ETH zürich

MLP-Mixer: an all-MLP Architecture for Vision

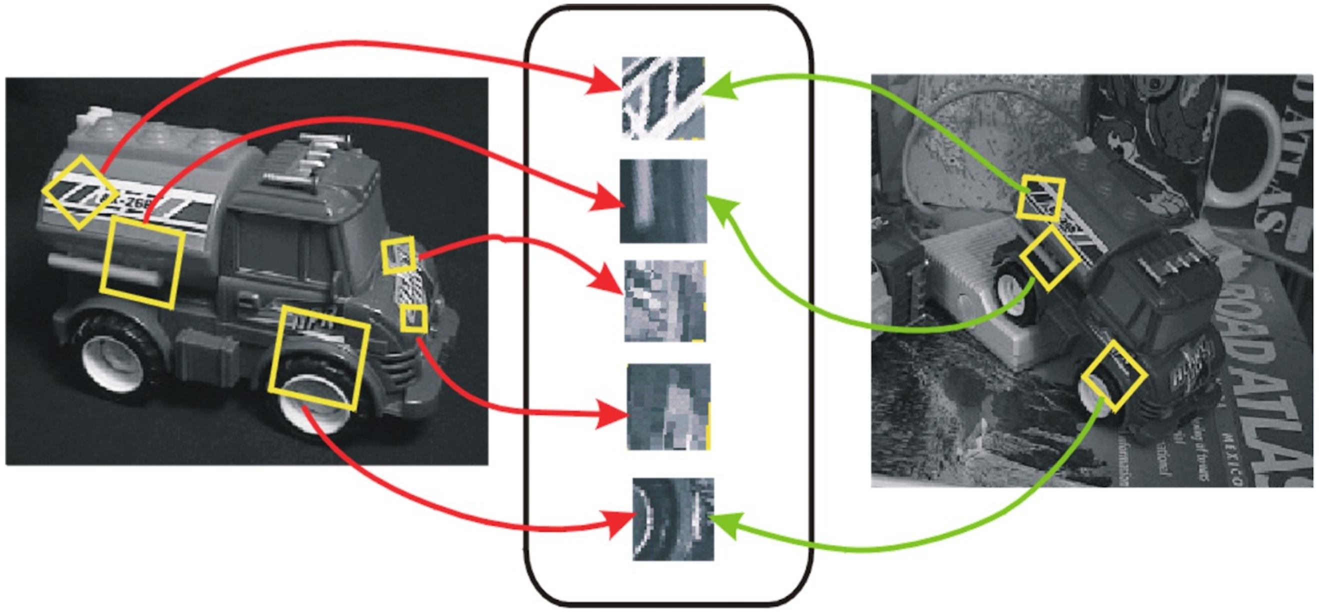
Lara Nonino

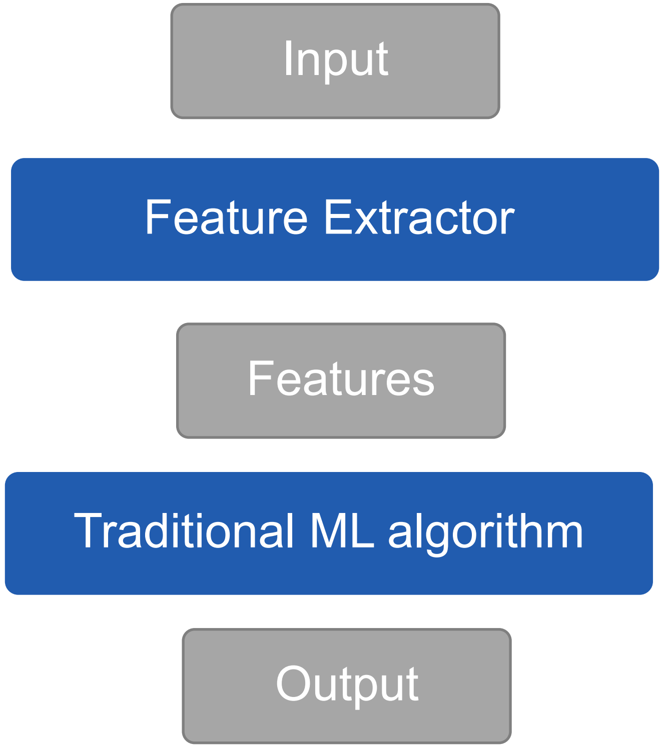
Seminar in Deep Neural Networks (FS 2024)
16 April 2024, ETH Zürich

Traditional Computer Vision

- Hand-crafted image features, meaning that specific filters or feature detectors are designed based on the task.

SIFT descriptor

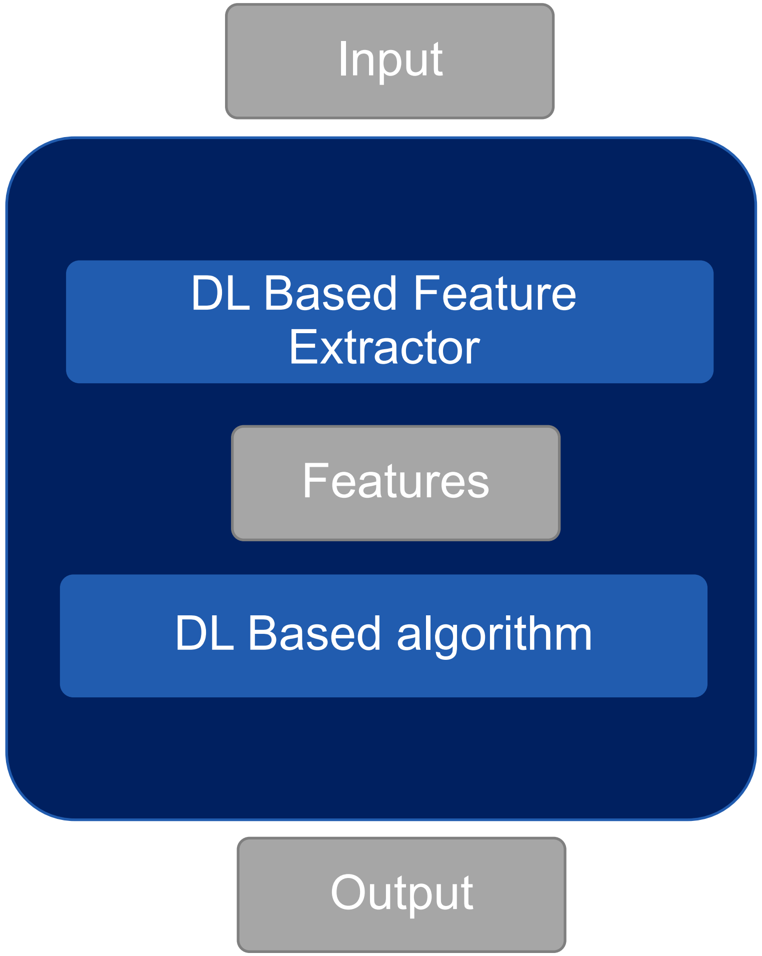




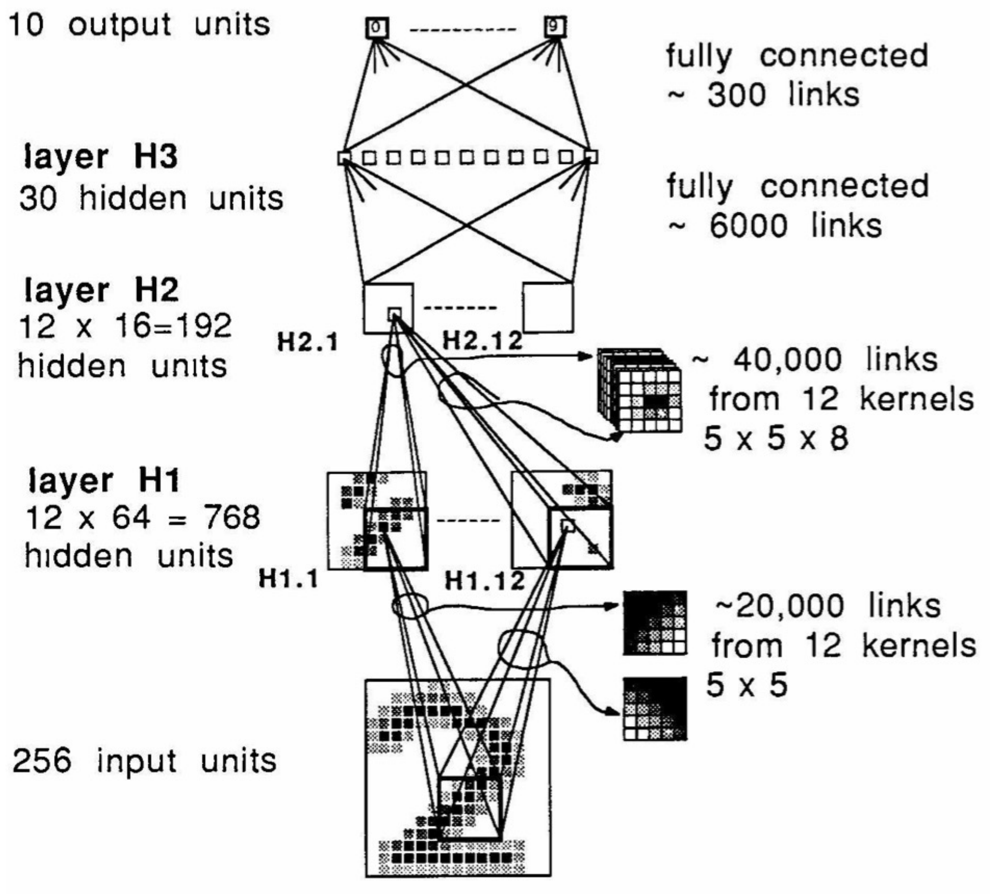
Medium Blogpost [1]

Deep Neural Networks

- Multilayer perceptrons
- Convolutional Neural Networks
- Attention-based Neural Networks



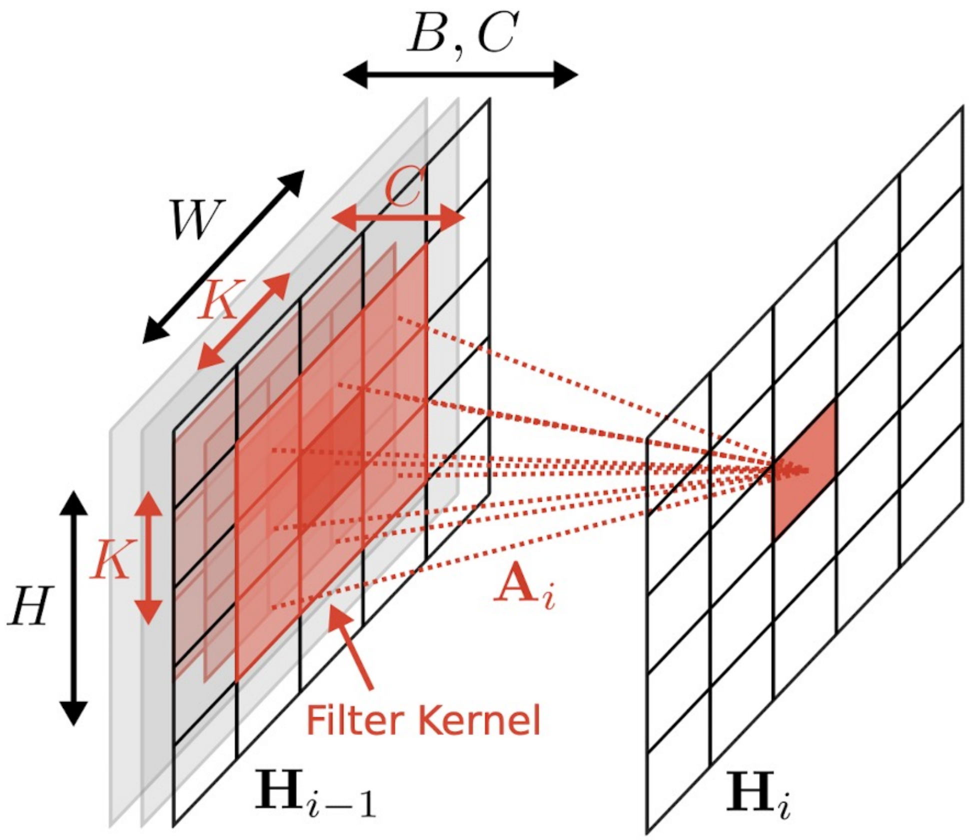
Multilayer perceptrons (MLPs)



