MLP-Mixer
An all-MLP Architecture for Vision

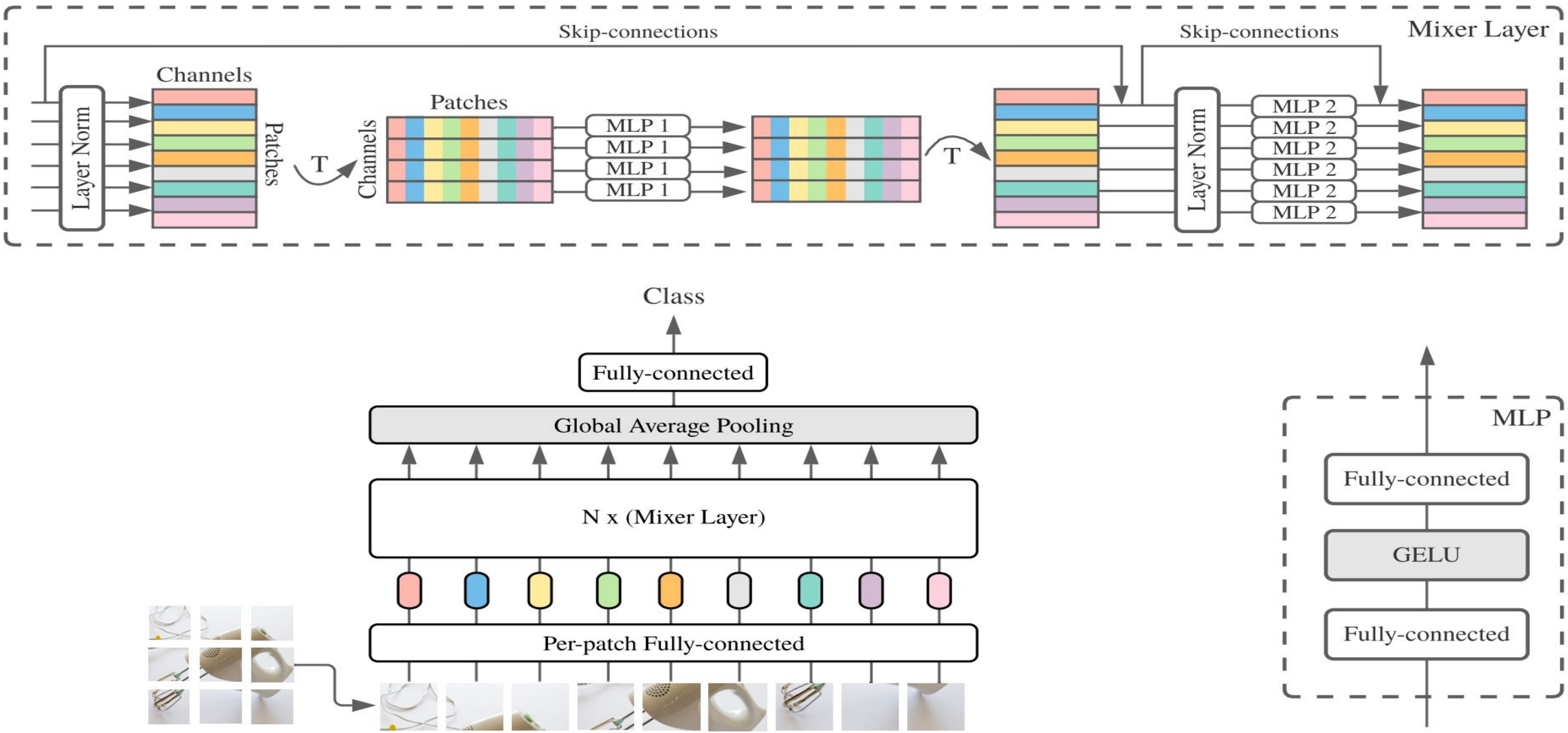
Sebenele Thwala

IDEA

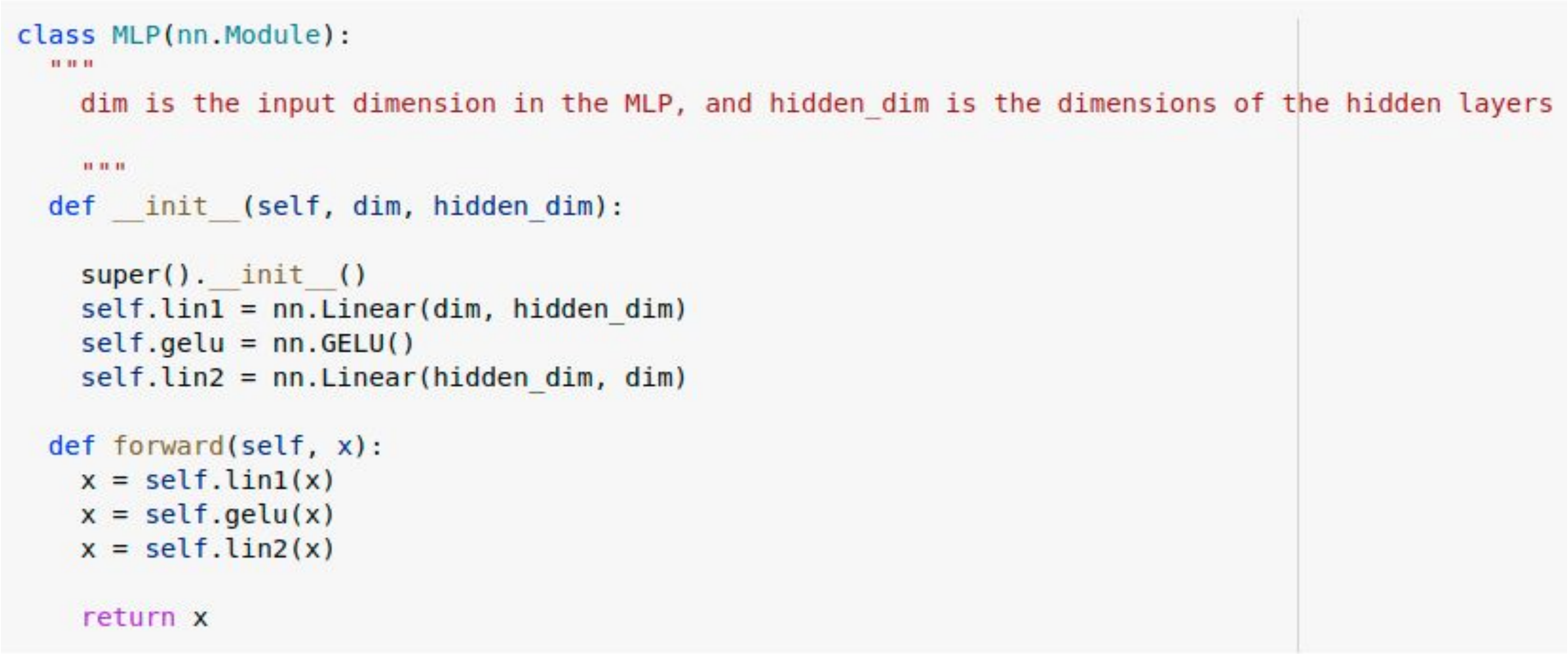
- Alternative and Simpler approach to vision networks

- Show that CNNs and Attention although sufficient for good performance, are not necessary

