NeurIPS2021 – key takeaways*

Paulina Tomaszewska

***regarding Computer Vision and Knowledge Transfer**

MLP-Mixer: An all-MLP Architecture for Vision (Tolstikhin et al.)

*from May 21' → 164 citations (Semantic Scholar)

Constant channel, different patches

Constant patch, different channes

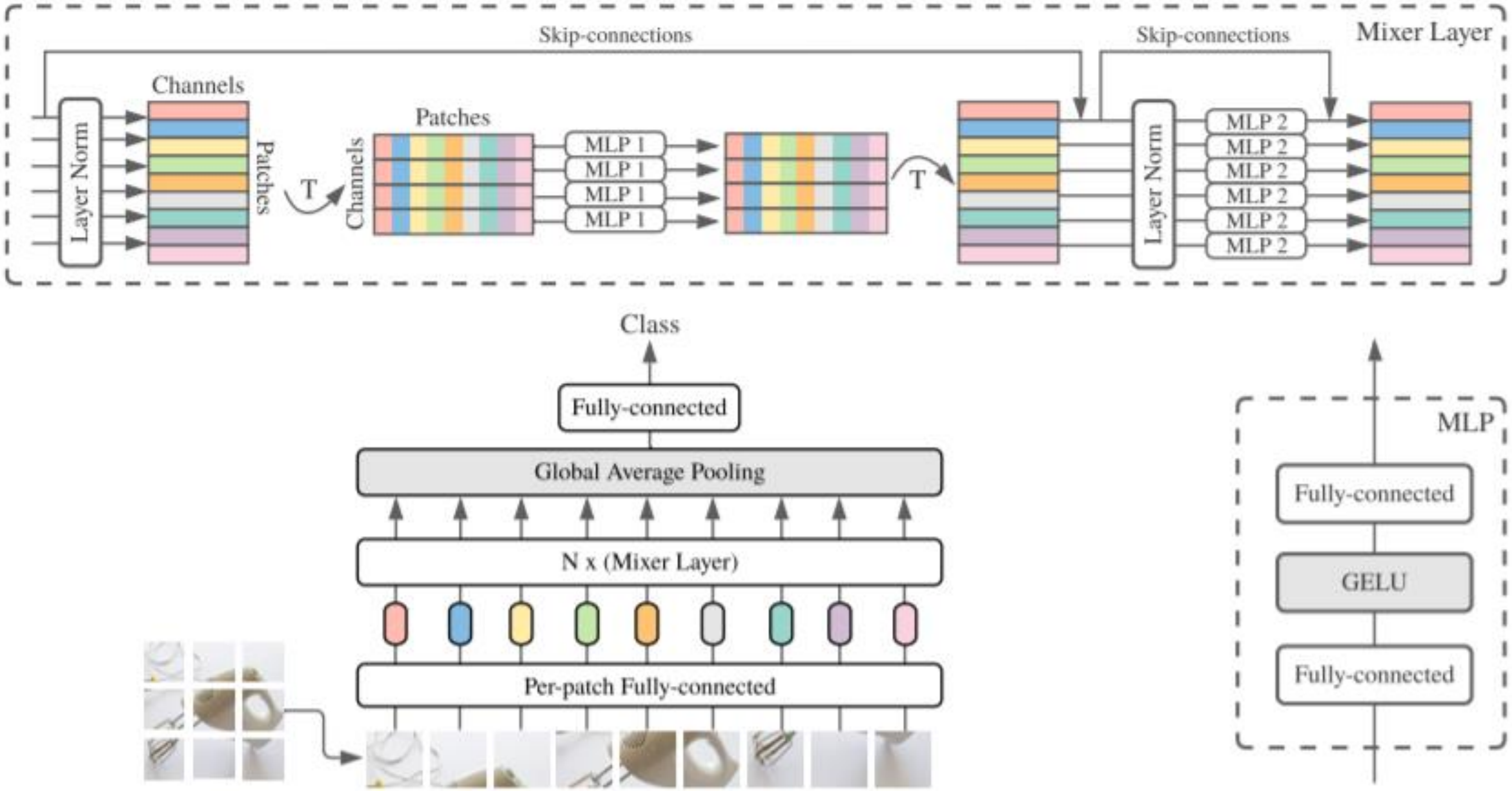
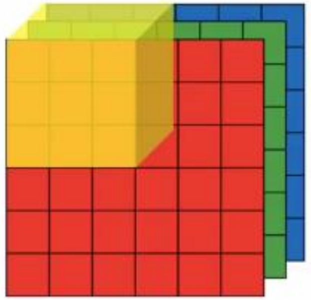


Figure 1: MLP-Mixer consists of per-patch linear embeddings, Mixer layers, and a classifier head. Mixer layers contain one token-mixing MLP and one channel-mixing MLP, each consisting of two fully-connected layers and a GELU nonlinearity. Other components include: skip-connections, dropout, and layer norm on the channels.