1989 - Backpropagation Applied to Handwritten Zip Code Recognition

Yann LeCun et al. [14]

Convolutional Neural Networks (CNNs)

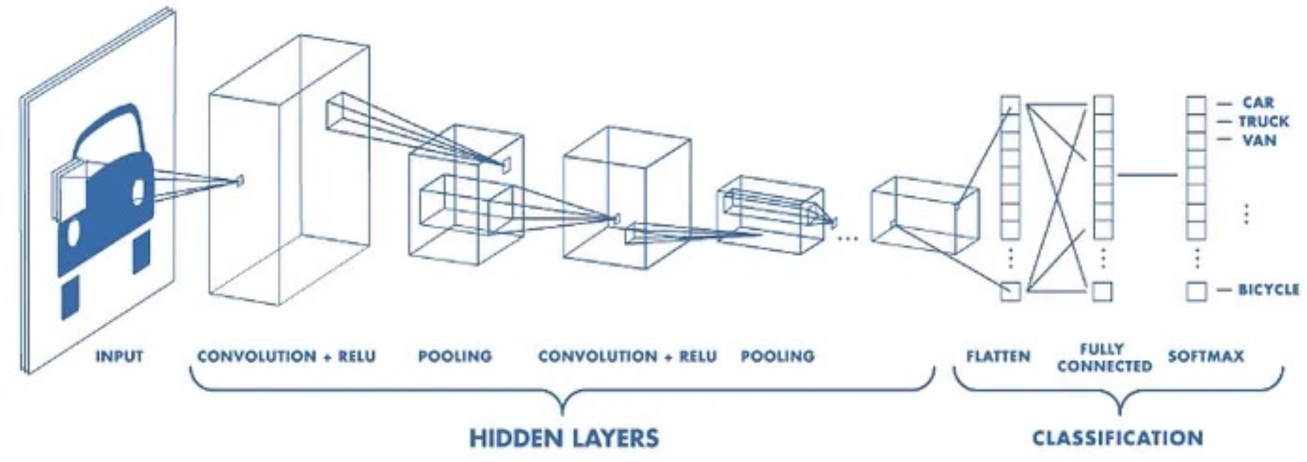


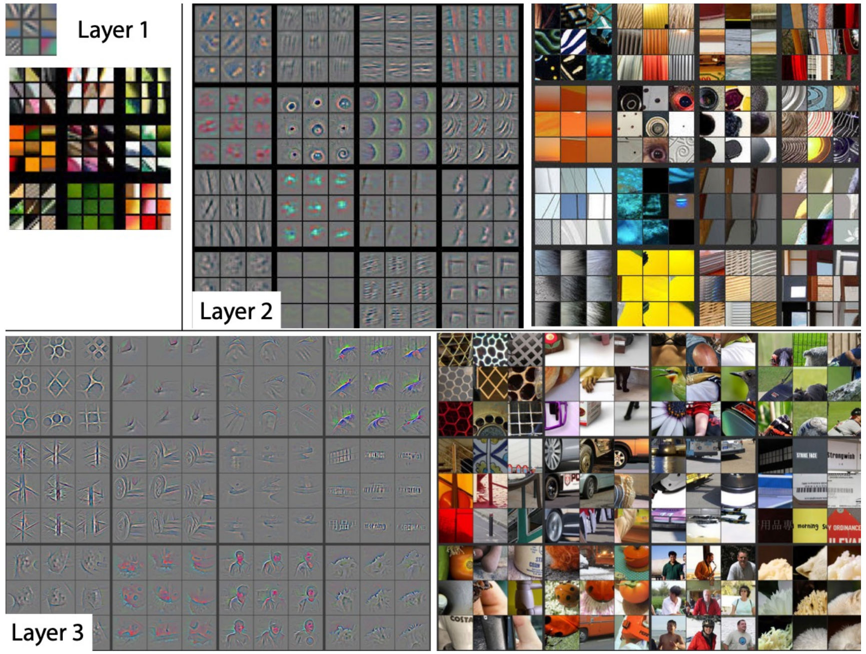
Convolution

$$H_i(x,y) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} K(-m,-n) H_{i-1}(x+m, y+n)$$

Convolutional Neural Networks (CNNs)

- Features are learnt directly from data through convolutions.
- CNNs bring inductive biases (hierarchical structure, local connectivity, parameter sharing, translation equivariance, etc).

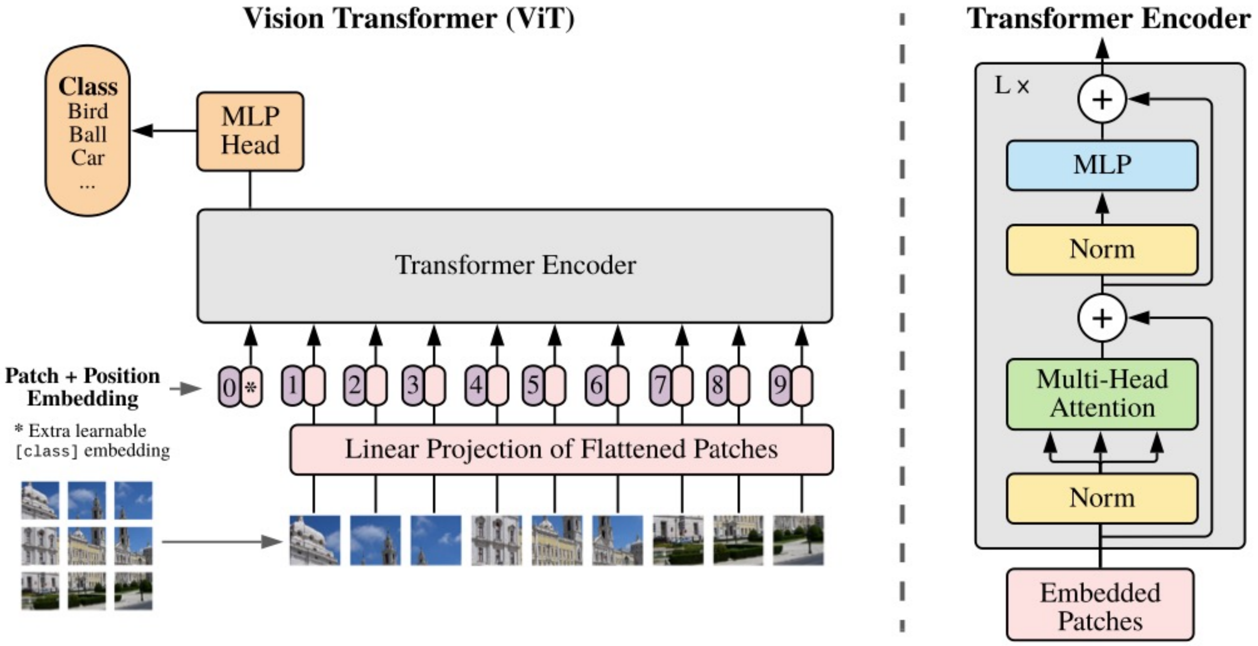




TowardsDataScience Blogpost [2]
Zeiler & Fergus [3]

Attention-based Networks

- Features are learnt directly from data through self-attention.
- Bring fewer inductive biases compared to CNNs (global receptive field, lesser spatial bias, etc).

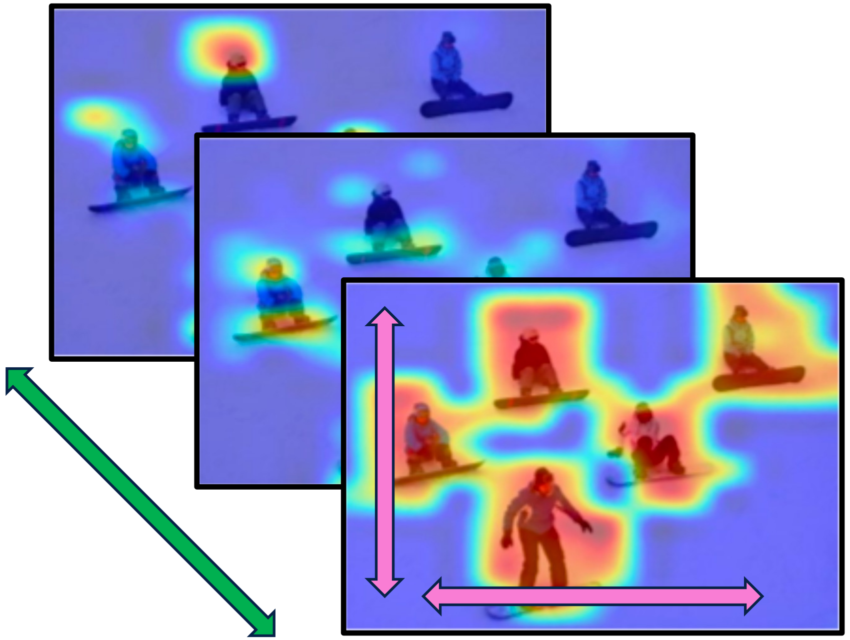


Dosovitskiy et al. [4]

Mixer architecture

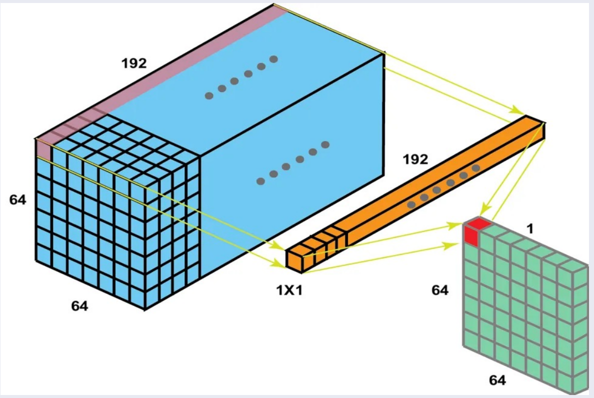
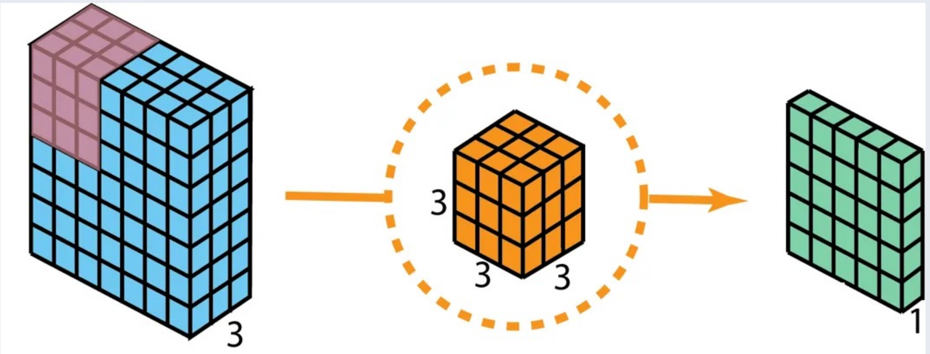
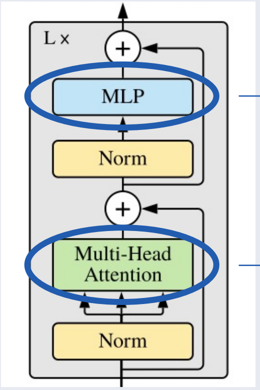
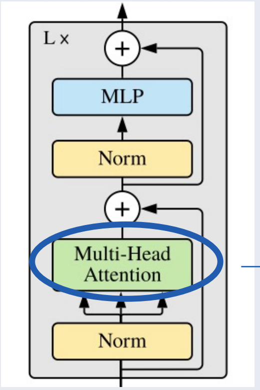
Modern deep vision architectures consist of layers that mix features

