Computer Vision

MLP Mixer: An all-MLP Architecture for Vision

Мошков Иван, БПМИ-191

Цели на сегодня

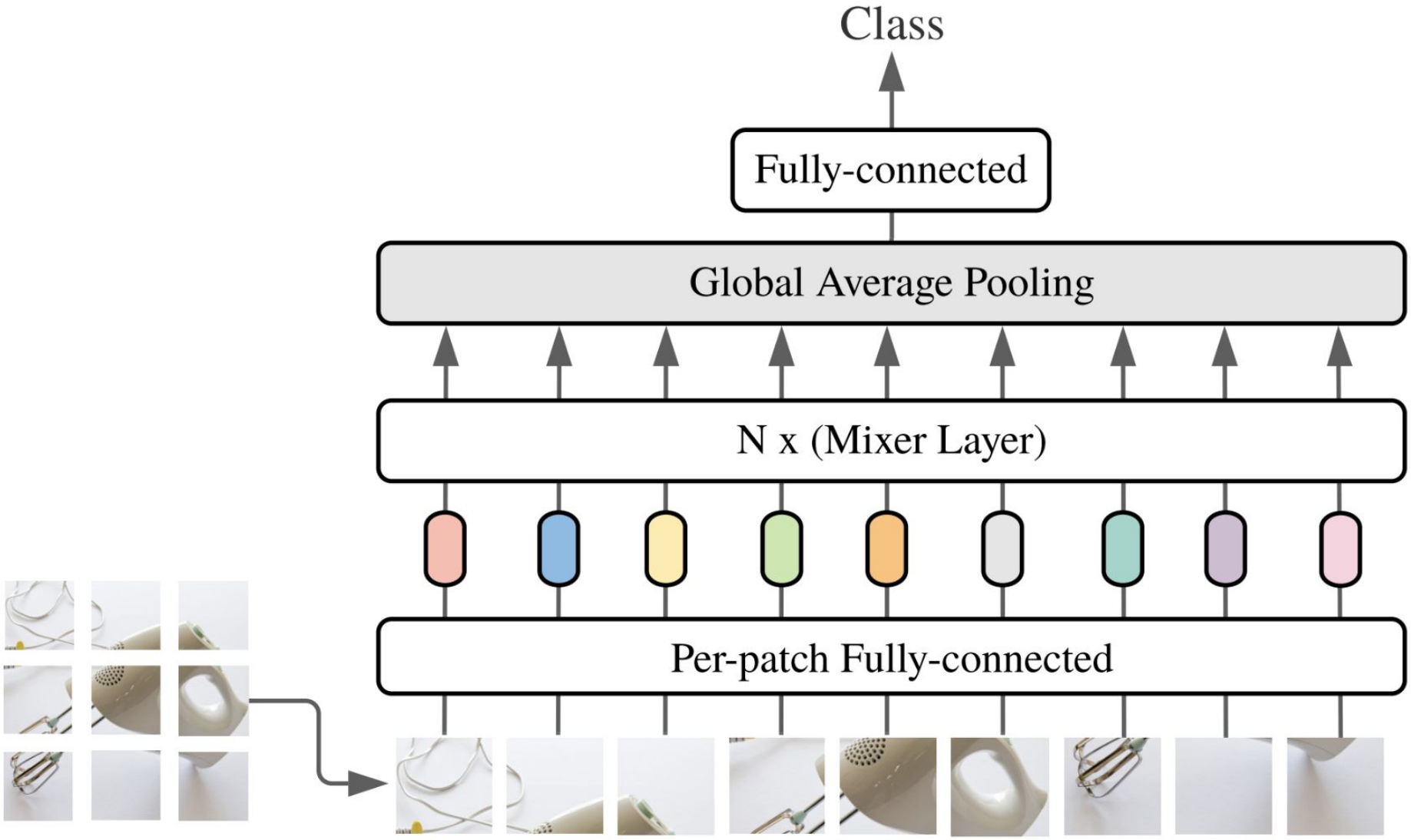
- Понять что делает

- Понять с кем он это делает
- Понять насколько хорошо он это делает

Как это было

- Странные эвристики
- Попытки прикрутить MLP
- CNN
- Трансформеры
- Прикрутили MLP

