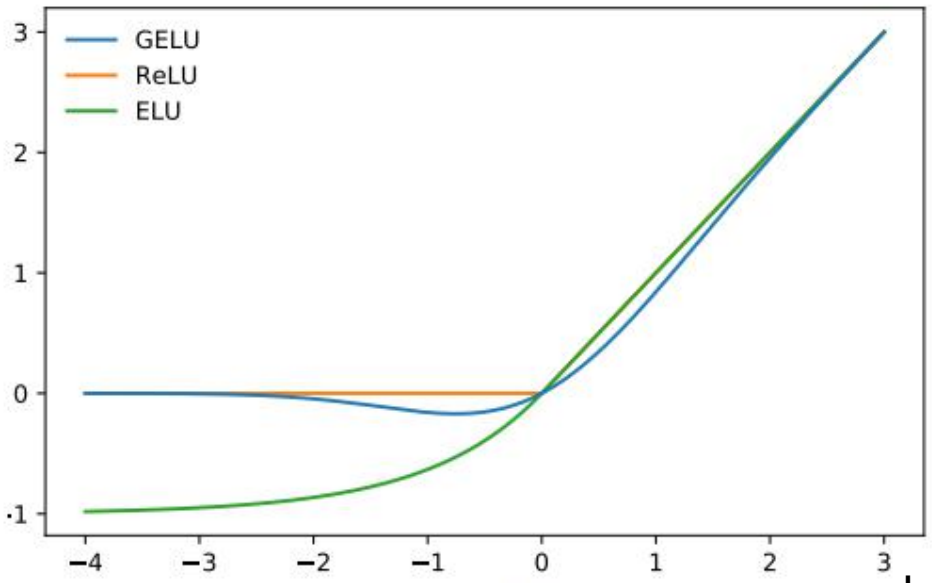
Since m is a Bernoulli Random Variable, its expected value is $\Phi(x)$

$$\implies E[mx] = x\Phi(x)$$

由于Φ（x）是高斯分布的累积分布，并且通常使用误差函数进行计算，因此我们将高斯误差线性单位（GELU）定义为：

$$\mathrm{GELU}(x) = xP(X \leq x) = x\Phi(x)$$

$$\approx 0.5x\left(1 + \tanh\left[\sqrt{2/\pi}\left(x + 0.044715x^3\right)\right]\right)$$



GELU（μ=0，σ=1），ReLU和ELU（α=1）

https://arxiv.org/abs/1606.08415

MLP-Mixer：An all-MLP Architecture for Vision

```python
def MLPMixer(*, image_size, patch_size, dim, depth, num_classes, expansion_factor = 4, dropout = 0.):
    assert (image_size % patch_size) == 0, 'image must be divisible by patch size'
    num_patches = (image_size // patch_size) ** 2
    chan_first, chan_last = partial(nn.Conv1d, kernel_size = 1), nn.Linear

    return nn.Sequential(
        # 1. 将图片拆成多个patches
        Rearrange('b c (h p1) (w p2) -> b (h w) (p1 p2 c)', p1 = patch_size, p2 = patch_size),
        # 2. 用一个全连接网络对所有patch进行处理，提取出tokens
        nn.Linear((patch_size ** 2) * 3, dim),
        # 3. 经过N个Mixer层，混合提炼特征信息
        *[nn.Sequential(
            PreNormResidual(dim, FeedForward(num_patches, expansion_factor, dropout, chan_first)),
            PreNormResidual(dim, FeedForward(dim, expansion_factor, dropout, chan_last))
        ) for _ in range(depth)],
        nn.LayerNorm(dim),
        Reduce('b n c -> b c', 'mean'),
        # 4. 最后一个全连接层进行类别预测
        nn.Linear(dim, num_classes)
    )
```