- between channels
- between spatial locations



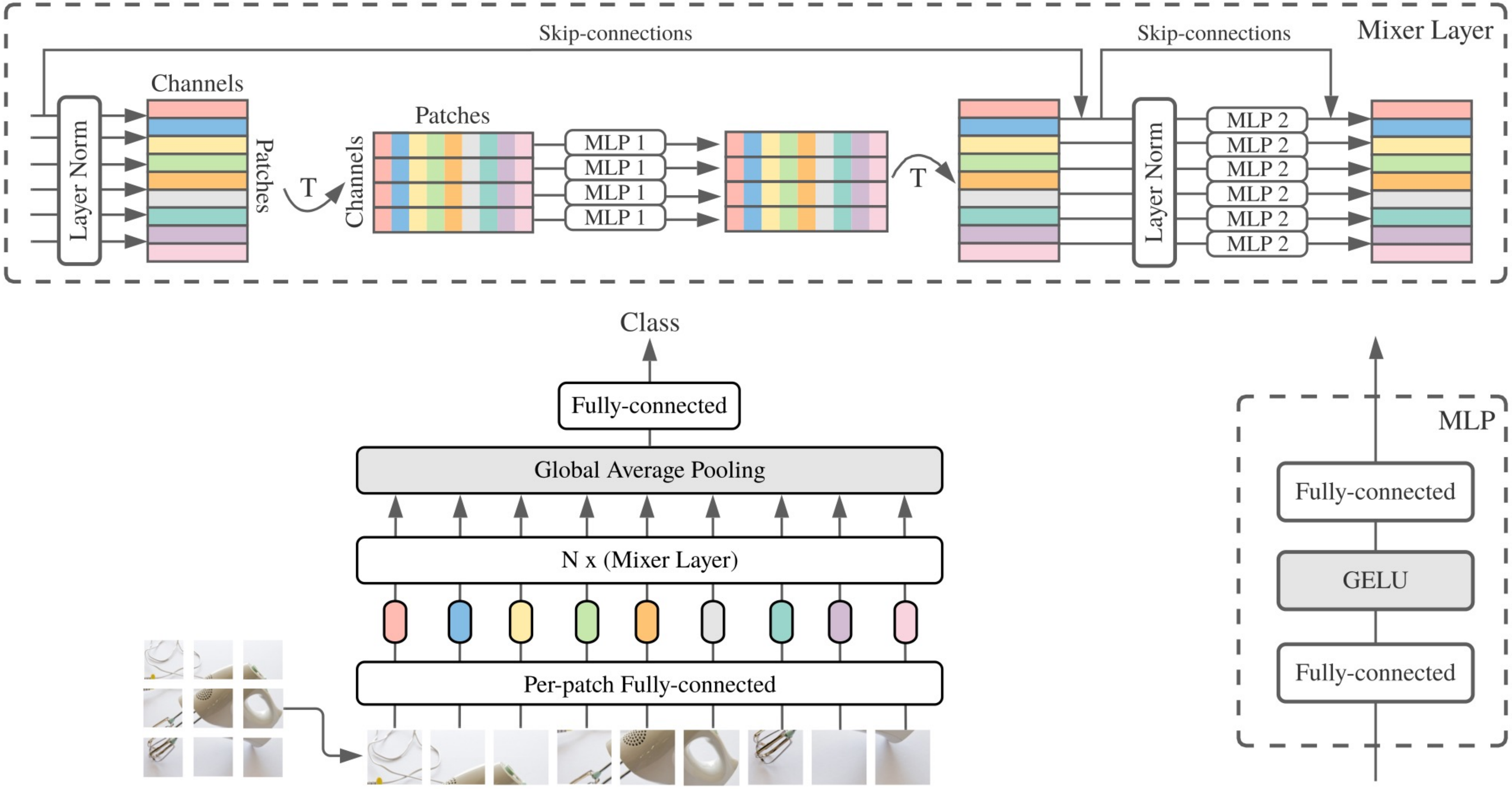
or both at once.

Mixer architecture

	CHANNELS	SPATIAL LOCATIONS
CNNs		
ViTs	 MLP block Self-attention	 Self-attention

Medium Blogpost [5]
Dosovitskiy et al. [4]

Mixer architecture

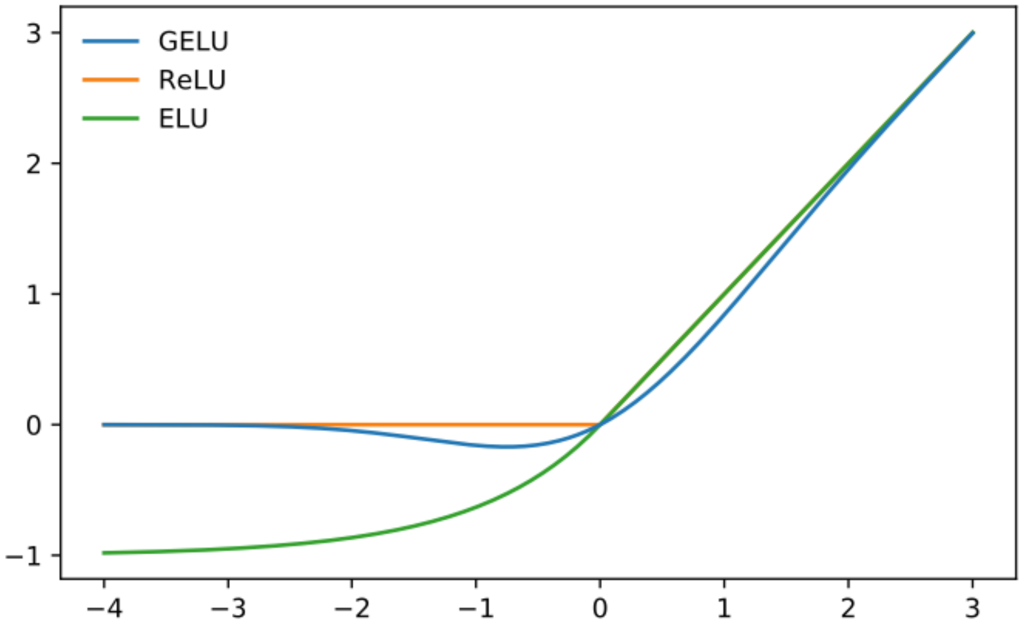


Tolstikhin et al. [6]

Gaussian Error Linear Unit - GELU

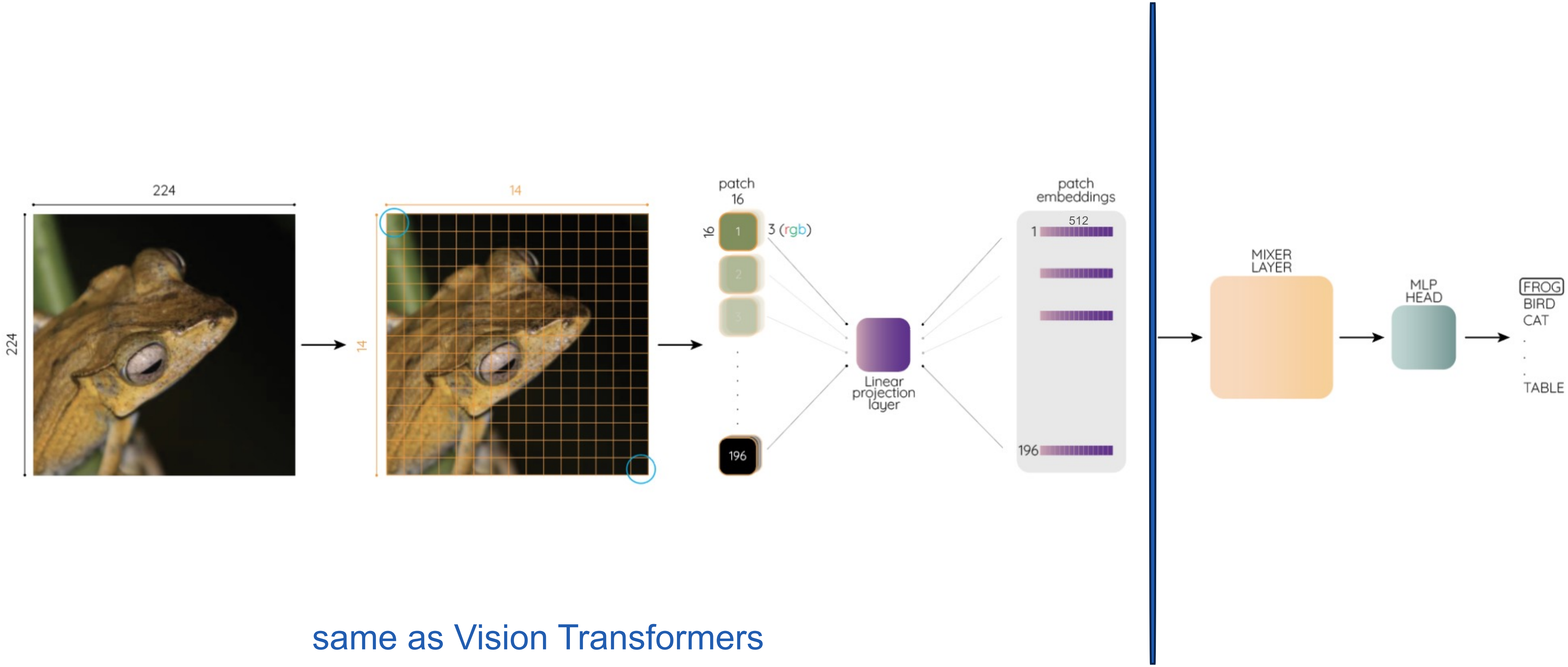
$$GELU(x) = xP(X \leq x) = x\Phi(x)$$

$$X \sim \mathcal{N}(0,1)$$



Weights & Biases Blogpost [7]