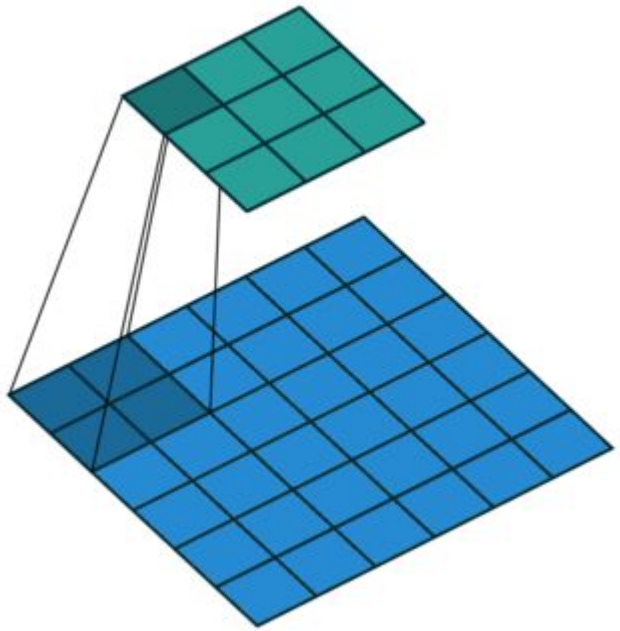
MLP? Картинки? Да!



Разбираемся 1

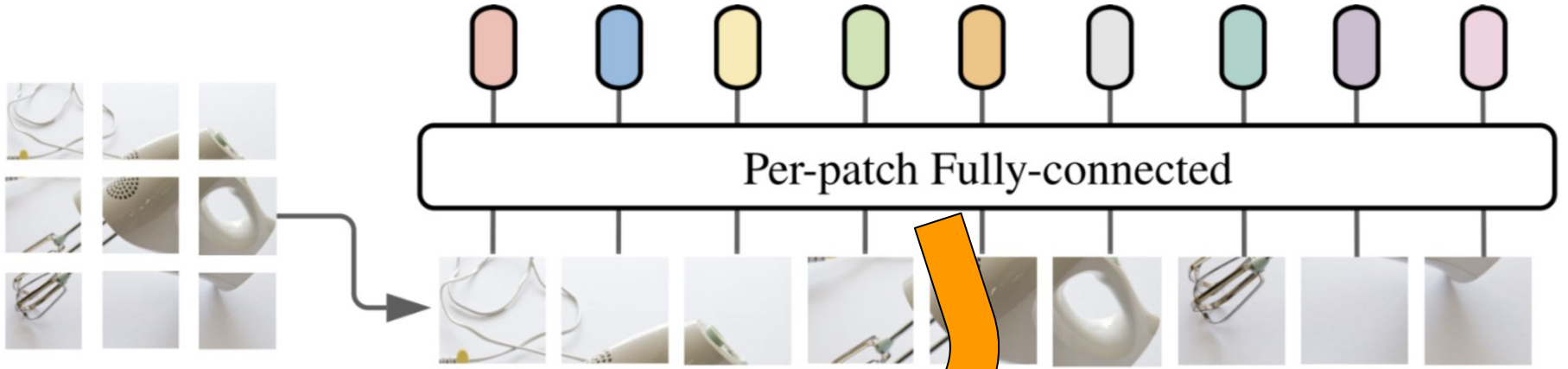
```
x = nn.Conv(
    out_channels=hidden_dim,
    conv_size=(s, s), stride=(s, s)
)
```