- In classical MLP, the image would be represented as a vector
- LN: normalize the activations of the previous layer <u>for each given example in a batch independently</u> (mean = 0, std = 1)



https://stackoverflow.com/questions/47982594/how-a-convolutional-neural-net-handles-channels

GeLU = Gaussian Error Linear Unit

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

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

- Uses standard Gaussian cumulative distribution function
- Smoother than ReLU

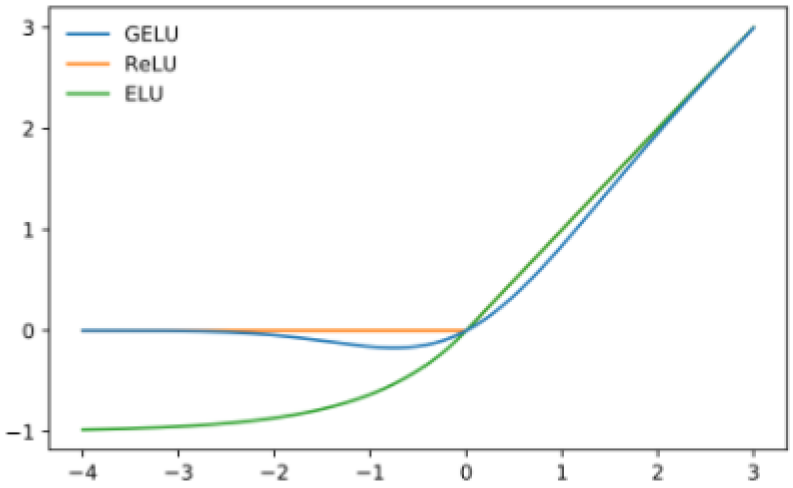


Figure 1: The GELU ($\mu = 0, \sigma = 1$), ReLU, and ELU ($\alpha = 1$).

Another „degree of freedom" when searching for the optimal architecture?

https://paperswithcode.com/method/gelu

Invariance to input permutations

- Differences in inductive bias of Mixer and CNN architectures
- Same permutation is used across all images

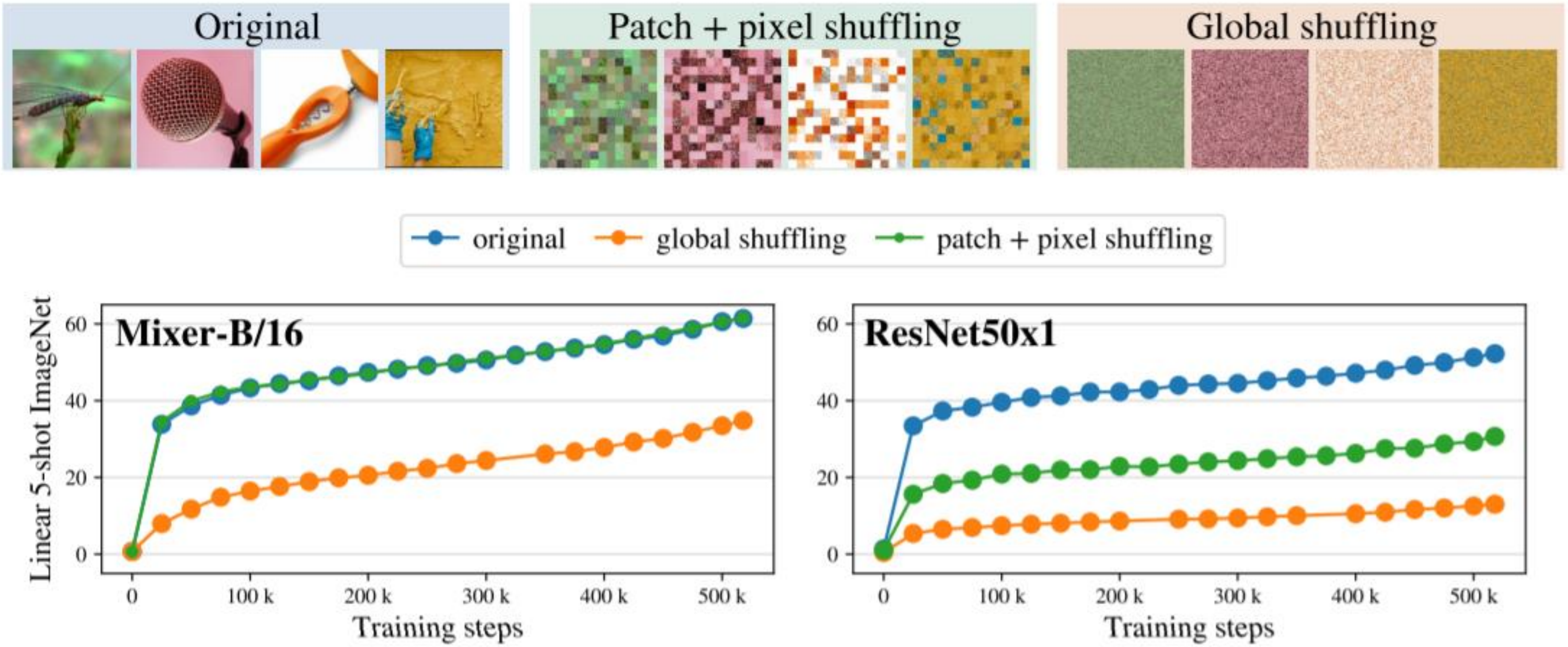
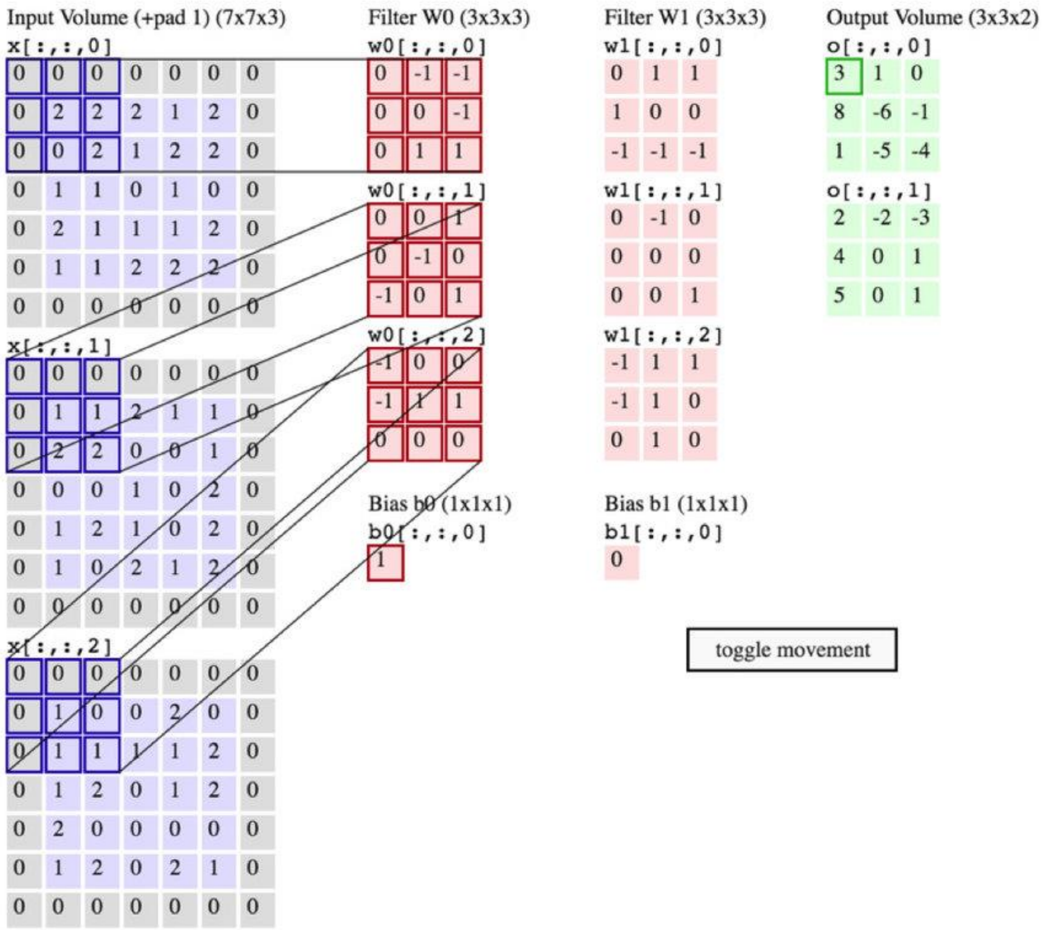