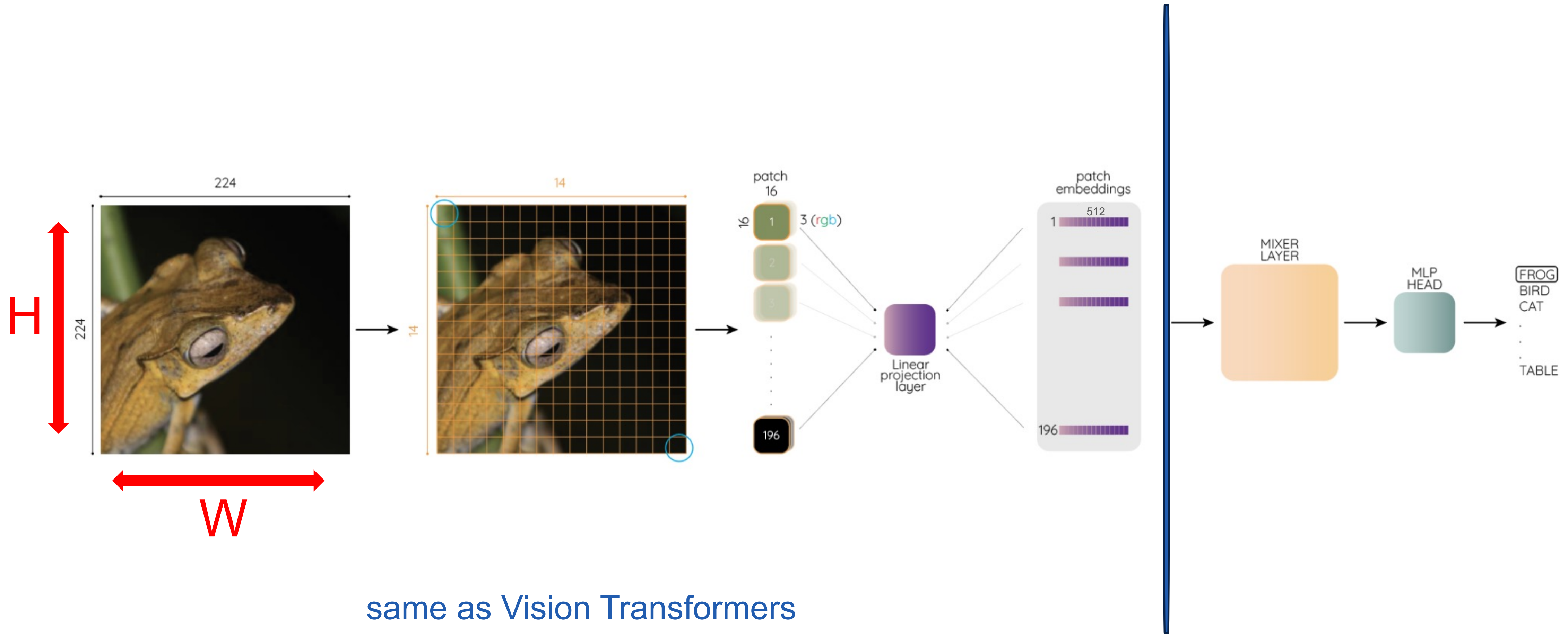
Mixer architecture



same as Vision Transformers

Weights & Biases Blogpost [7]

Mixer architecture

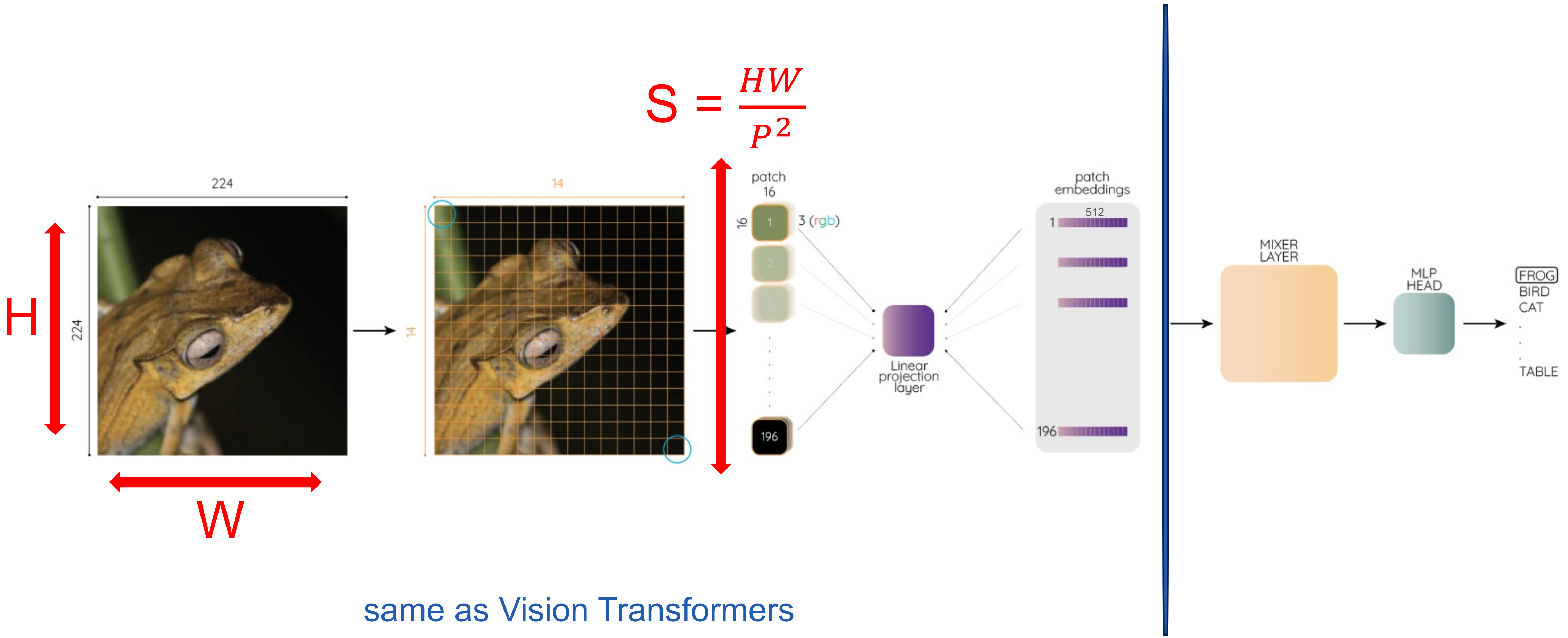


same as Vision Transformers

Weights & Biases Blogpost [7]

Mixer architecture

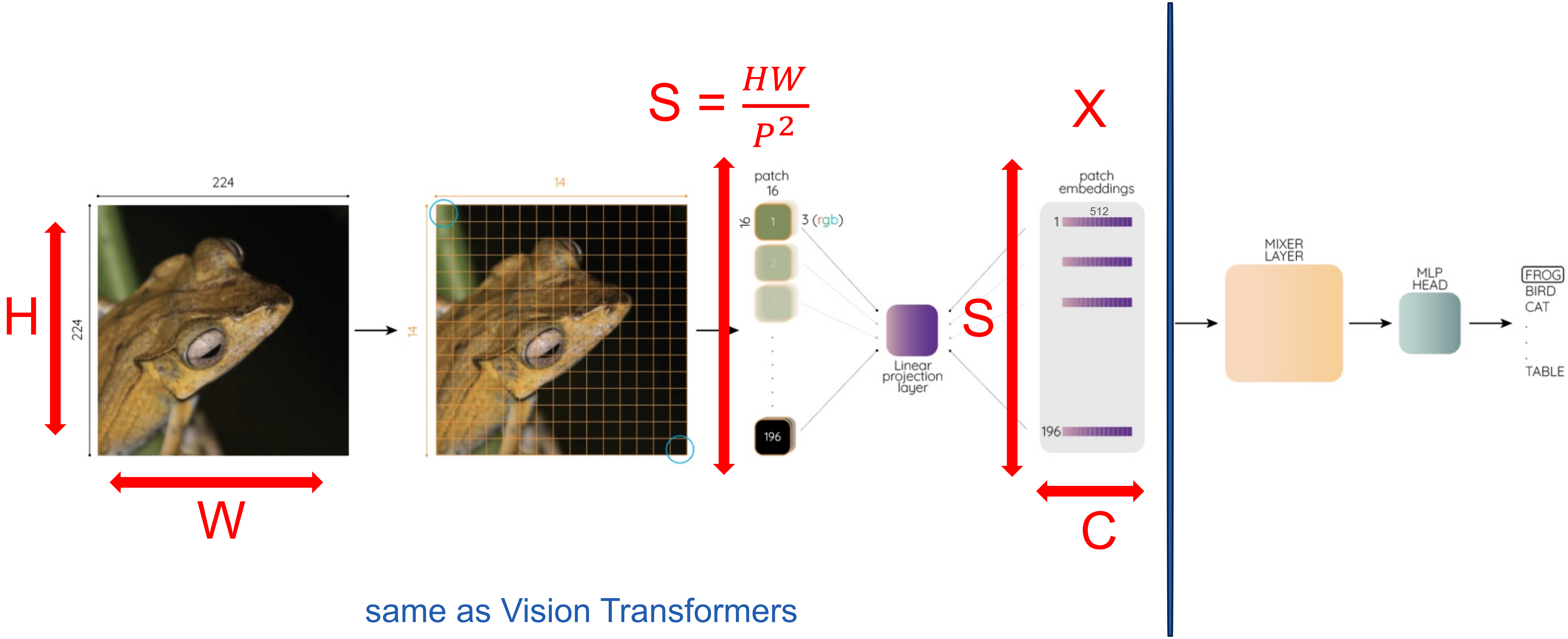
$$S = \frac{HW}{P^2}$$



same as Vision Transformers

Weights & Biases Blogpost [7]

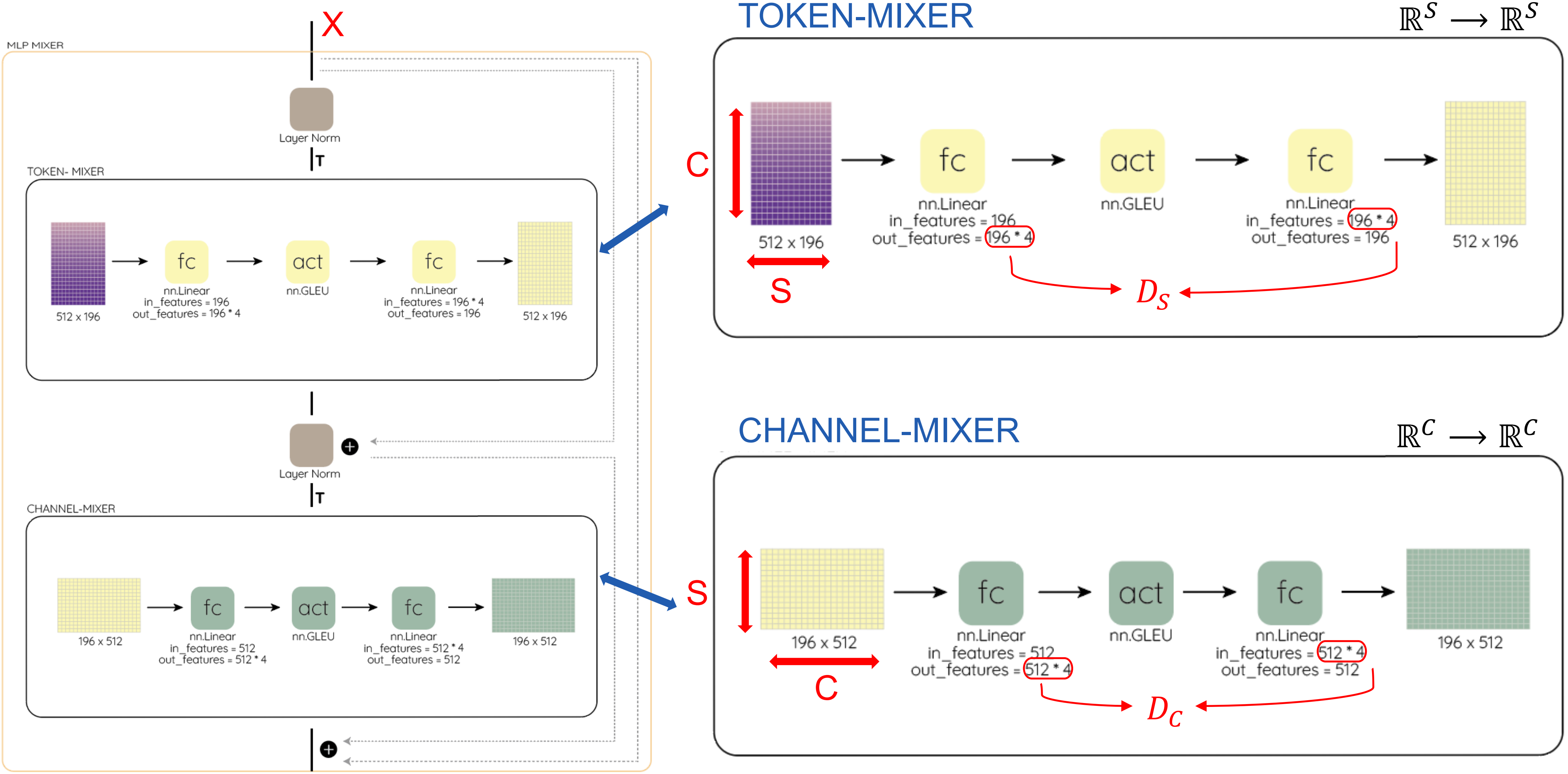
Mixer architecture



same as Vision Transformers

Weights & Biases Blogpost [7]

Mixer architecture



Weights & Biases Blogpost [7]

Experiments

Evaluate according to three primary quantities:

1. Accuracy on the downstream task
2. Total computation cost of pre-training
3. Test-time throughput

Our goal is not to demonstrate state-of-the-art results, but to show that, remarkably, a simple MLP-based model is competitive with today's best convolutional and attention-based models.