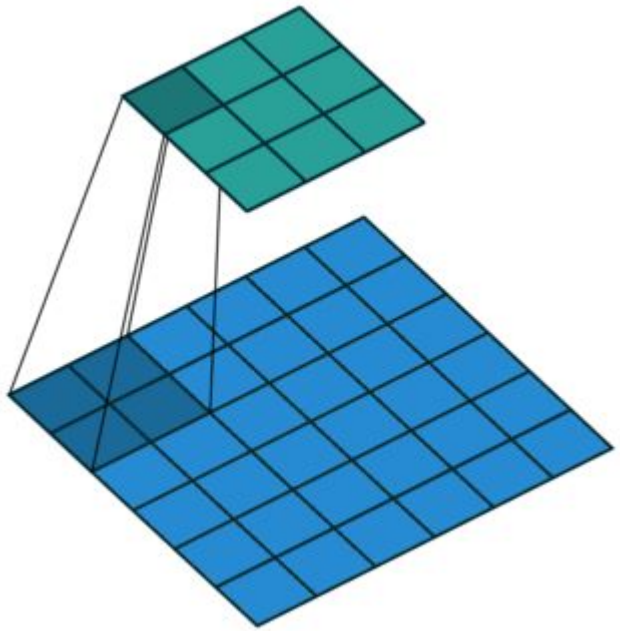
Разбираемся 2



```
x = nn.Conv(
    out_channels=1,
    conv_size=(2, 2), stride=(2, 2)
)
```