Figure 4: **Top:** Input examples from ImageNet before permuting the contents (left); after shuffling the 16 × 16 patches and pixels within the patches (center); after shuffling pixels globally (right). **Bottom:** Mixer-B/16 (left) and ResNet50x1 (right) trained with three corresponding input pipelines.

No Conv?



https://stats.stackexchange.com/questions/369227/rgb-images-as-input-to-cnn

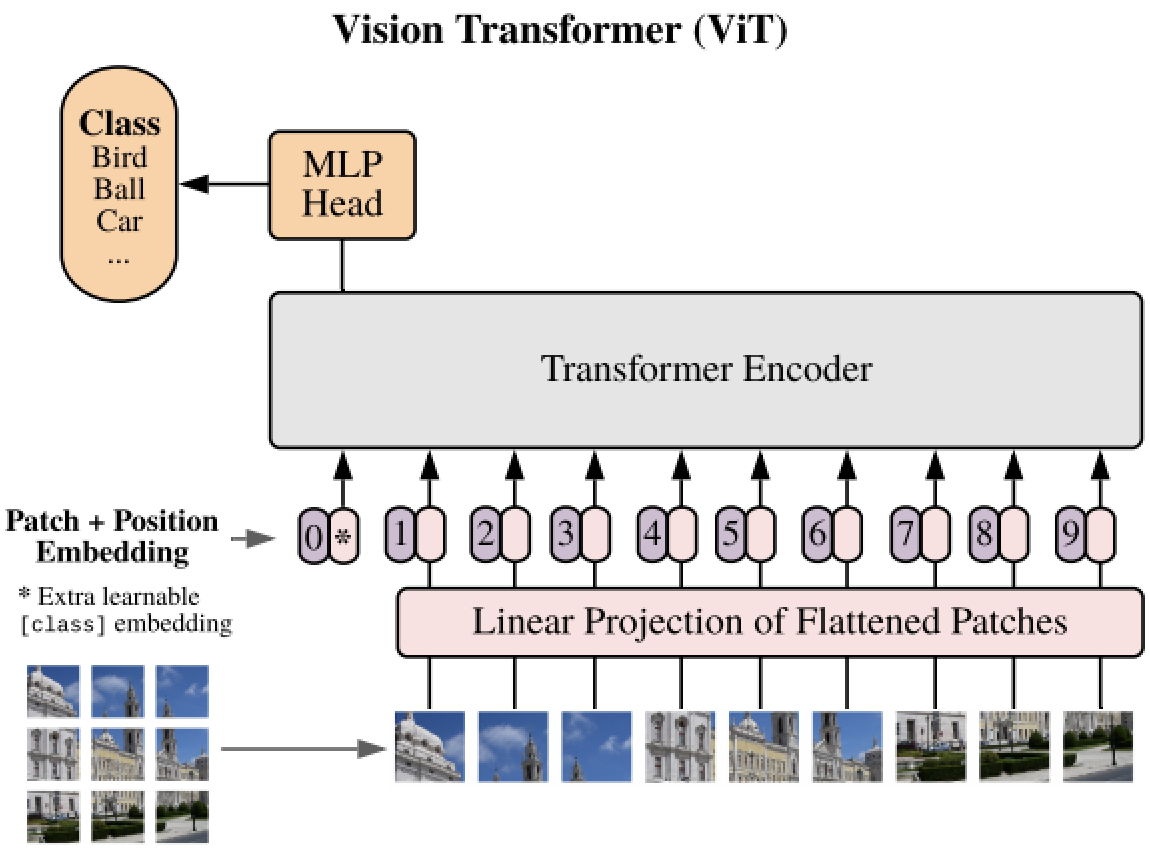
E MLP-Mixer code

```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)
```