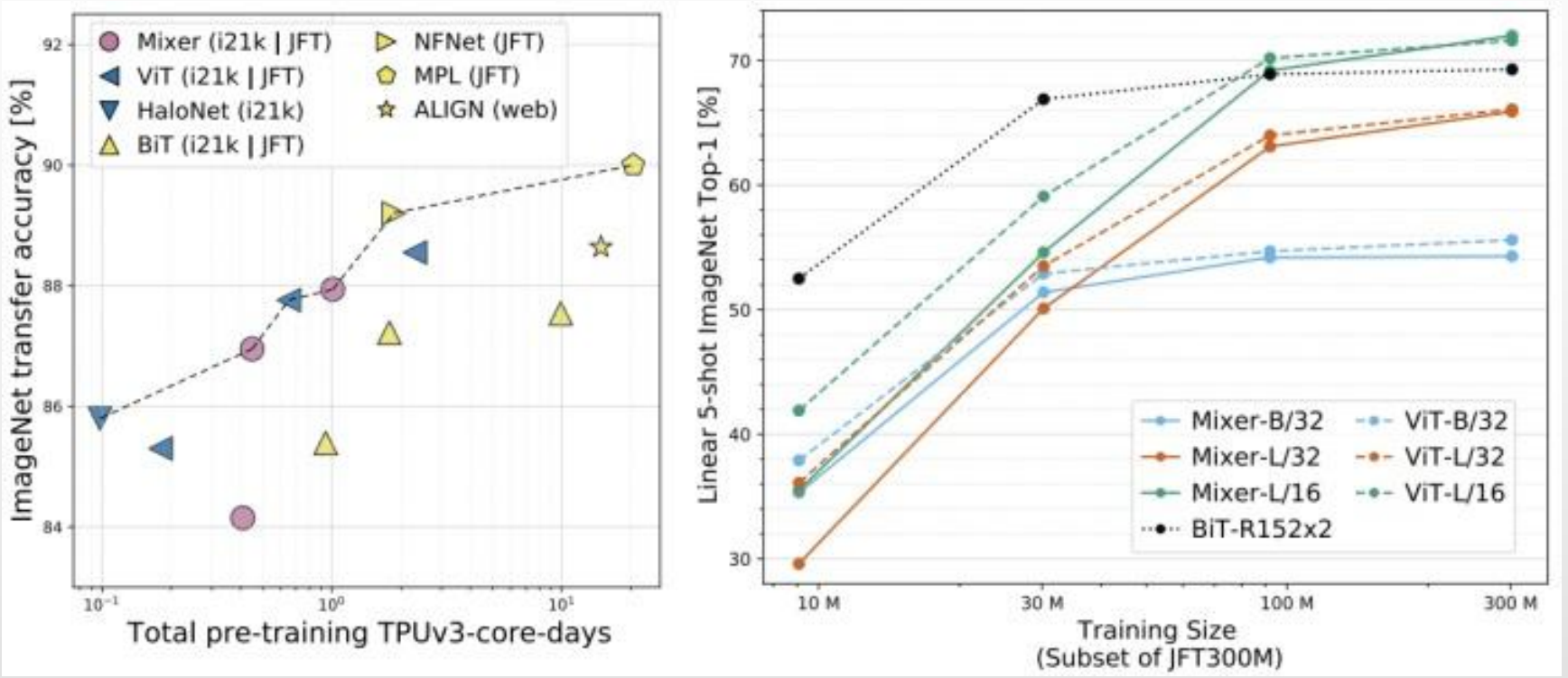
MLP-Mixer：An all-MLP Architecture for Vision

```python
class PreNormResidual(nn.Module):
    def __init__(self, dim, fn):
        super().__init__()
        self.fn = fn
        self.norm = nn.LayerNorm(dim)

    def forward(self, x):
        return self.fn(self.norm(x)) + x

def FeedForward(dim, expansion_factor = 4, dropout = 0., dense = nn.Linear):
    return nn.Sequential(
        dense(dim, dim * expansion_factor),
        nn.GELU(),
        nn.Dropout(dropout),
        dense(dim * expansion_factor, dim),
        nn.Dropout(dropout)
    )
```

MLP-Mixer：An all-MLP Architecture for Vision

	ImNet top-1	ReaL top-1	Avg 5 top-1	VTAB-1k 19 tasks	Throughput img/sec/core	TPUv3 core-days
Pre-trained on ImageNet-21k (public)						
● HaloNet [49]	85.8	—	—	—	120	0.10k
● Mixer-L/16	84.15	87.86	93.91	74.95	105	0.41k
● ViT-L/16 [14]	85.30	88.62	94.39	72.72	32	0.18k
● BiT-R152x4 [22]	85.39	—	94.04	70.64	26	0.94k
Pre-trained on JFT-300M (proprietary)						
● NFNet-F4+ [7]	89.2	—	—	—	46	1.86k
● Mixer-H/14	87.94	90.18	95.71	75.33	40	1.01k
● BiT-R152x4 [22]	87.54	90.54	95.33	76.29	26	9.90k
● ViT-H/14 [14]	88.55	90.72	95.97	77.63	15	2.30k
Pre-trained on unlabelled or weakly labelled data (proprietary)						
● MPL [33]	90.0	91.12	—	—	—	20.48k
● ALIGN [21]	88.64	—	—	79.99	15	14.82k

MLP-Mixer：An all-MLP Architecture for Vision



谢谢大家