Разбираемся 3

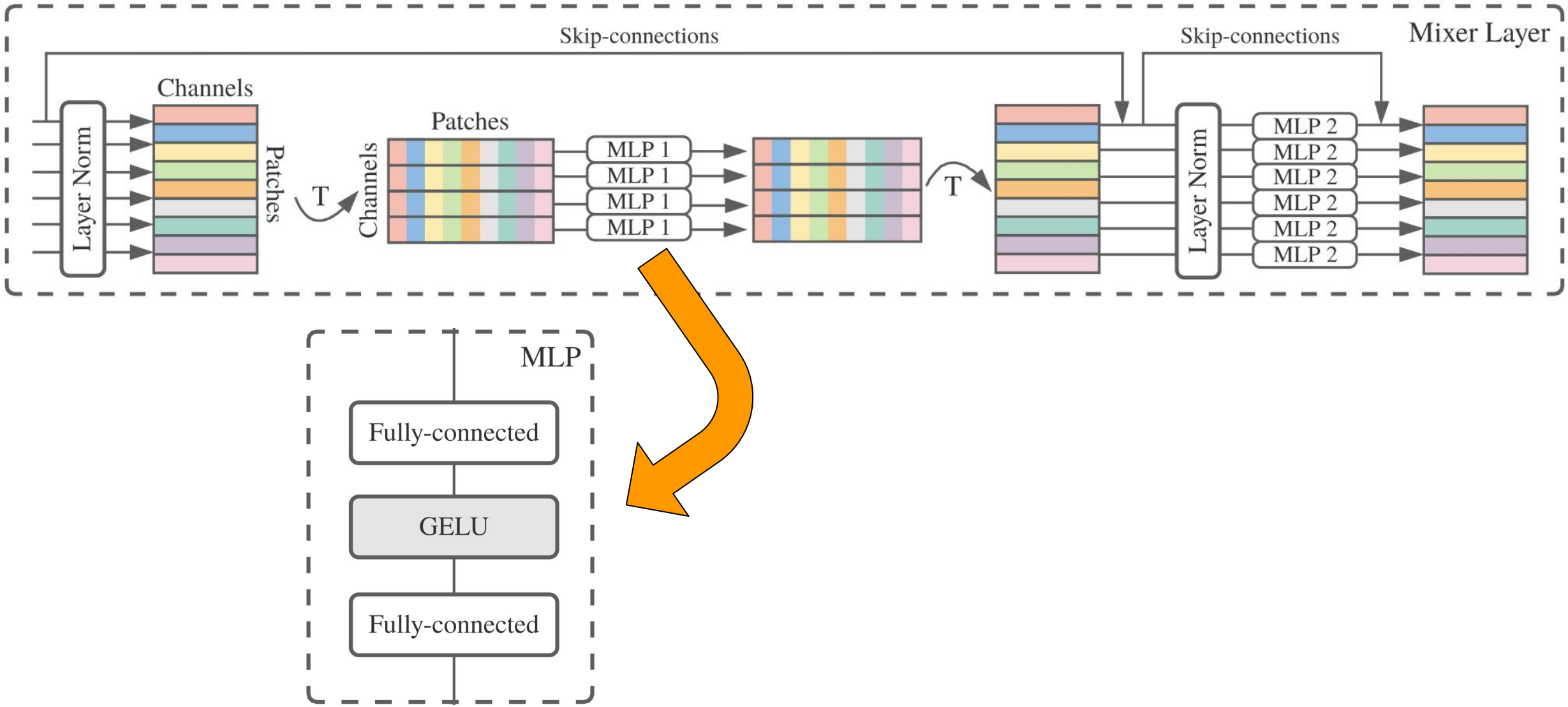




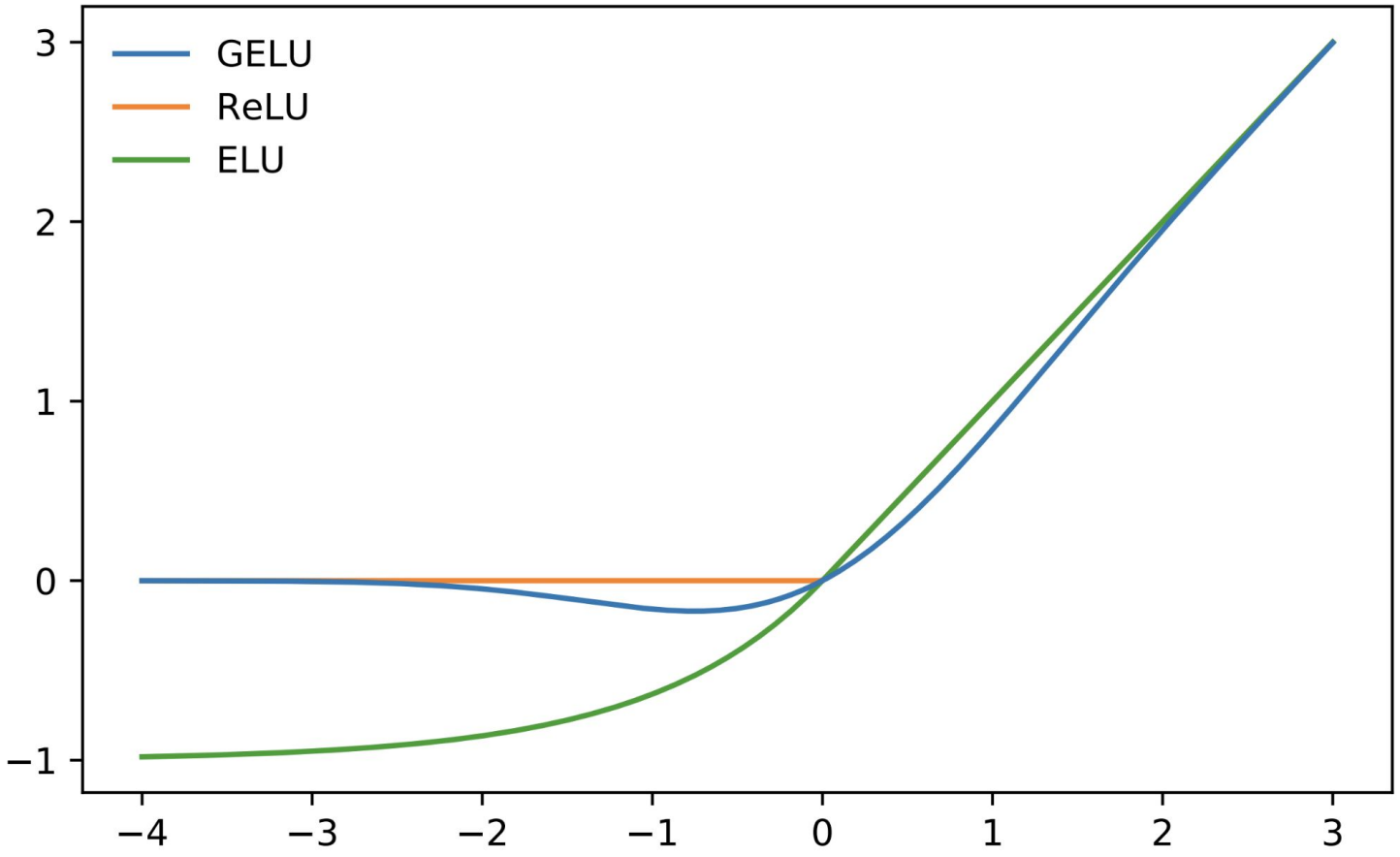
```
x = nn.Conv(out_channels=hidden_dim,
            conv_size=(s, s), stride=(s, s))(x)

x = einops.rearrange(x, “n h w c -> n (h w) c”)
```