ARCHITECTURE

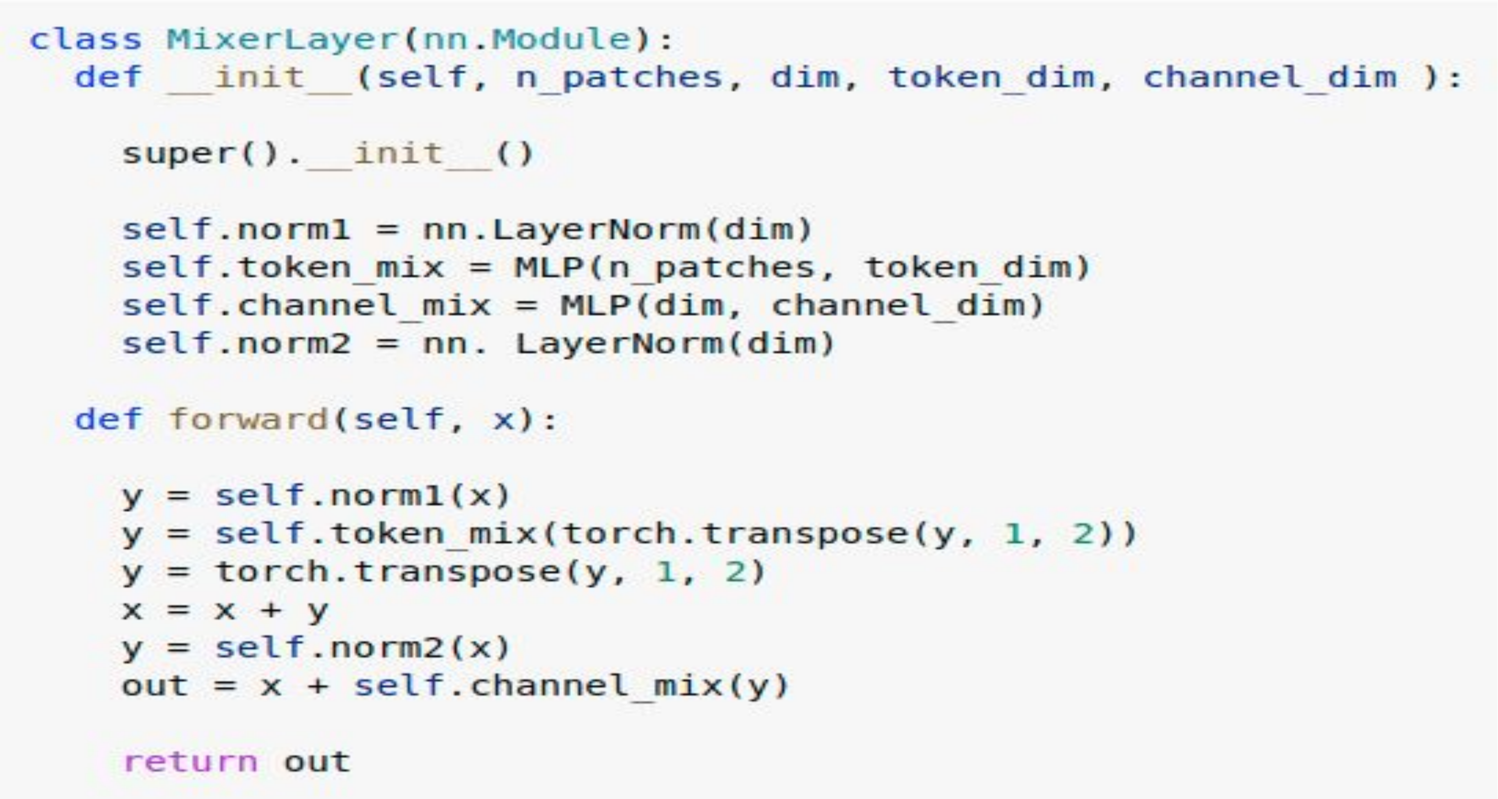


MLP BLOCK

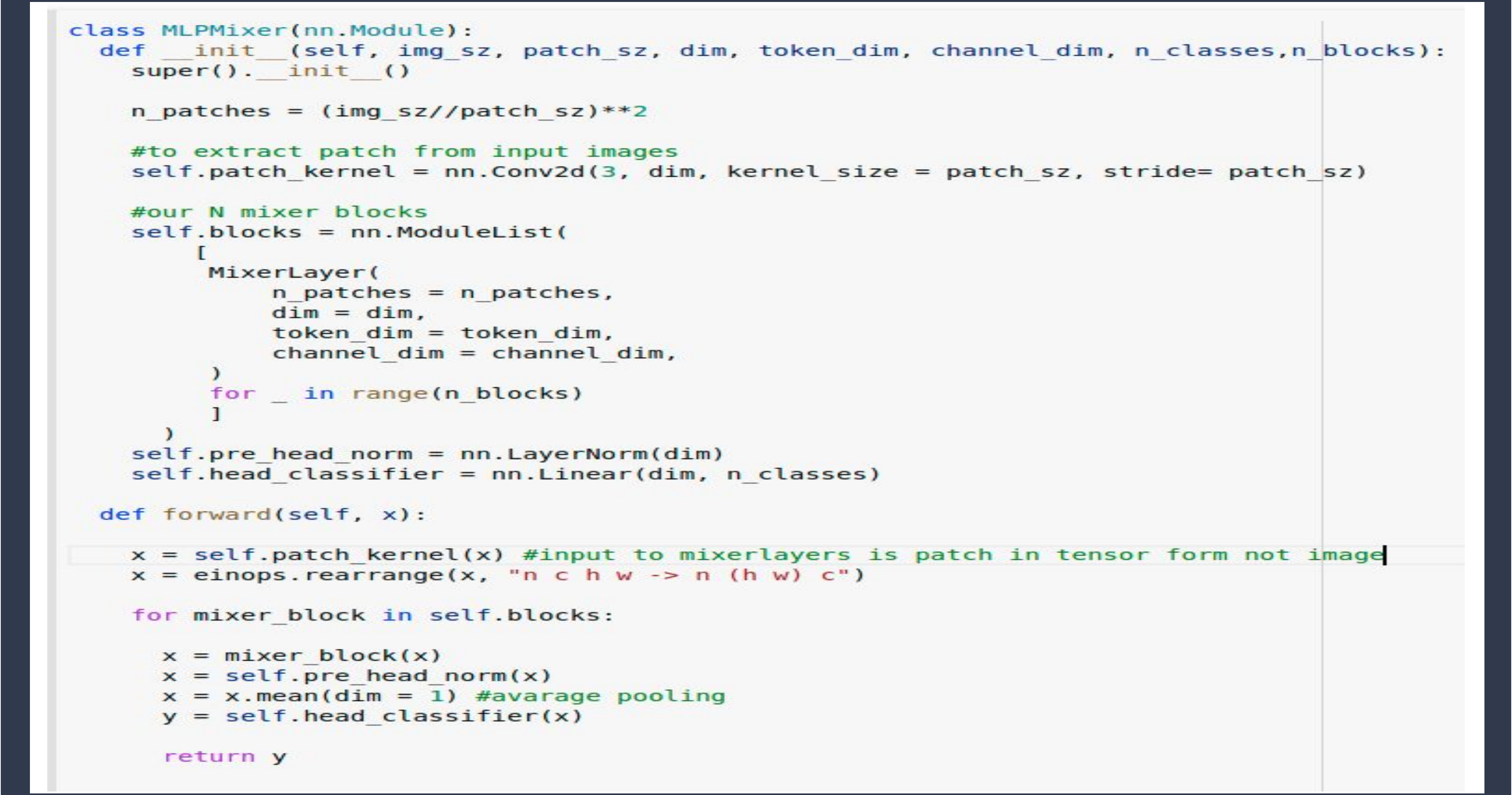


```python
class MLP(nn.Module):
  """
    dim is the input dimension in the MLP, and hidden_dim is the dimensions of the hidden layers

    """
  def __init__(self, dim, hidden_dim):

    super().__init__()
    self.lin1 = nn.Linear(dim, hidden_dim)
    self.gelu = nn.GELU()
    self.lin2 = nn.Linear(hidden_dim, dim)

  def forward(self, x):
    x = self.lin1(x)
    x = self.gelu(x)
    x = self.lin2(x)

    return x
```