Specifications of the Mixer architectures

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

Tolstikhin et al. [6]

Main results

	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

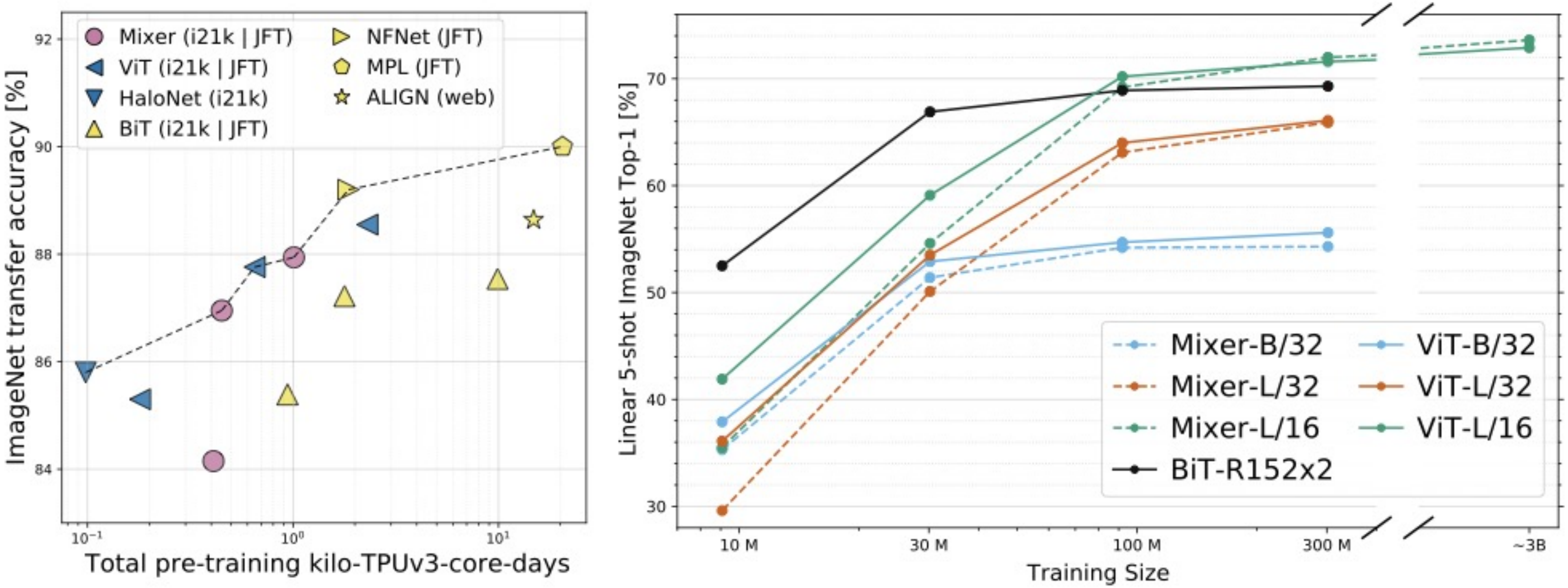
Tolstikhin et al. [6]

Main results

	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

Tolstikhin et al. [6]

Main results



Tolstikhin et al. [6]

Main results

	Image size	Pre-Train Epochs	ImNet top-1	ReaL top-1	Avg. 5 top-1	Throughput (img/sec/core)	TPUv3 core-days
			Pre-trained on ImageNet (with extra regularization)				
● Mixer-B/16	224	300	76.44	82.36	88.33	1384	0.01k(‡)
● ViT-B/16 (☎)	224	300	79.67	84.97	90.79	861	0.02k(‡)
● Mixer-L/16	224	300	71.76	77.08	87.25	419	0.04k(‡)
● ViT-L/16 (☎)	224	300	76.11	80.93	89.66	280	0.05k(‡)
			Pre-trained on ImageNet-21k (with extra regularization)				
● Mixer-B/16	224	300	80.64	85.80	92.50	1384	0.15k(‡)
● ViT-B/16 (☎)	224	300	84.59	88.93	94.16	861	0.18k(‡)
● Mixer-L/16	224	300	82.89	87.54	93.63	419	0.41k(‡)
● ViT-L/16 (☎)	224	300	84.46	88.35	94.49	280	0.55k(‡)
● Mixer-L/16	448	300	83.91	87.75	93.86	105	0.41k(‡)
			Pre-trained on JFT-300M				
● Mixer-S/32	224	5	68.70	75.83	87.13	11489	0.01k
● Mixer-B/32	224	7	75.53	81.94	90.99	4208	0.05k
● Mixer-S/16	224	5	73.83	80.60	89.50	3994	0.03k
● BiT-R50x1	224	7	73.69	81.92	—	2159	0.08k
● Mixer-B/16	224	7	80.00	85.56	92.60	1384	0.08k
● Mixer-L/32	224	7	80.67	85.62	93.24	1314	0.12k
● BiT-R152x1	224	7	79.12	86.12	—	932	0.14k
● BiT-R50x2	224	7	78.92	86.06	—	890	0.14k
● BiT-R152x2	224	14	83.34	88.90	—	356	0.58k
● Mixer-L/16	224	7	84.05	88.14	94.51	419	0.23k
● Mixer-L/16	224	14	84.82	88.48	94.77	419	0.45k
● ViT-L/16	224	14	85.63	89.16	95.21	280	0.65k
● Mixer-H/14	224	14	86.32	89.14	95.49	194	1.01k
● BiT-R200x3	224	14	84.73	89.58	—	141	1.78k
● Mixer-L/16	448	14	86.78	89.72	95.13	105	0.45k
● ViT-H/14	224	14	86.65	89.56	95.57	87	2.30k
● ViT-L/16 [14]	512	14	87.76	90.54	95.63	32	0.65k

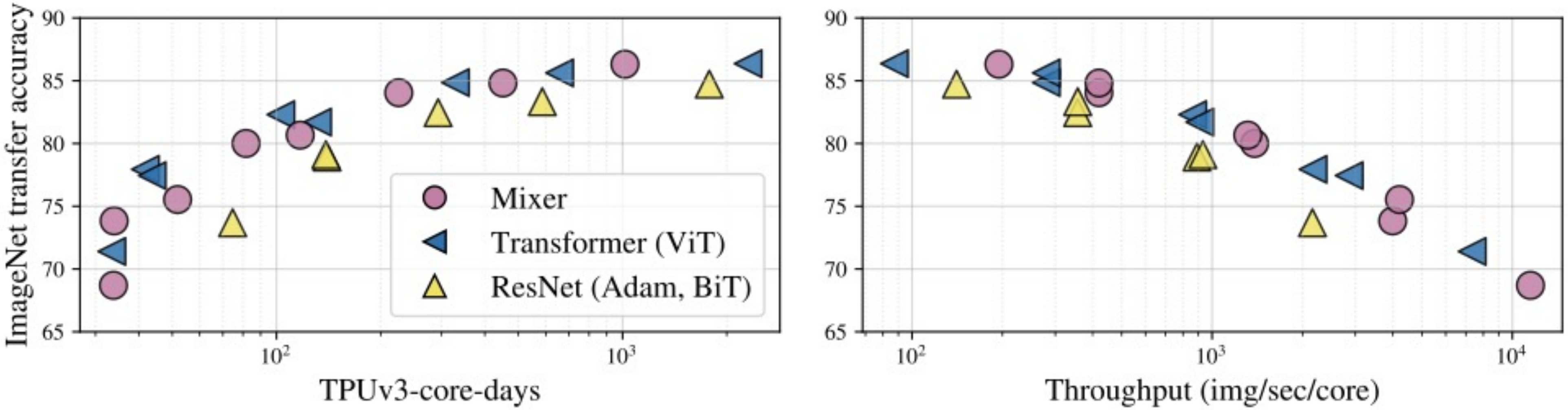
Tolstikhin et al. [6]

Main results

	Image size	Pre-Train Epochs	ImNet top-1	ReaL top-1	Avg. 5 top-1	Throughput (img/sec/core)	TPUv3 core-days
Pre-trained on ImageNet (with extra regularization)							
Mixer-B/16	224	300	76.44	82.36	88.33	1384	0.01k(‡)
ViT-B/16 (☎)	224	300	79.67	84.97	90.79	861	0.02k(‡)
Mixer-L/16	224	300	71.76	77.08	87.25	419	0.04k(‡)
ViT-L/16 (☎)	224	300	76.11	80.93	89.66	280	0.05k(‡)
Pre-trained on ImageNet-21k (with extra regularization)							
Mixer-B/16	224	300	80.64	85.80	92.50	1384	0.15k(‡)
ViT-B/16 (☎)	224	300	84.59	88.93	94.16	861	0.18k(‡)
Mixer-L/16	224	300	82.89	87.54	93.63	419	0.41k(‡)
ViT-L/16 (☎)	224	300	84.46	88.35	94.49	280	0.55k(‡)
Mixer-L/16	448	300	83.91	87.75	93.86	105	0.41k(‡)
Pre-trained on JFT-300M							
Mixer-S/32	224	5	68.70	75.83	87.13	11489	0.01k
Mixer-B/32	224	7	75.53	81.94	90.99	4208	0.05k
Mixer-S/16	224	5	73.83	80.60	89.50	3994	0.03k
BiT-R50x1	224	7	73.69	81.92	—	2159	0.08k
Mixer-B/16	224	7	80.00	85.56	92.60	1384	0.08k
Mixer-L/32	224	7	80.67	85.62	93.24	1314	0.12k
BiT-R152x1	224	7	79.12	86.12	—	932	0.14k
BiT-R50x2	224	7	78.92	86.06	—	890	0.14k
BiT-R152x2	224	14	83.34	88.90	—	356	0.58k
Mixer-L/16	224	7	84.05	88.14	94.51	419	0.23k
Mixer-L/16	224	14	84.82	88.48	94.77	419	0.45k
ViT-L/16	224	14	85.63	89.16	95.21	280	0.65k
Mixer-H/14	224	14	86.32	89.14	95.49	194	1.01k
BiT-R200x3	224	14	84.73	89.58	—	141	1.78k
Mixer-L/16	448	14	86.78	89.72	95.13	105	0.45k
ViT-H/14	224	14	86.65	89.56	95.57	87	2.30k
ViT-L/16 [14]	512	14	87.76	90.54	95.63	32	0.65k