Еще разбираемся



Держу в курсе



$$\mathrm{GELU}(x) = xP(X \leq x) = x\Phi(x) = x \cdot \frac{1}{2}\left[1 + \mathrm{erf}(x/\sqrt{2})\right]$$

Ставим класс

Pre-train

- ImageNet 21k
- JFT-300M

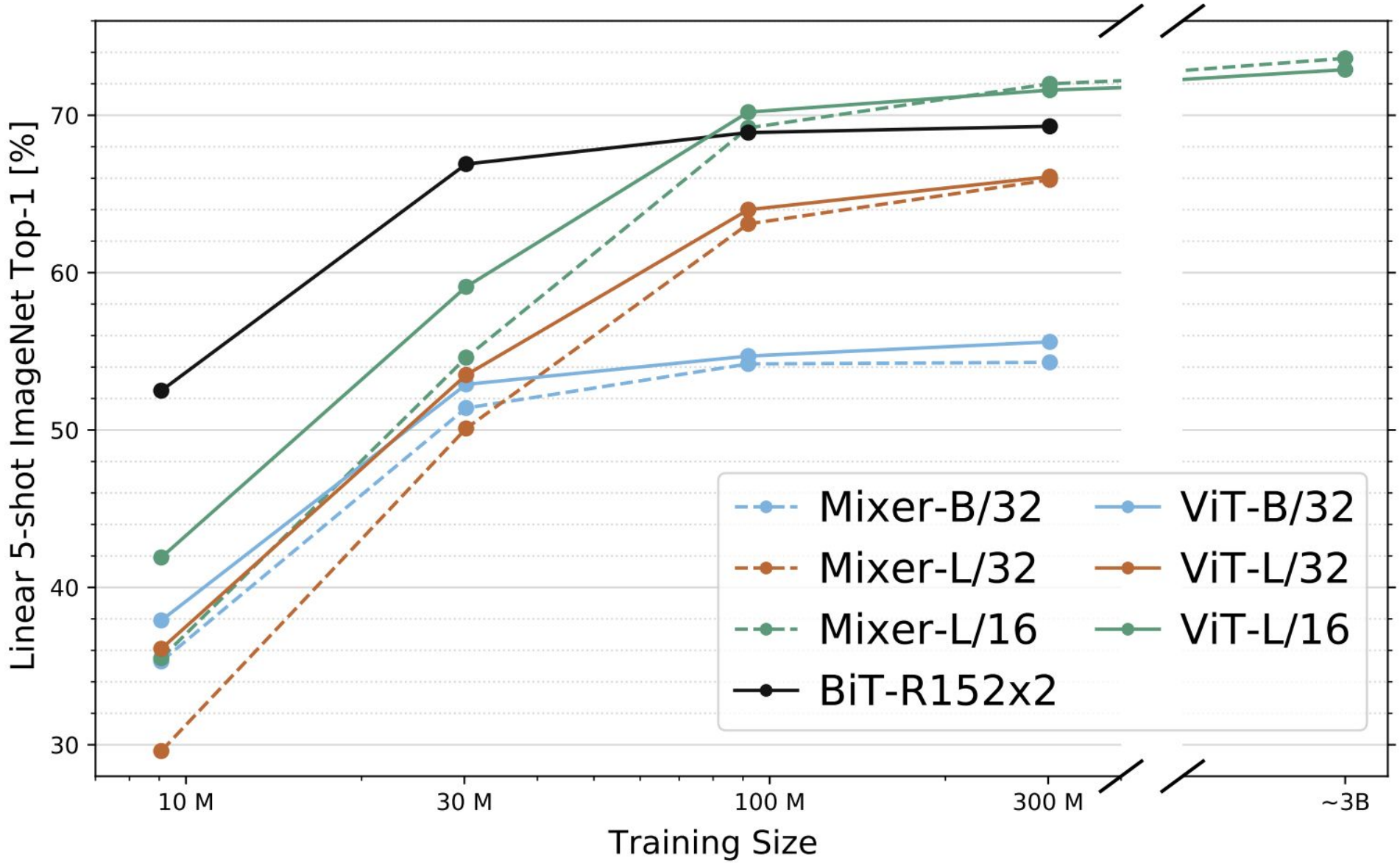
Fine-tune

- ImageNet
- Cifar
- Flowers
- Pets
- VTAB 1k

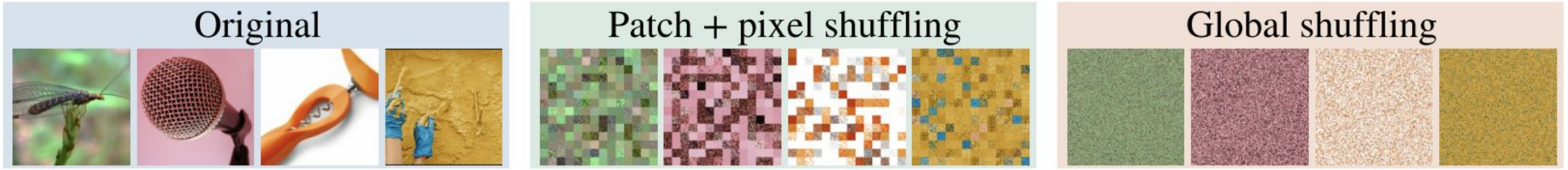
Все напрасно?

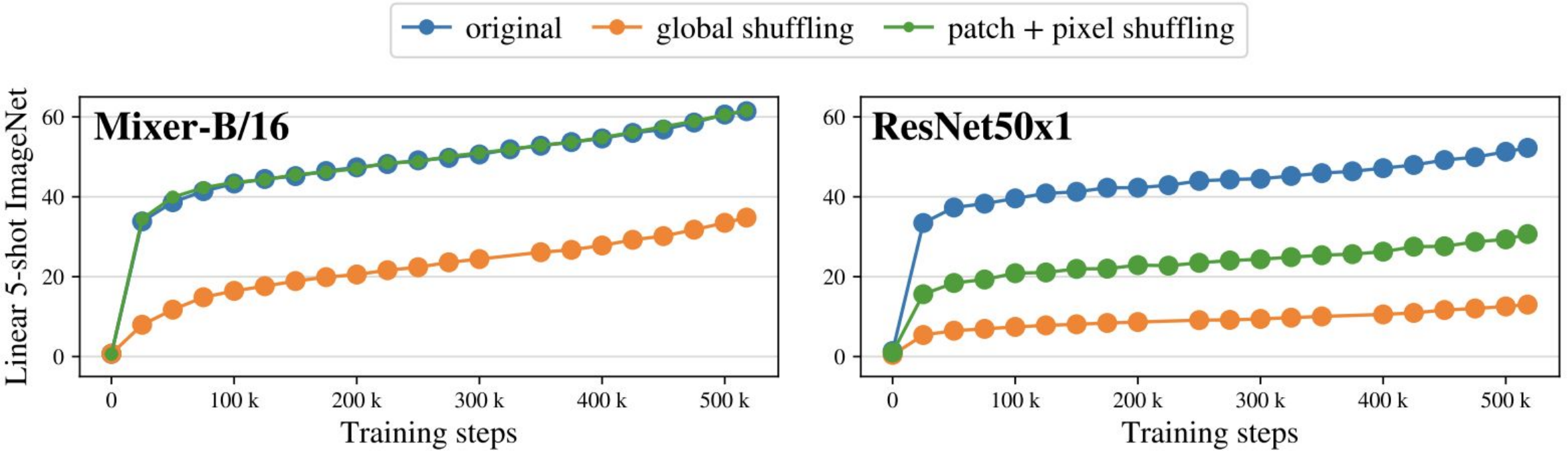
	ImNet top-1	ReaL top-1	Avg 5 top-1	VTAB-1k 19 tasks	Throughput img/sec/core	TPUv3 core-days
Pre-trained on ImageNet-21k (public)						
● HaloNet [51]	85.8	—	—	—	120	0.10k
● Mixer-L/16	84.15	87.86	93.91	74.95	105	0.41k
● ViT-L/16 [14]	85.30	88.62	94.39	72.72	32	0.18k
● BiT-R152x4 [22]	85.39	—	94.04	70.64	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 [34]	90.0	91.12	—	—	—	20.48k
● ALIGN [21]	88.64	—	—	79.99	15	14.82k