Mixer Layer

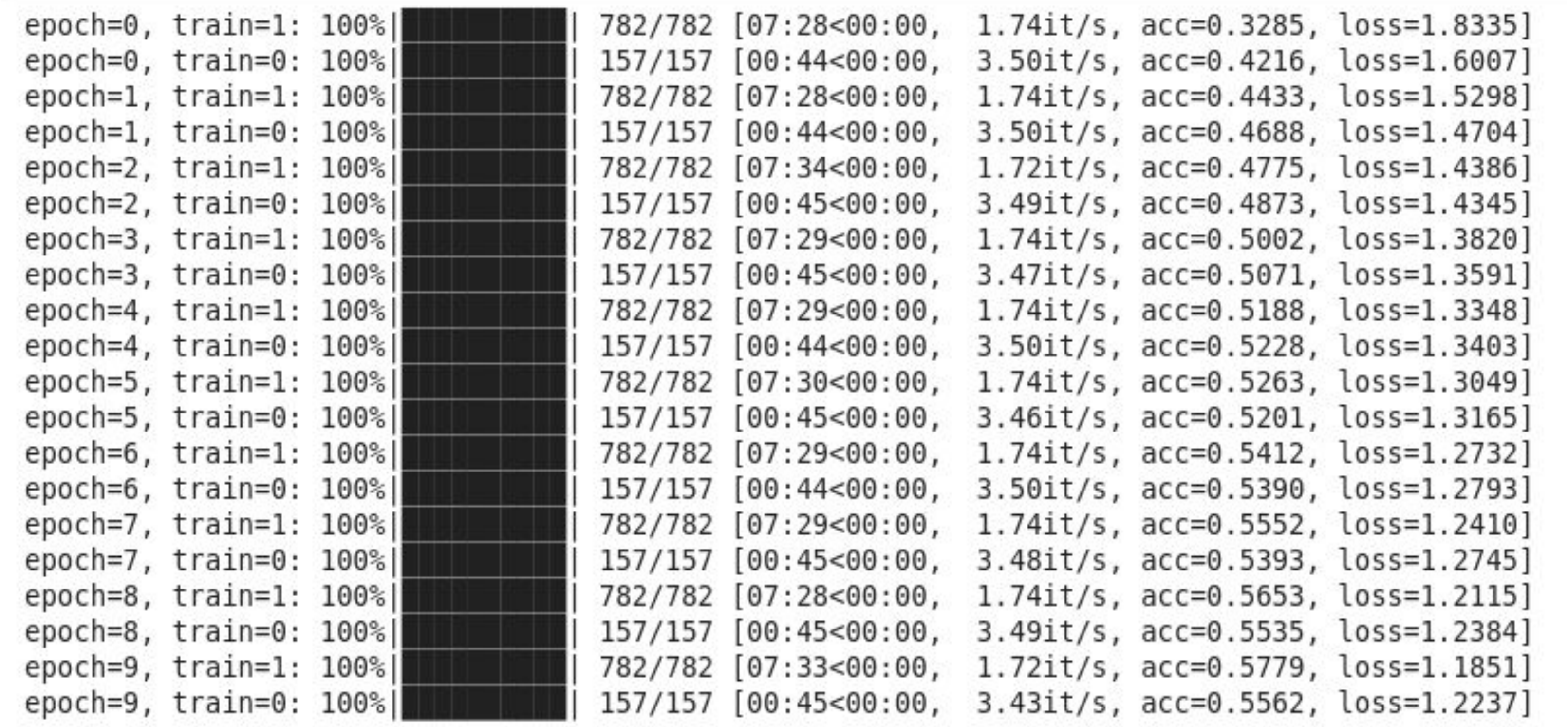


```python
class MixerLayer(nn.Module):
  def __init__(self, n_patches, dim, token_dim, channel_dim ):

    super().__init__()

    self.norm1 = nn.LayerNorm(dim)
    self.token_mix = MLP(n_patches, token_dim)
    self.channel_mix = MLP(dim, channel_dim)
    self.norm2 = nn. LayerNorm(dim)

  def forward(self, x):

    y = self.norm1(x)
    y = self.token_mix(torch.transpose(y, 1, 2))
    y = torch.transpose(y, 1, 2)
    x = x + y
    y = self.norm2(x)
    out = x + self.channel_mix(y)

    return out
```



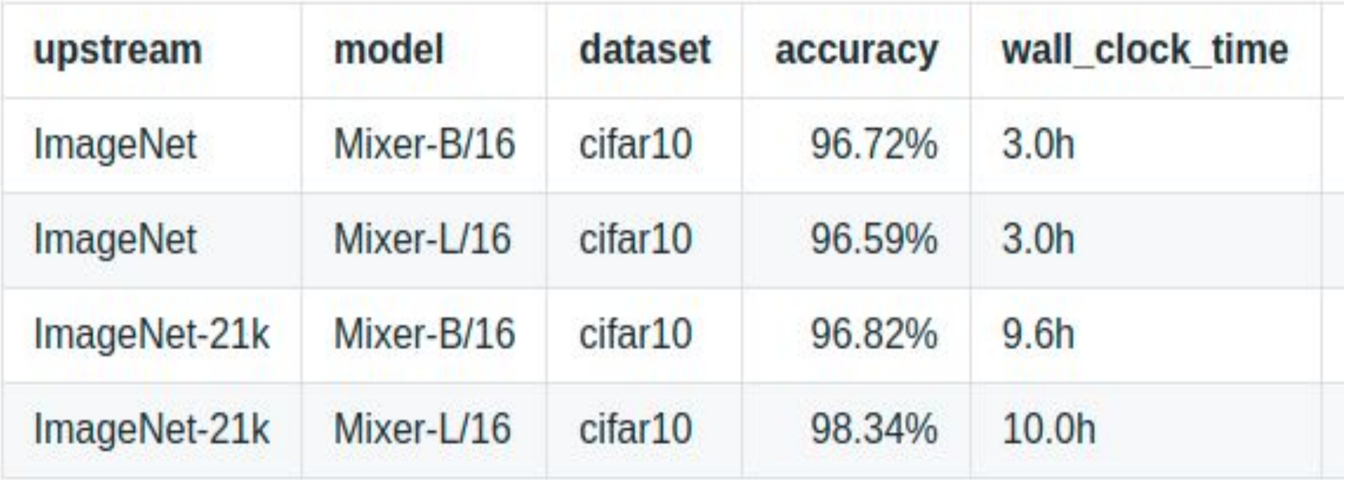
```
class MLPMixer(nn.Module):
  def __init__(self, img_sz, patch_sz, dim, token_dim, channel_dim, n_classes,n_blocks):
    super().__init__()

    n_patches = (img_sz//patch_sz)**2

    #to extract patch from input images
    self.patch_kernel = nn.Conv2d(3, dim, kernel_size = patch_sz, stride= patch_sz)

    #our N mixer blocks
    self.blocks = nn.ModuleList(
        [
         MixerLayer(
             n_patches = n_patches,
             dim = dim,
             token_dim = token_dim,
             channel_dim = channel_dim,
         )
         for _ in range(n_blocks)
         ]
      )
    self.pre_head_norm = nn.LayerNorm(dim)
    self.head_classifier = nn.Linear(dim, n_classes)

  def forward(self, x):

    x = self.patch_kernel(x) #input to mixerlayers is patch in tensor form not image
    x = einops.rearrange(x, "n c h w -> n (h w) c")

    for mixer_block in self.blocks:

      x = mixer_block(x)
      x = self.pre_head_norm(x)
      x = x.mean(dim = 1) #avarage pooling
      y = self.head_classifier(x)

      return y
```