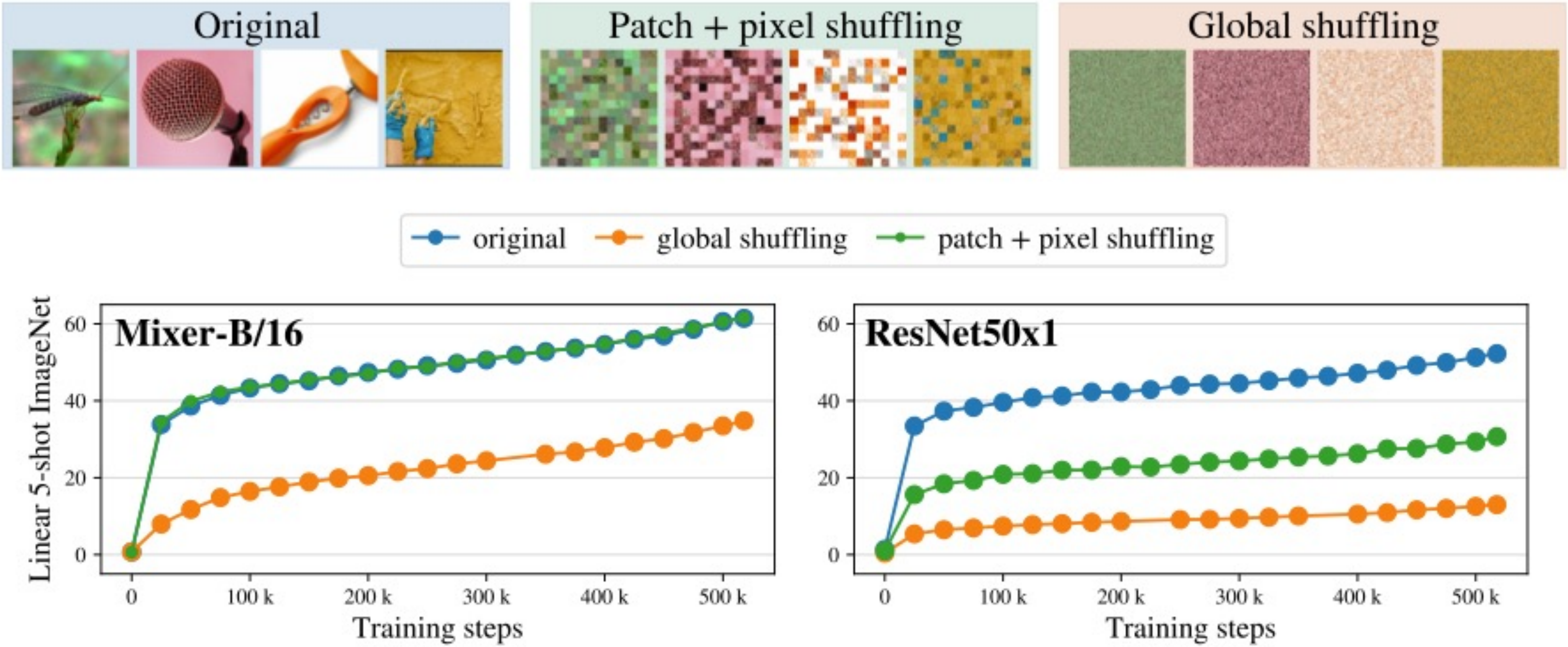
Tolstikhin et al. [6]

Main results



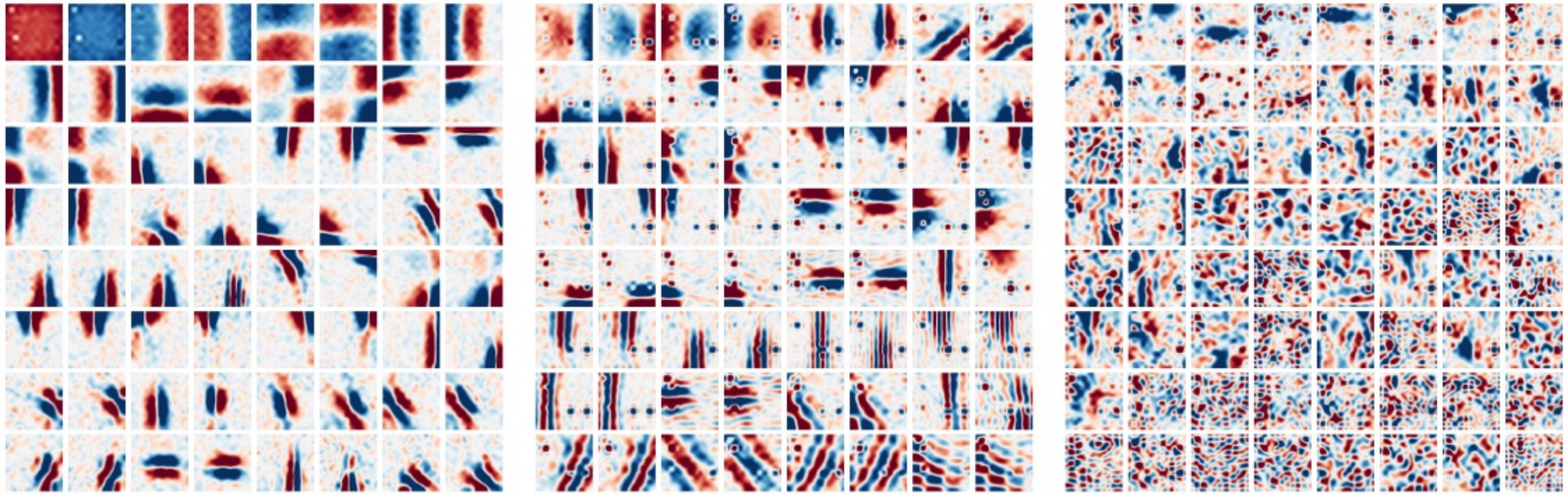
Tolstikhin et al. [6]

Invariance to input permutation

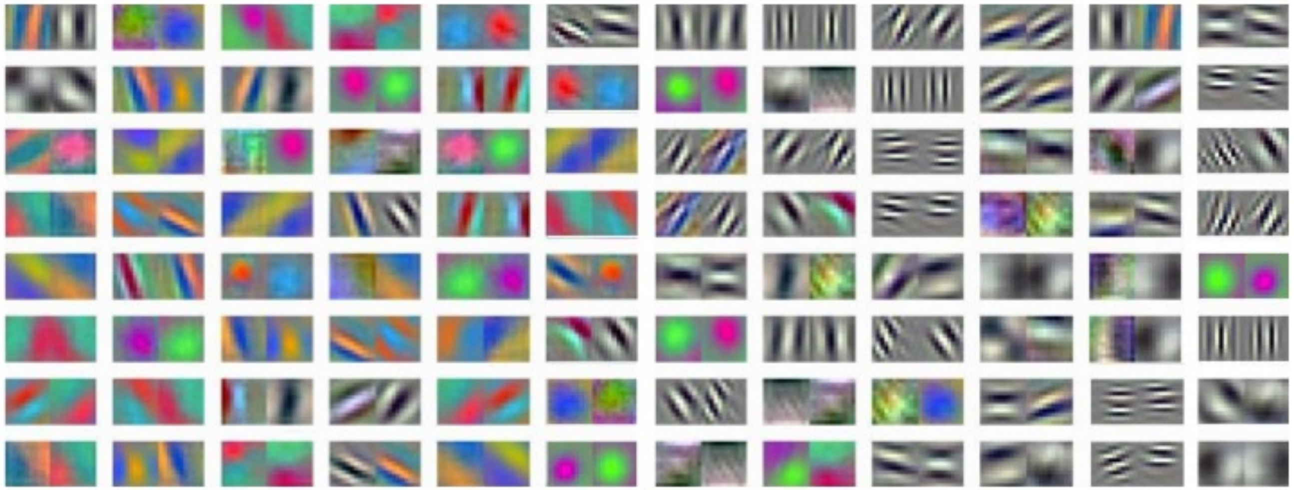


Tolstikhin et al. [6]

Visualization



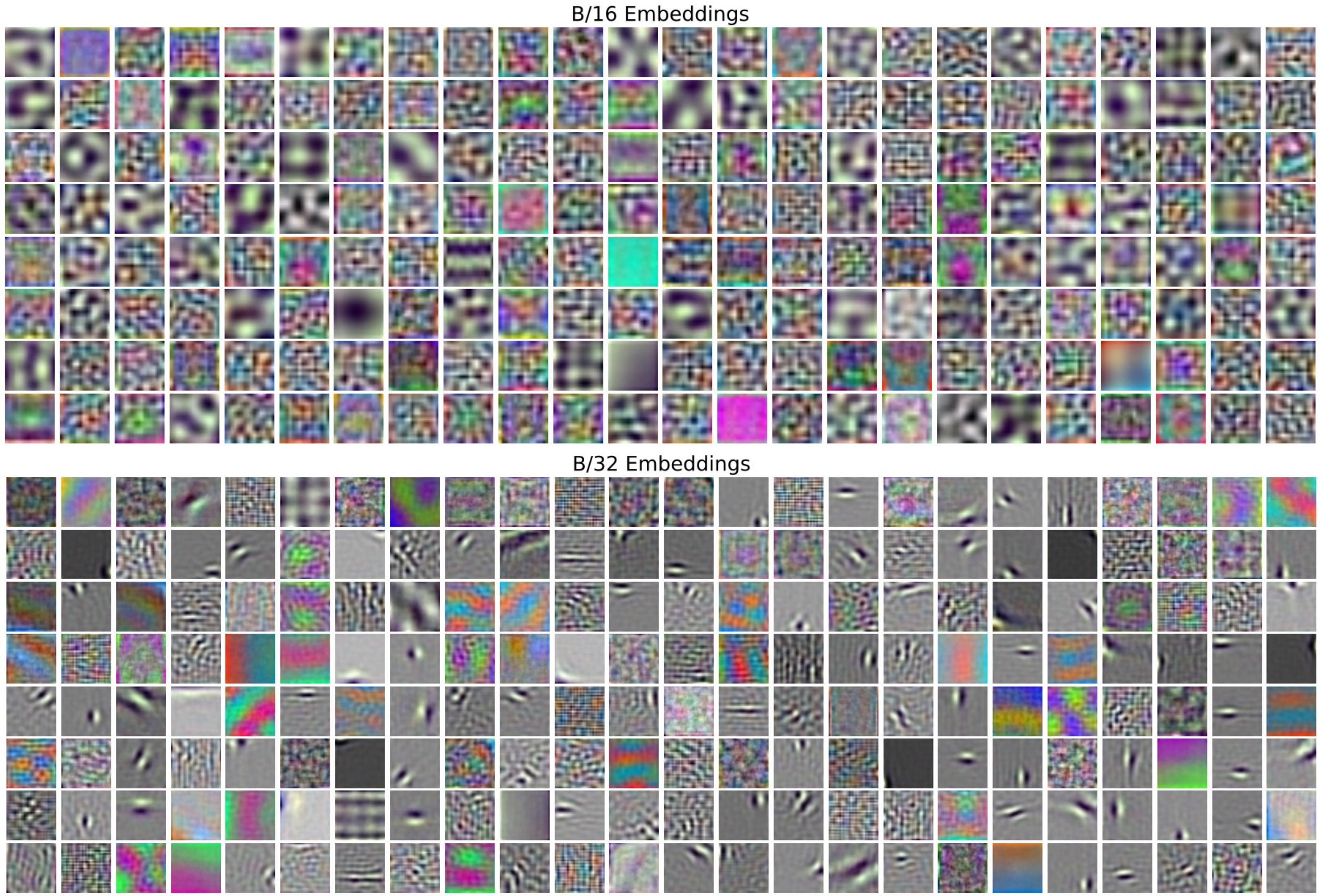
MLP-mixer



AlexNet

Tolstikhin et al. [6]
Shang et al. [10]

Visualization



Linear projection units of the embedding layer for Mixer-B/16 (**Top**) and Mixer-B/32 (**Bottom**) models pre-trained on JFT-300M.