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

MLP-Mixer: summary

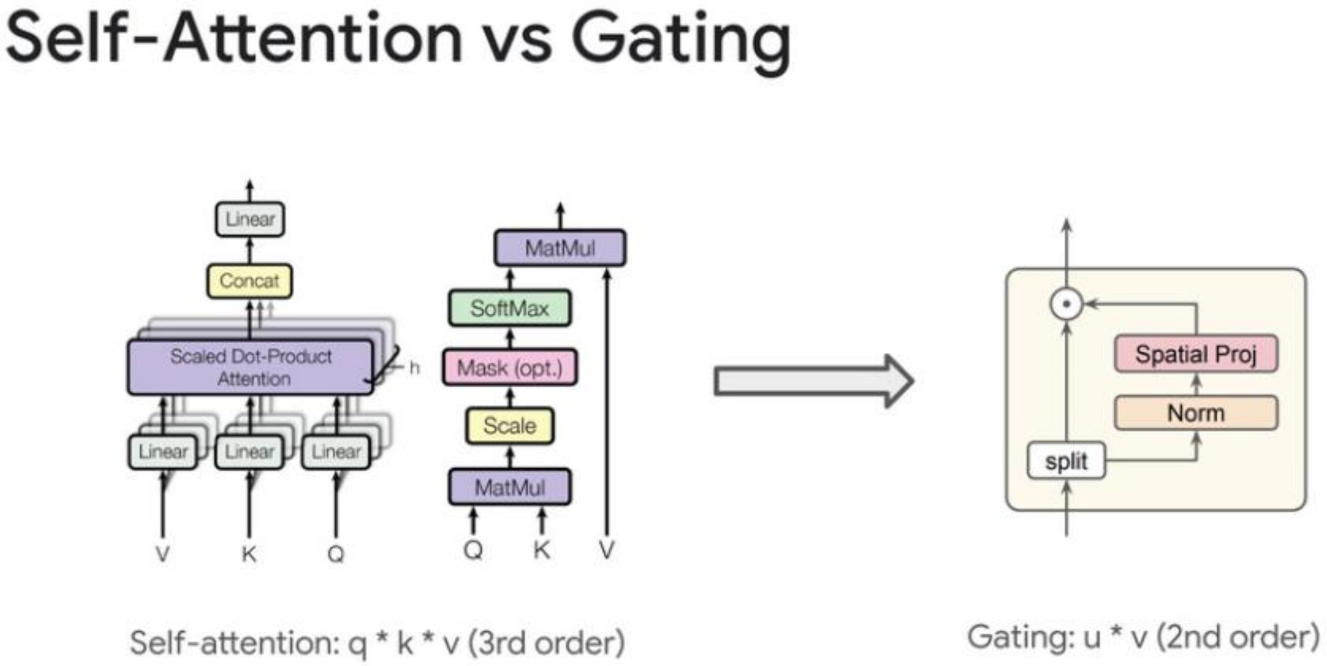
- Patches like in Vision Tramsformer (ViT)
- 2.5 times faster than ViT but comparable accuracy



https://paperswithcode.com/method/vision-transformer

Pay Attention to MLPs (Liu et al.)

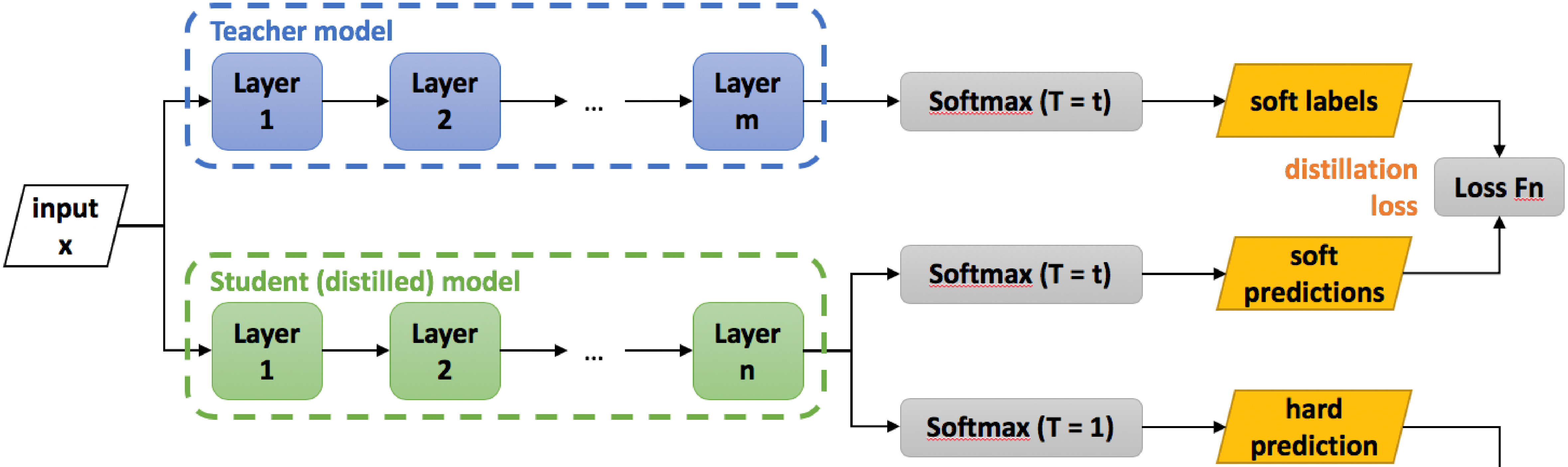
- Gated MLPs
- On par with ViT
- Claim: „self-attention is not the key for model scalability"