EXPERIMENTS



```
epoch=0, train=1: 100%|██████████| 782/782 [07:28<00:00,  1.74it/s, acc=0.3285, loss=1.8335]
epoch=0, train=0: 100%|██████████| 157/157 [00:44<00:00,  3.50it/s, acc=0.4216, loss=1.6007]
epoch=1, train=1: 100%|██████████| 782/782 [07:28<00:00,  1.74it/s, acc=0.4433, loss=1.5298]
epoch=1, train=0: 100%|██████████| 157/157 [00:44<00:00,  3.50it/s, acc=0.4688, loss=1.4704]
epoch=2, train=1: 100%|██████████| 782/782 [07:34<00:00,  1.72it/s, acc=0.4775, loss=1.4386]
epoch=2, train=0: 100%|██████████| 157/157 [00:45<00:00,  3.49it/s, acc=0.4873, loss=1.4345]
epoch=3, train=1: 100%|██████████| 782/782 [07:29<00:00,  1.74it/s, acc=0.5002, loss=1.3820]
epoch=3, train=0: 100%|██████████| 157/157 [00:45<00:00,  3.47it/s, acc=0.5071, loss=1.3591]
epoch=4, train=1: 100%|██████████| 782/782 [07:29<00:00,  1.74it/s, acc=0.5188, loss=1.3348]
epoch=4, train=0: 100%|██████████| 157/157 [00:44<00:00,  3.50it/s, acc=0.5228, loss=1.3403]
epoch=5, train=1: 100%|██████████| 782/782 [07:30<00:00,  1.74it/s, acc=0.5263, loss=1.3049]
epoch=5, train=0: 100%|██████████| 157/157 [00:45<00:00,  3.46it/s, acc=0.5201, loss=1.3165]
epoch=6, train=1: 100%|██████████| 782/782 [07:29<00:00,  1.74it/s, acc=0.5412, loss=1.2732]
epoch=6, train=0: 100%|██████████| 157/157 [00:44<00:00,  3.50it/s, acc=0.5390, loss=1.2793]
epoch=7, train=1: 100%|██████████| 782/782 [07:29<00:00,  1.74it/s, acc=0.5552, loss=1.2410]
epoch=7, train=0: 100%|██████████| 157/157 [00:45<00:00,  3.48it/s, acc=0.5393, loss=1.2745]
epoch=8, train=1: 100%|██████████| 782/782 [07:28<00:00,  1.74it/s, acc=0.5653, loss=1.2115]
epoch=8, train=0: 100%|██████████| 157/157 [00:45<00:00,  3.49it/s, acc=0.5535, loss=1.2384]
epoch=9, train=1: 100%|██████████| 782/782 [07:33<00:00,  1.72it/s, acc=0.5779, loss=1.1851]
epoch=9, train=0: 100%|██████████| 157/157 [00:45<00:00,  3.43it/s, acc=0.5562, loss=1.2237]
```

Comparison



upstream	model	dataset	accuracy	wall_clock_time
ImageNet	Mixer-B/16	cifar10	96.72%	3.0h
ImageNet	Mixer-L/16	cifar10	96.59%	3.0h
ImageNet-21k	Mixer-B/16	cifar10	96.82%	9.6h
ImageNet-21k	Mixer-L/16	cifar10	98.34%	10.0h

CONCLUSION