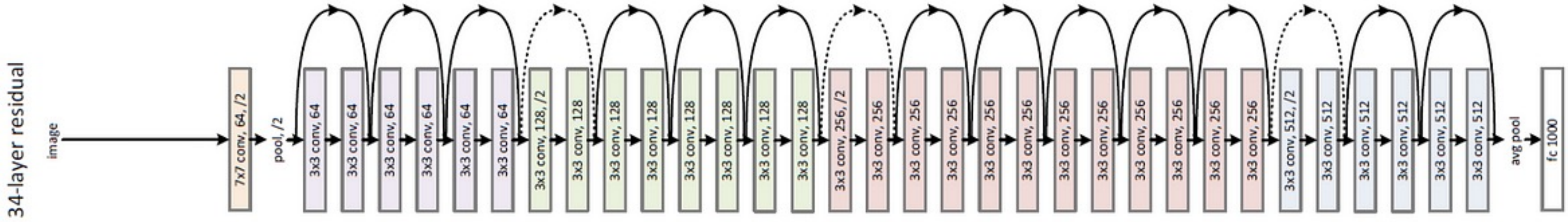
Tolstikhin et al. [6]

Conclusions

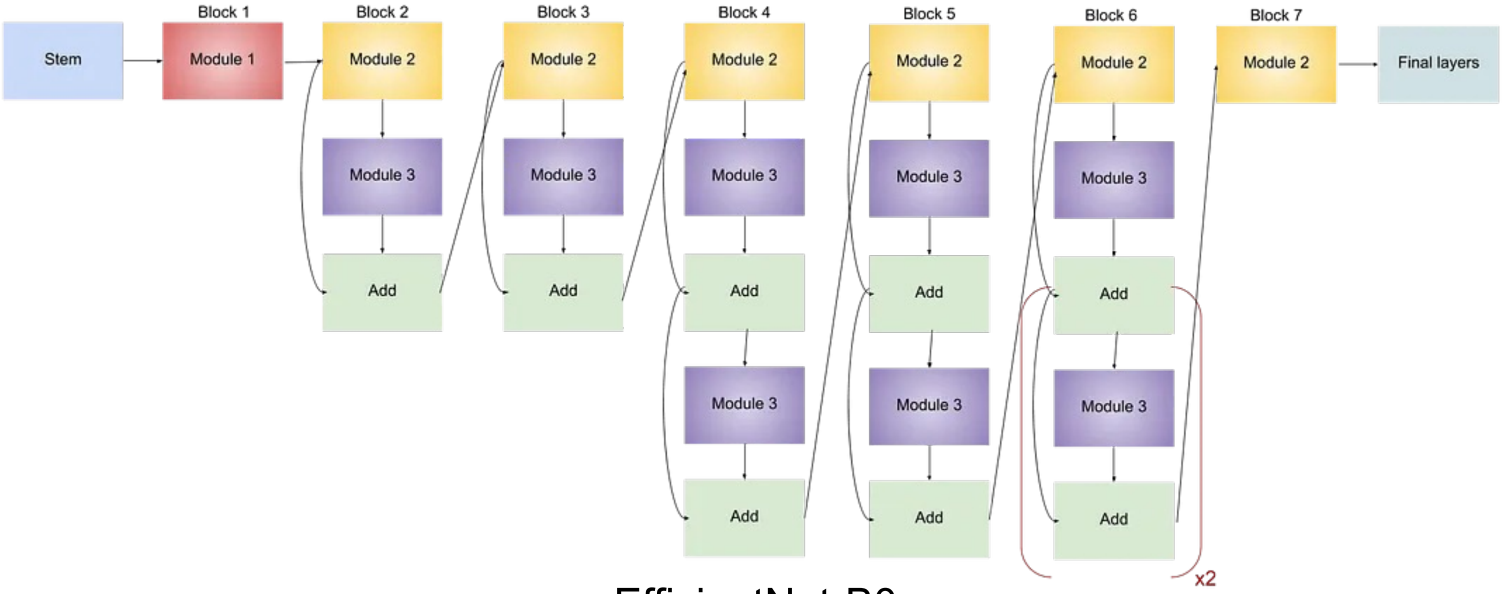
- Very simple architecture for vision
- As good as existing state-of-the-art methods in terms of trade-off between accuracy and computational resources required for training and inference
- Open questions:
  - Practical side: study the features learnt by the model and identify the main differences from those learnt by CNNs and Transformers.
  - Theoretical side: understand the inductive biases hidden in these various features and their role.
- It would be interesting to see whether such a design works in NLP or other domains.

Tolstikhin et al. [6]

Conclusions – similar architectures



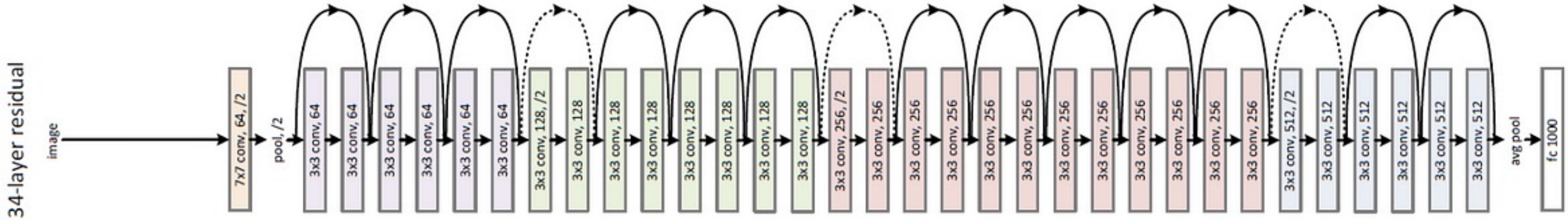
ResNet-34



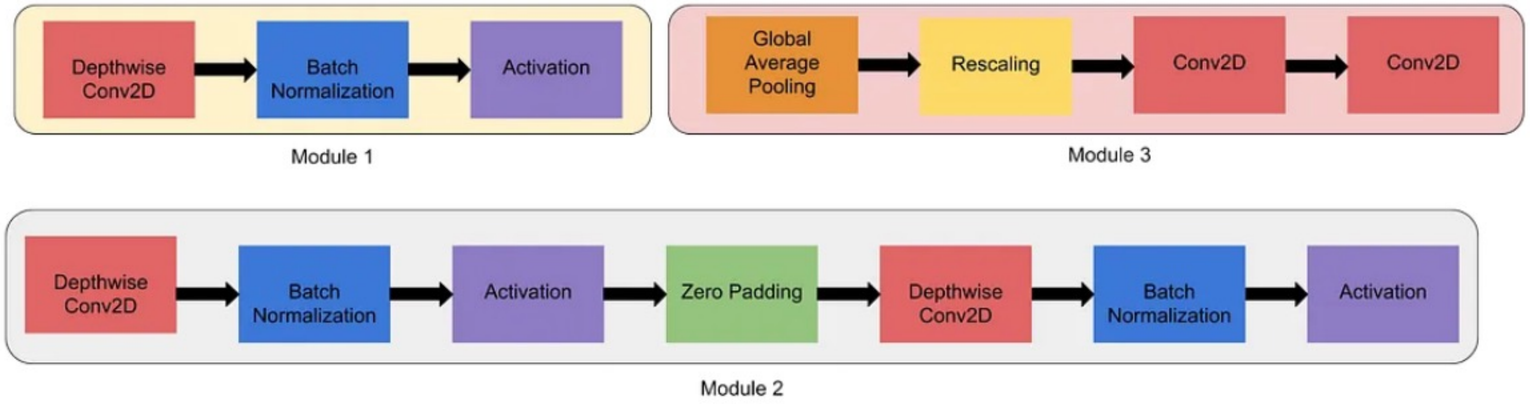
EfficientNet-B0

He et al. [11]
Tan et al. [12]

Conclusions – similar architectures



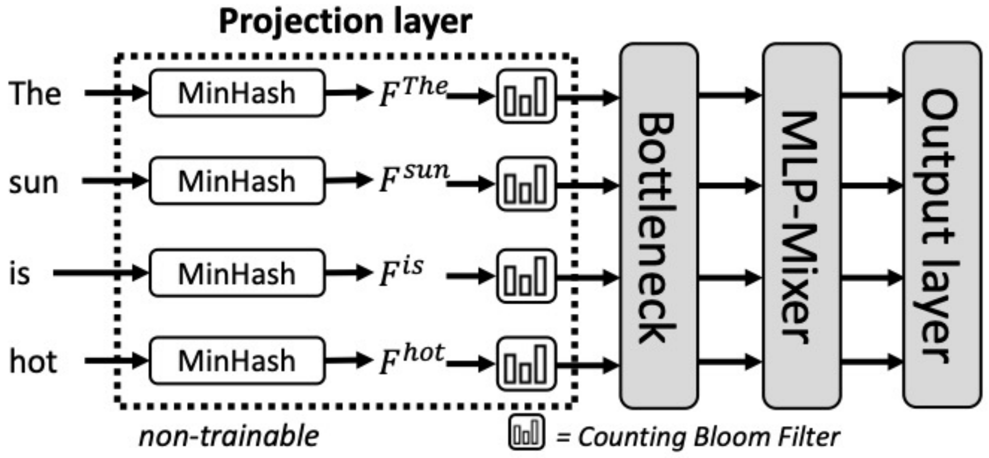
ResNet-34



EfficientNet-B0

He et al. [11]
Tan et al. [12]