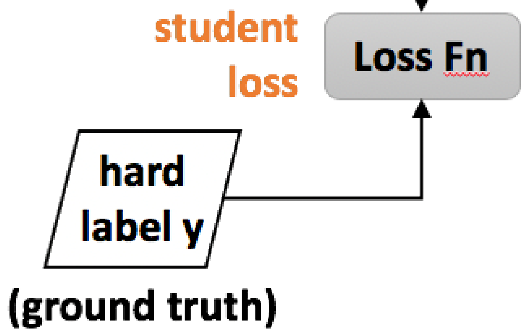


Source: poster at NeurIPS

Knowledge distillation (KD)

Architecture





$$\mathcal{L}_s := \alpha \mathcal{L}_{\mathrm{NLL}} + (1-\alpha)\mathcal{L}_{\mathrm{KD}},$$

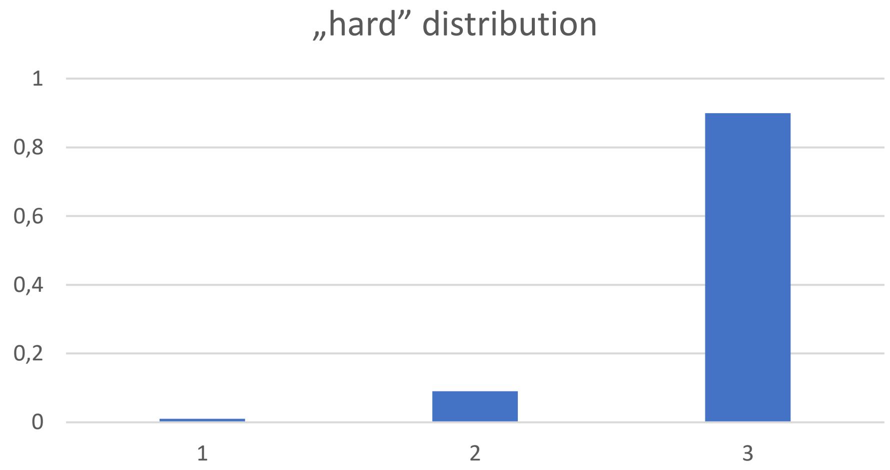
$$\mathcal{L}_{\mathrm{NLL}}(\mathbf{z}_s, \mathbf{y}) := -\sum_{j=1}^{c} y_j \log \sigma_j(\mathbf{z}_s), \quad \mathcal{L}_{\mathrm{KD}}(\mathbf{z}_s, \mathbf{z}_t) := -\tau^2 \sum_{j=1}^{c} \sigma_j\left(\frac{\mathbf{z}_t}{\tau}\right) \log \sigma_j\left(\frac{\mathbf{z}_s}{\tau}\right).$$

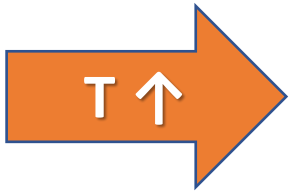
https://devopedia.org/knowledge-distillation

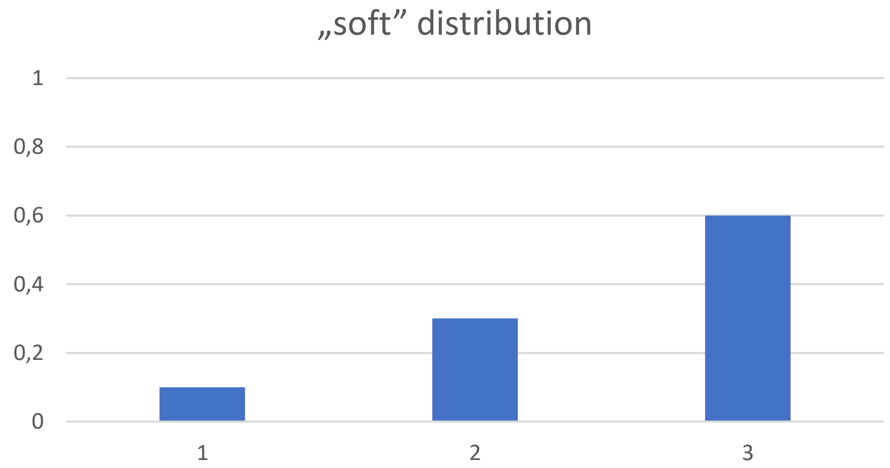
Temperature scaling

$$P_i = \frac{e^{\frac{y_i}{T}}}{\sum_{k=1}^{n} e^{\frac{y_k}{T}}}$$

T=1 → „normal softmax”







Example - RNN:

We are sampling from output distribution and choosing the sampled word as your output token (and next input). If the model is extremely confident, it may produce very repetitive and uninteresting text.

We want it to produce more diverse text which it will not produce because when the sampling procedure is going on, most of the probability mass will be concentrated in a few tokens and thus your model will keep selecting a select number of words over and over again.

In order to give other words a chance of being sampled as well, you could plug in the temperature and produce more diverse text.

https://stackoverflow.com/questions/58764619/why-should-we-use-temperature-in-softmax

Temperature scaling in KD

- „Hard” distribution doesn't provide much information beyond the ground truth labels already provided in the dataset.
- The „softer” probability provides more information as to which classes the teacher found more similar to the predicted class.
- Previous approach: using rather than the probabilities produced by the softmax as the targets for learning the small model
  - Target: minimize the squared difference between the logits produced by the cumbersome model and the logits produced by the small model

Geoffrey Hinton et al., *Distilling the Knowledge in a Neural Network*, 2015

Does Knowledge Distillation Really Work? (Stanton et al.)