Скейлим


Linear 5-shot ImageNet Top-1 [%]
70
60
50
40
30
10 M
30 M
100 M
300 M
~3B
Training Size
Mixer-B/32
Mixer-L/32
Mixer-L/16
BiT-R152x2
ViT-B/32
ViT-L/32
ViT-L/16

Кручу верчу


Original
Patch + pixel shuffling
Global shuffling


original
global shuffling
patch + pixel shuffling
Mixer-B/16
ResNet50x1
Linear 5-shot ImageNet
Training steps
0
20
40
60
0
100 k
200 k
300 k
400 k
500 k

Для справки

Разные хаки

- GELU
- RandAugment
- MixUp
- Stochastic Depth
- Dropout
- Weight Decay

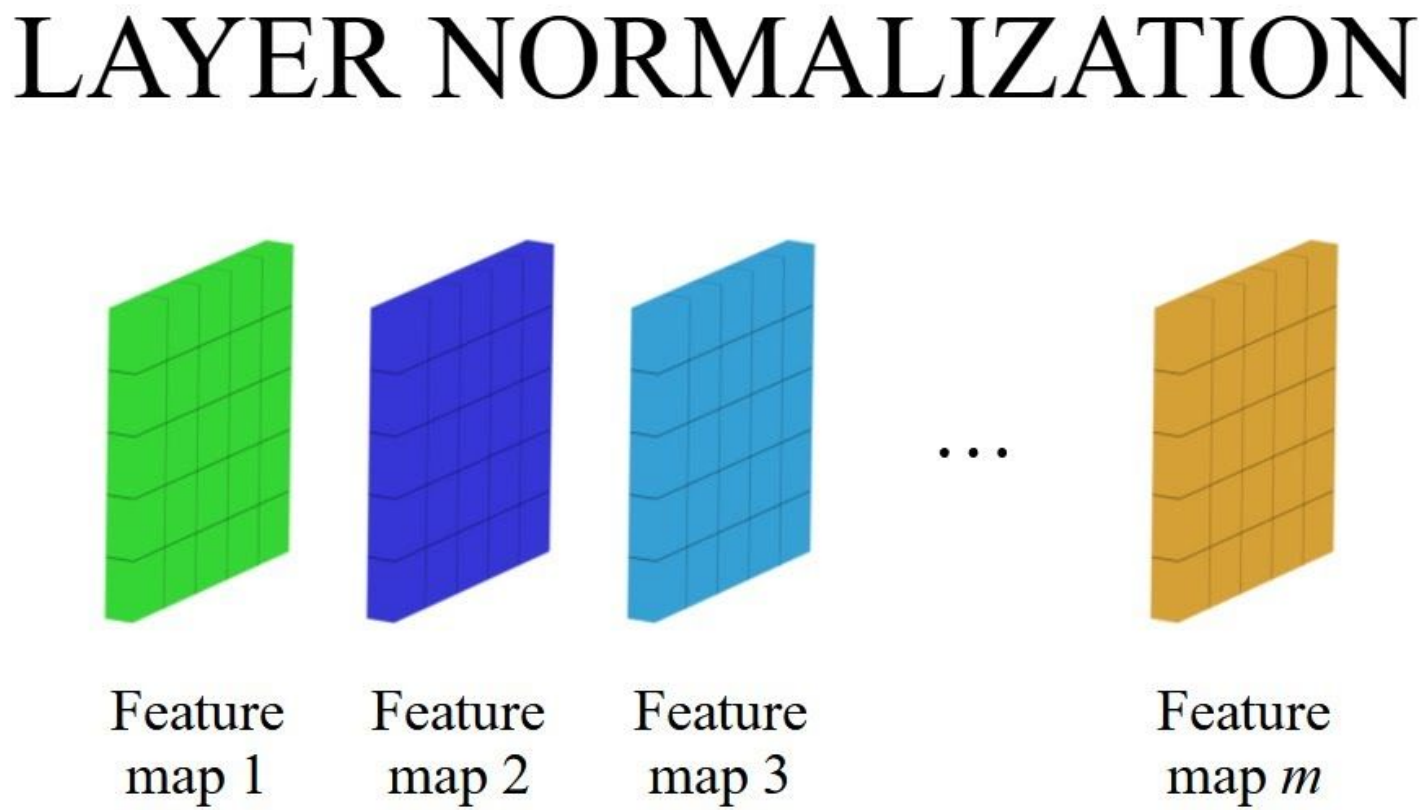
Параметры

Specification	S/32	S/16	B/32	B/16	L/32	L/16	H/14
Number of layers	8	8	12	12	24	24	32
Patch resolution $P \times P$	32×32	16×16	32×32	16×16	32×32	16×16	14×14
Hidden size C	512	512	768	768	1024	1024	1280
Sequence length S	49	196	49	196	49	196	256
MLP dimension D_C	2048	2048	3072	3072	4096	4096	5120
MLP dimension D_S	256	256	384	384	512	512	640
Parameters (M)	19	18	60	59	206	207	431

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

Layer Norm


LAYER NORMALIZATION
Feature map 1
Feature map 2
Feature map 3
...
Feature map *m*
Normalize layer output for one training sample

Код модели

```python
import einops
import flax.linen as nn
import jax.numpy as jnp

class MlpBlock(nn.Module):
  mlp_dim: int
  @nn.compact
  def __call__(self, x):
    y = nn.Dense(self.mlp_dim)(x)
    y = nn.gelu(y)
    return nn.Dense(x.shape[-1])(y)

class MixerBlock(nn.Module):
  tokens_mlp_dim: int
  channels_mlp_dim: int
  @nn.compact
  def __call__(self, x):
    y = nn.LayerNorm()(x)
    y = jnp.swapaxes(y, 1, 2)
    y = MlpBlock(self.tokens_mlp_dim, name='token_mixing')(y)
    y = jnp.swapaxes(y, 1, 2)
    x = x+y
    y = nn.LayerNorm()(x)
    return x+MlpBlock(self.channels_mlp_dim, name='channel_mixing')(y)

class MlpMixer(nn.Module):
  num_classes: int
  num_blocks: int
  patch_size: int
  hidden_dim: int
  tokens_mlp_dim: int
  channels_mlp_dim: int
  @nn.compact
  def __call__(self, x):
    s = self.patch_size
    x = nn.Conv(self.hidden_dim, (s,s), strides=(s,s), name='stem')(x)
    x = einops.rearrange(x, 'n h w c -> n (h w) c')
    for _ in range(self.num_blocks):
      x = MixerBlock(self.tokens_mlp_dim, self.channels_mlp_dim)(x)
    x = nn.LayerNorm(name='pre_head_layer_norm')(x)
    x = jnp.mean(x, axis=1)
    return nn.Dense(self.num_classes, name='head',
                    kernel_init=nn.initializers.zeros)(x)
```

Listing 1: MLP-Mixer code written in JAX/Flax.

Ссылки

- Mixer
- GELU
- ViT
- BIT
- Funny Mixer Explanation