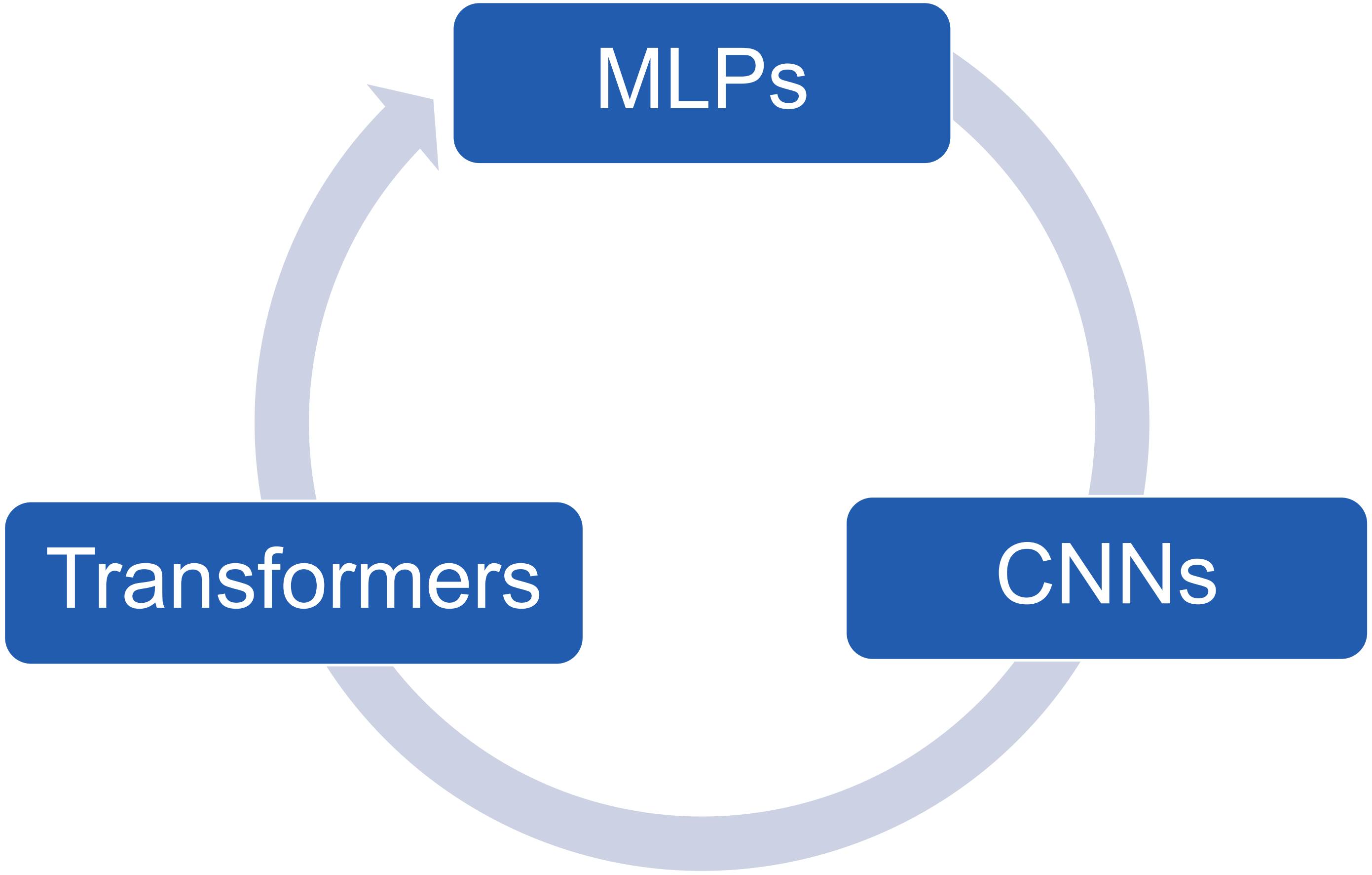
Conclusions – similar architectures



Model	# Param.	EN	ES	FR	DE	HI	JA	PT	TR	ZH	Avg
		Intent Accuracy									
LSTM	28M	96.1	93.0	94.7	94.0	84.5	91.2	92.7	81.1	92.5	91.1
mBERT	170M	98.3	97.4	98.6	98.5	94.5	98.6	97.4	91.2	97.5	96.9
Transformer	2M	96.8	92.1	93.1	93.2	79.6	90.7	92.1	78.3	88.1	89.3
pQRNN	2M(8bit)	98.0	97.0	97.9	96.6	**90.7**	88.7	**97.2**	86.2	93.5	94.0
pNLP-Mixer	1M(8bit)	**98.1**	**97.1**	**98.1**	**97.3**	**90.7**	**92.3**	**97.2**	**87.3**	**95.1**	**94.8**

Fusco et al. [13]

Conclusions – closing the circle



Thank you!

MLP-mixer code

```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)
```


Tolstikhin et al. [6]

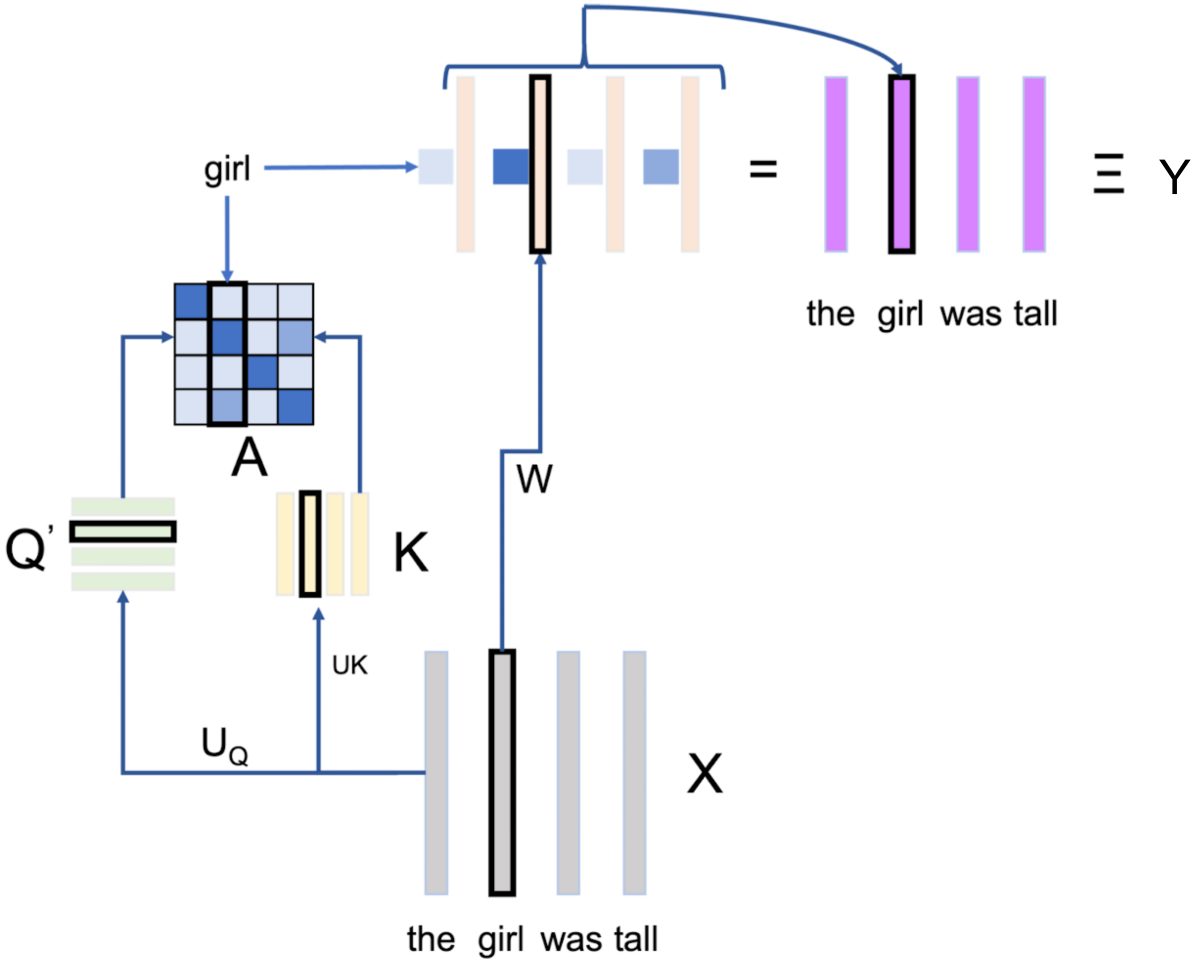
Convolutional Neural Networks (CNNs)

brought to extreme using dense matrix multiplication

- 2012 – AlexNet
- 2015 – State-of-the-art model using convolutions with small 3x3 kernels
- 2016 – Skip-connections and batch-normalization enabled very deep NNs
- 2016 – Sparse convolutions together with depth-wise variants
- 2018 – Augment CNNs with non-local operations
- 2019 – Shared parameters in depth-wise convolutions for NLP

matrix multiplications are applied row-wise or column-wise on the 'patches x features' input table

Attention-based Networks



$$Y = WXA^T$$

$$A = \text{softmax}(\beta Q'K), \quad a_{st} \equiv \frac{e^{\beta[Q'K]_{st}}}{\sum_r e^{\beta[Q'K]_{sr}}}$$

→ β: inverse temperature, e.g. $1/\beta = \sqrt{q}$

→ each column is normalized to sum to 1

Computation cost

'patches x features' matrix
$X \in \mathbb{R}^S \times \mathbb{R}^C$

MLP-mixer

$$\mathbf{U}_{*,i} = \mathbf{X}_{*,i} + \mathbf{W}_2\,\sigma\big(\mathbf{W}_1\,\mathrm{LayerNorm}(\mathbf{X})_{*,i}\big), \qquad \text{for } i = 1 \ldots C,$$
$$\mathbf{Y}_{j,*} = \mathbf{U}_{j,*} + \mathbf{W}_4\,\sigma\big(\mathbf{W}_3\,\mathrm{LayerNorm}(\mathbf{U})_{j,*}\big), \qquad \text{for } j = 1 \ldots S.$$

$$W_1 \in \mathbb{R}^{D_S \times S}$$
$$W_2 \in \mathbb{R}^{S \times D_S}$$
$$W_3 \in \mathbb{R}^{D_C \times C}$$
$$W_4 \in \mathbb{R}^{\mathbb{R}^{C \times D_C}}$$

Total cost: linear in #input_pixels

Vision Transformer

$$A = \mathrm{softmax}(\beta Q' K), \qquad a_{st} \equiv \frac{e^{\beta [Q'K]_{st}}}{\sum_r e^{\beta [Q'K]_{sr}}}$$

Feed Forward Network: $Y = WXA^T$

$$Q \in \mathbb{R}^{C \times S}$$
$$K \in \mathbb{R}^{C \times S}$$
$$W \in \mathbb{R}^{M \times S}$$
$$Y_t = \sum_s a_{t,s} W x^t$$

Total cost: quadratic in #input_pixels

Tolstikhin et al. [6]
Vaswani et al. [8]

Sources

[1]: https://medium.com/@deepanshut041/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40

[2]: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[3]: Matthew D. Zeiler, Rob Fergus: Visualizing and Understanding Convolutional Networks. *arXiv preprint arXiv:1311.2901*, 2013.

[4]: Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*, 2021.

[5]: https://medium.com/analytics-vidhya/talented-mr-1x1-comprehensive-look-at-1x1-convolution-in-deep-learning-f6b355825578

[6]: Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, Alexey Dosovitskiy: MLP-Mixer: An all-MLP Architecture for Vision. *arXiv preprint arXiv:2105.01601*, 2021.

[7]: https://wandb.ai/wandb_fc/pytorch-image-models/reports/Is-MLP-Mixer-a-CNN-in-Disguise---Vmlldzo4NDE1MTU#:~:text=(Body)%20Mixer%20Layer&text=If%20so%2C%20how%3F%22%20a,extreme%20case%20of%20a%20CNN!

[8]: Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin: Attention Is All You Need. *arXiv preprint arXiv:1706.03762,* 2017.

Sources

[9]: Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*, 2012.

[10]: Wenling Shang, Kihyuk Sohn, Diogo Almeida, Honglak Lee: Enderstanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units. *arXiv preprint arXiv:1603.05201*, 2016.

[11]: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun: Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[12]: Mingxing Tan, Quoc V. Le: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv preprint arXiv:1905.11946*, 2019.

[13]: Francesco Fusco, Damian Pascual, Peter Staar, Diego Antognini: pNLP-Mixer: an Efficient all-MLP Architecture for Language. *arXiv preprint arXiv:2202.04350*, 2023.

[14]: Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne Hubbard, Lawrence D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation, 1(4):541-551*, 1989.

ETH zürich

Lara Nonino
lnonino@student.ethz.ch

Seminar in Deep Neural Networks (FS 2024)