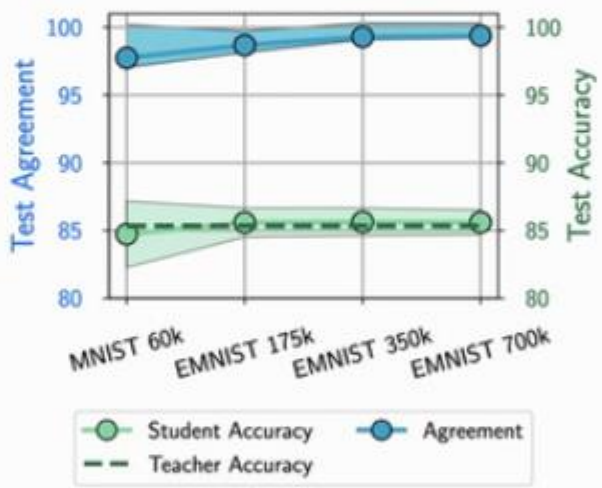
Decoupling accuracy and fidelity

Expectation: as the size of the distillation dataset increases teacher and student become functionally equivalent.

$$f_{\text{teacher}}(x) = a_0 + a_1 x + \ldots + a_n x^n \qquad f_{\text{student}}(x) = b_0 + b_1 x + \ldots + b_n x^n$$

$$f_{\text{student}}(x) = f_{\text{teacher}}(x) \ \ \forall x \in \{x_1, \ldots, x_{n+1}\} \Longrightarrow \ f_{\text{student}} \equiv f_{\text{teacher}}$$

LeNet-5 distillation on MNIST



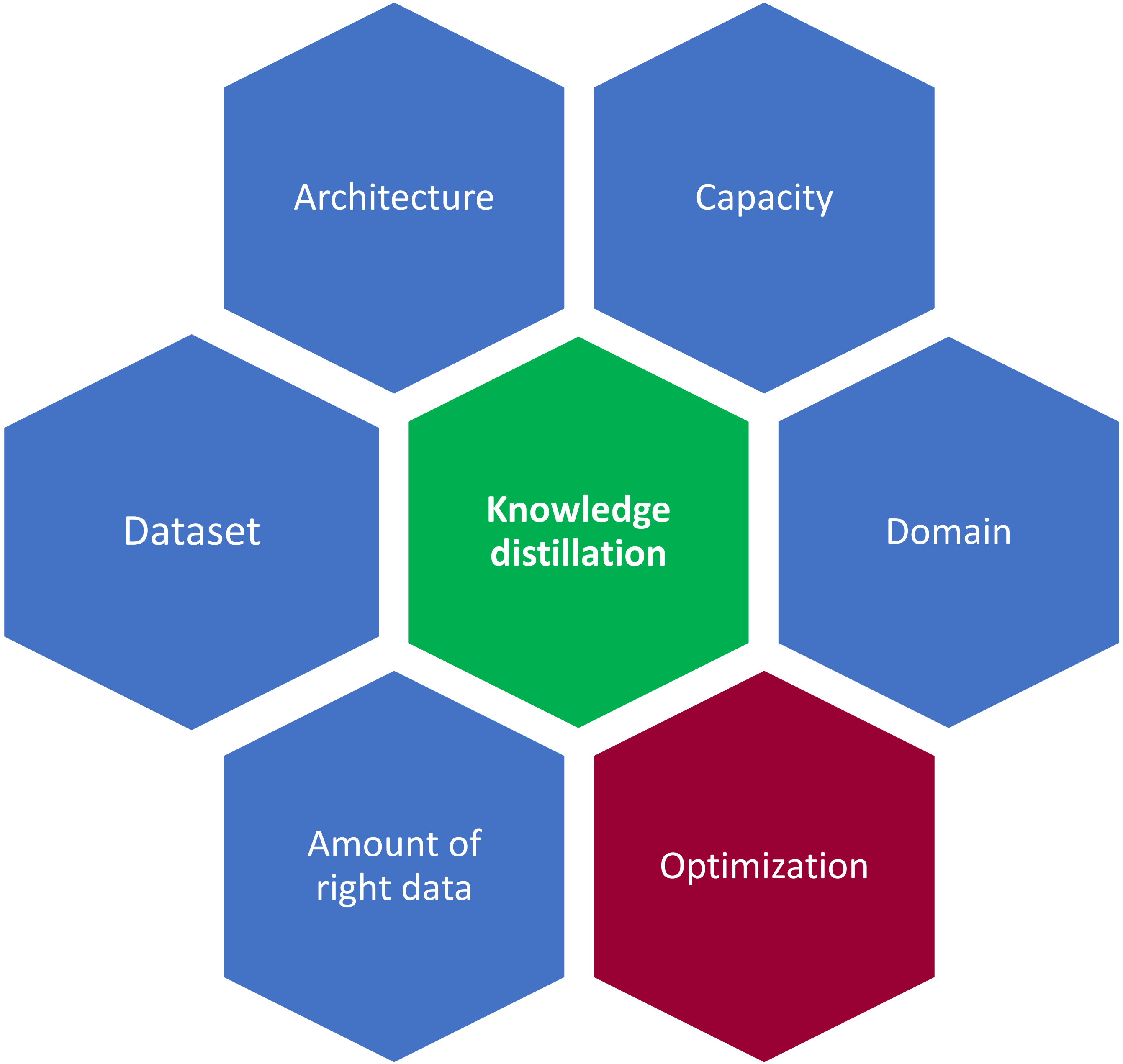
Source: poster at NeurIPS

Analogy to self-distillation

Observations

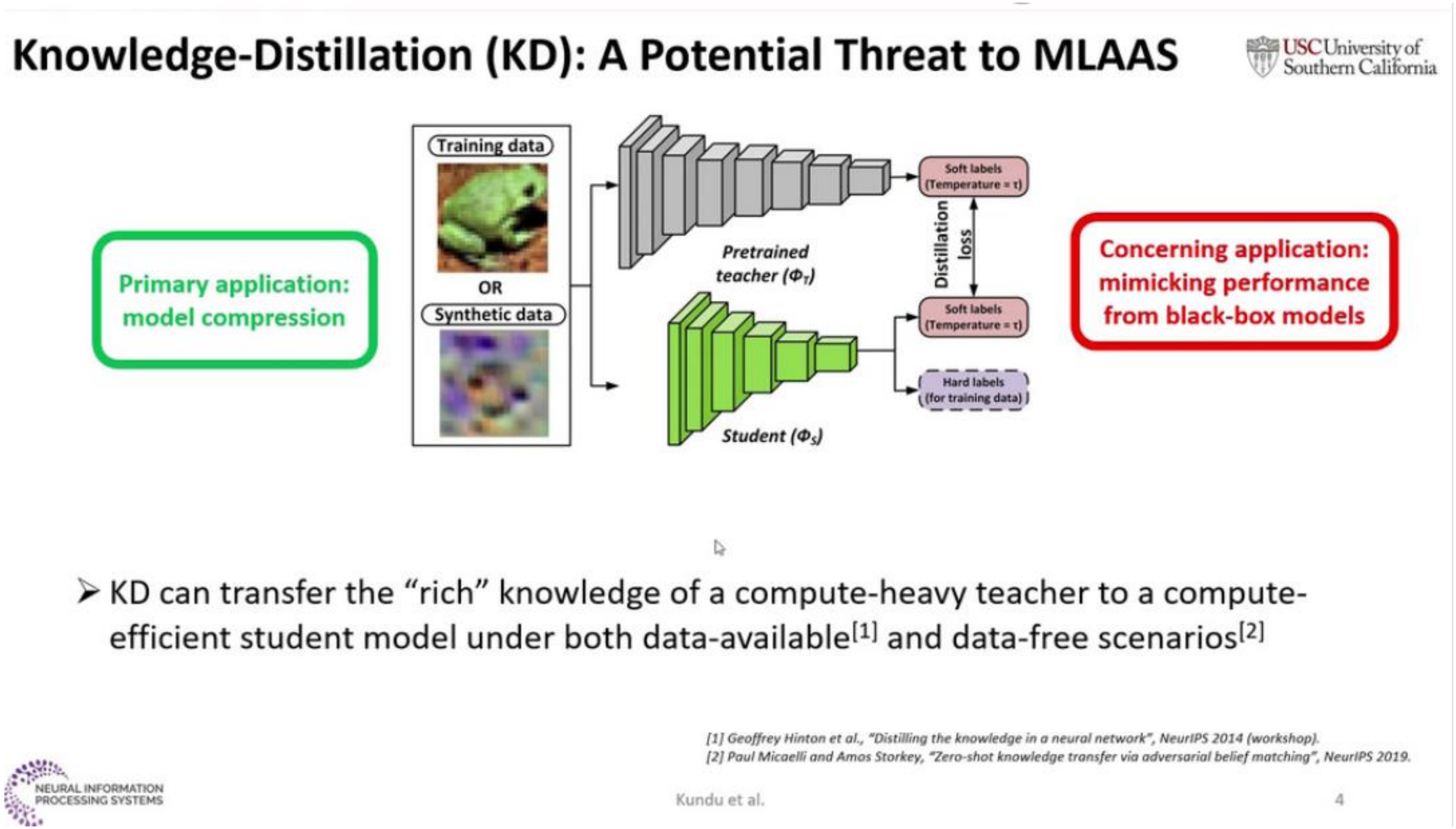
- Even in case of self-distillation, low fidelity occurs
- Increasing student depth has very little effect on fidelity

Why does knowledge distillation result in low fidelity?



Analyzing the Confidentiality of Undistillable Teachers in Knowledge Distillation (Kundu et al.)

- Motivation
  - Machine Learning as a Service is on a rise
  - Models are released as black-boxes APIs so that competitors can't replicate them



Source: poster at NeurIPS

Undistillable Models[1]

USC University of Southern California

- A class of models that
  - Perform similar to standard teacher models to maintain their own performance
  - However, act as "**nasty**" teachers to any student model by not allowing it to mimic performance.
- Core idea
  - Inject **false** sense of generalization to the student[1]

Training loss of Undistillable models ($\boldsymbol{\Phi_T}$):

$$\boxed{\mathcal{L}_N = \underbrace{\mathcal{L}_{\mathcal{CE}}(\sigma(g_{\Phi_T}(\boldsymbol{x},\boldsymbol{y})))}_{\text{Cross-entropy (CE) loss}} - \underbrace{\alpha_N * \tau_N^2 * \mathcal{L}_{\mathcal{KL}}(\sigma(g_{\Phi_T}(\boldsymbol{x},\boldsymbol{y}),\tau_N),\sigma(g_{\Phi_A}(\boldsymbol{x},\boldsymbol{y}),\tau_N))}_{\text{Self-undermining loss}}}$$

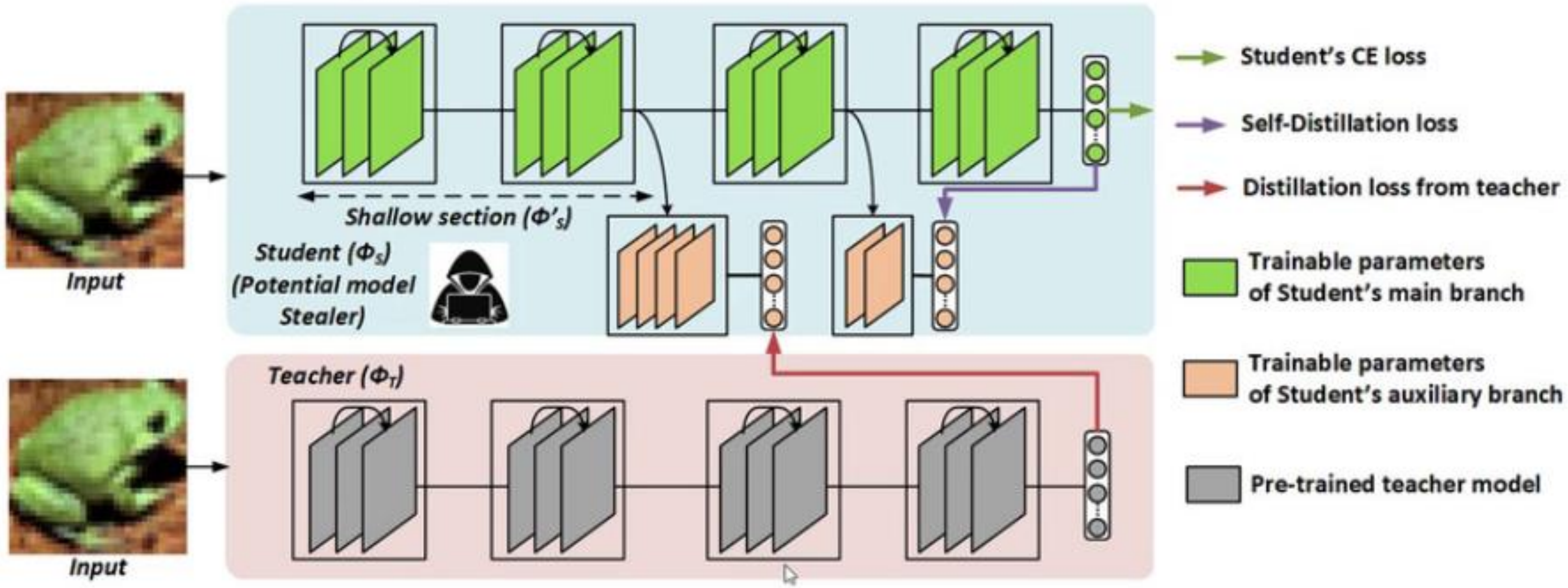
[1] Haoyu Ma et al., "Undistillable: Making a nasty teacher that cannot teach students", ICLR 2021 (spotlight).

Source: poster at NeurIPS

Proposed architecture – Skeptical Student

Motivation:

- authors observed that impact of nasty teacher on student reduces when the depth of student decreases



➢ Transfer knowledge to shallow depth ($\boldsymbol{\Phi'_S}$) of a student via aux. classifier (AC)

➢ Use self-distillation at AC in $\boldsymbol{\Phi_S}$ - $\boldsymbol{\Phi'_S}$ to boost performance of student $\boldsymbol{\Phi_S}$

NEURAL INFORMATION PROCESSING SYSTEMS

Kundu et al.

Source: poster at NeurIPS

Properties

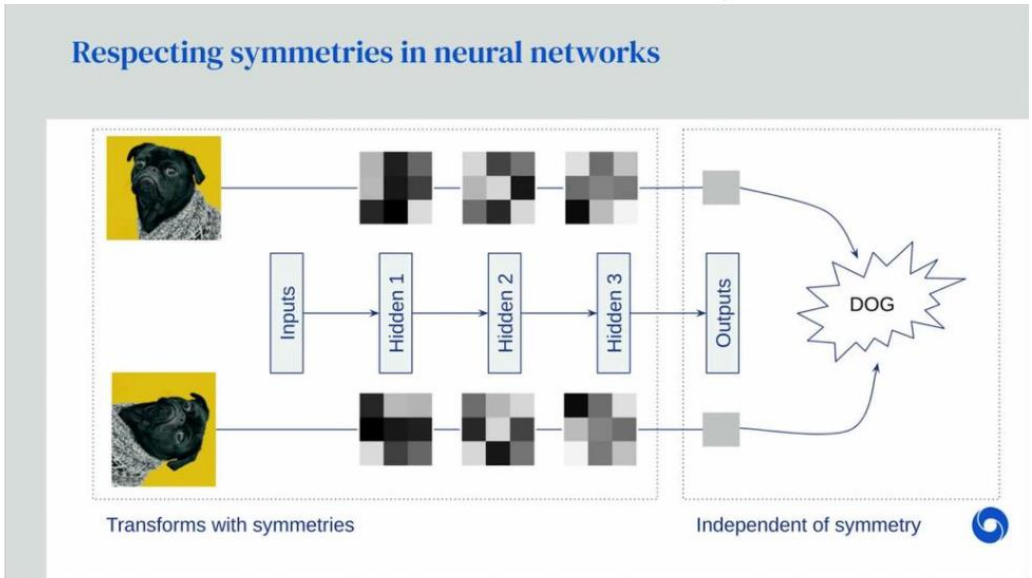
Skeptical students achieve similar to teacher performance even when the teacher is Undistillable (or nasty).

Skeptical students achieve similar to normal students' performance upon distillation from a normal teacher.

Source: poster at NeurIPS

Highlights

- Equivariant/invariant layers
- Data-centric AI
- Demonstrations
- Different approach to Continaul Learing



Source: posters at NeurIPS

Thank you for attention