Deep Learning
lecture 12
Beyond Supervised Learning

Yi Wu, IIIS

Spring 2025

May-10

Today’s Topic

- Unsupervised Learning and Self-supervised Learning
- Learning to Efficiently Learn Neural Networks
  - Aka. Meta-Learning, Learning to Learn
- Reinforcement Learning and Human-AI Collaboration
  - Some interesting projects from Prof. Wu’s group

Today's Topic

- **Unsupervised Learning and Self-supervised Learning**

- Learning to Efficiently Learn Neural Networks
  - Aka. Meta-Learning, Learning to Learn

- Reinforcement Learning and Human-AI Collaboration
  - Some interesting projects from Prof. Wu's group

Supervised Learning

- The Core Idea of Deep Learning
  - Use non-linear function approximators to represent high-dimensional data
  - $Y = f(X;\theta)$

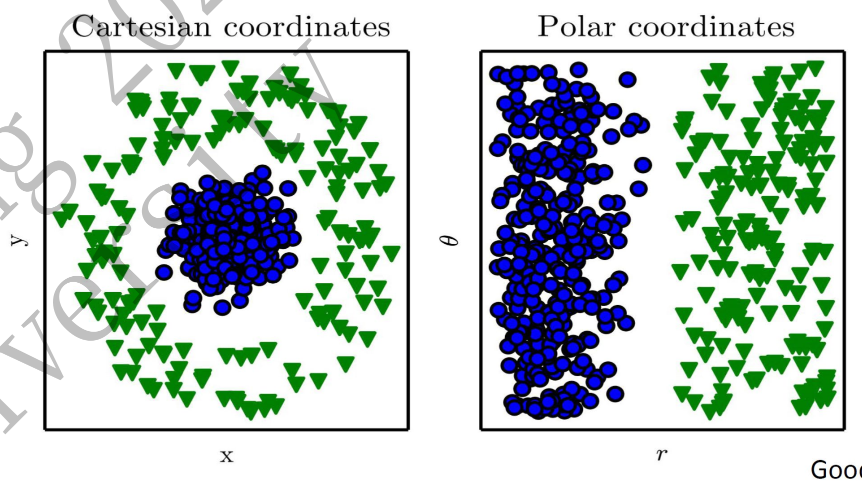
- We need a LOT of labeled data!
  - But where are the labels from?
  - Human efforts!

Labeled Data

Unlabeled Data

- What if we do NOT have labels?

Representation Learning



Goodfellow

- Representations Matter!
    - Deep learning is a tool to learn representations
- Can we learn useful representations from unlabeled data?
    - We have tremendous (unlabeled) data!
        - Internet, videos, texts, audio
    - The representations can be used for supervised tasks

Representation Learning

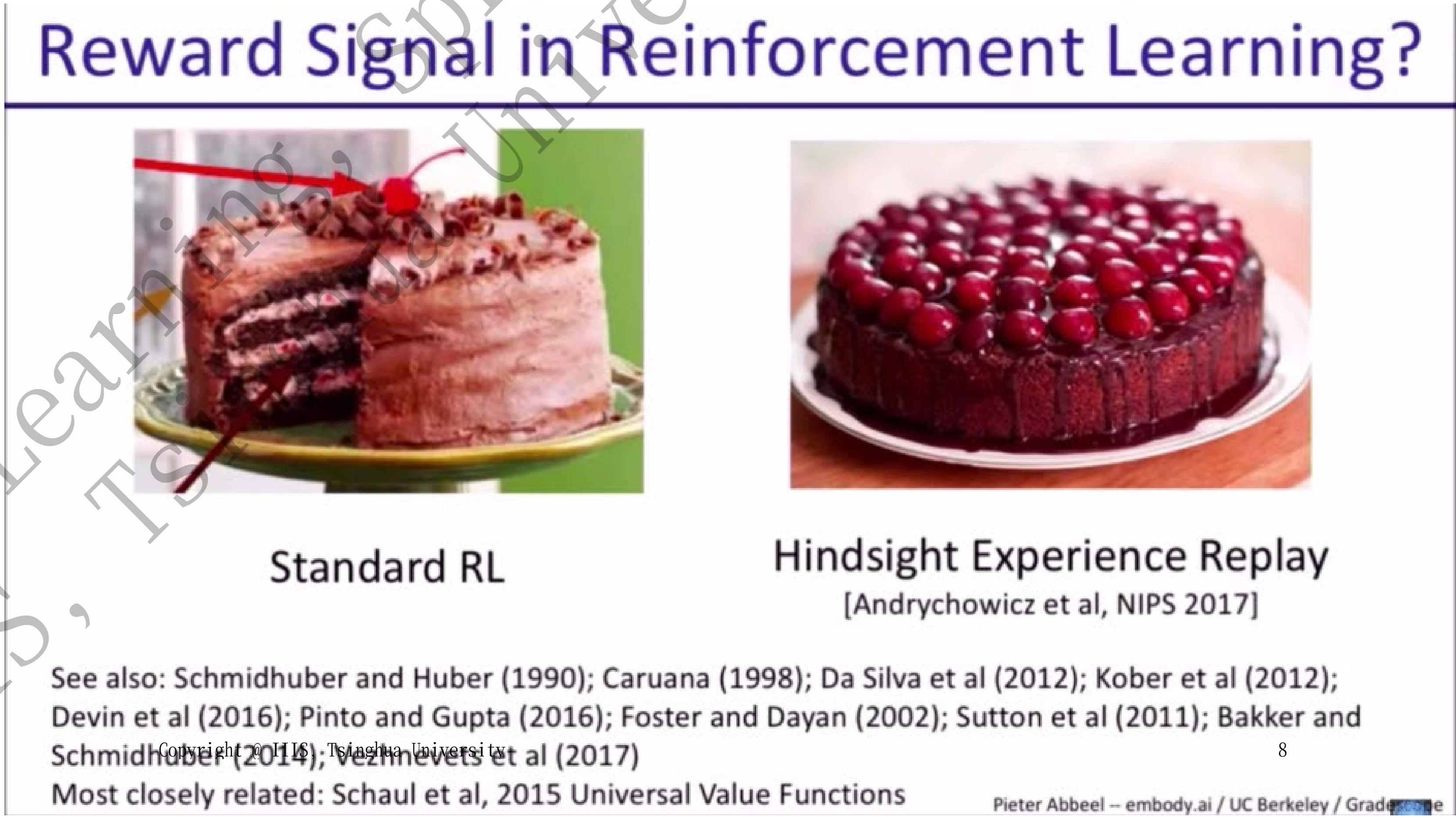
- Generative Model
  - Learn a probabilistic model $P(X;\theta)$ to fit $P_{data}(X)$ via samples
    - EBM, Flow Model, VAE, GAN, Autoregressive model
    - Sampling: $z \rightarrow X$
  - Representations: $X \rightarrow z$
    - Latent embeddings of X
    - VAE; Autoregressive model; BiGAN; EBM; Flow
  - Generative models *implicitly* learn representations
    - GM are also harder to train (v.s. supervised learning)
  - *Can we run unsupervised learning in a discriminative way?*

How Much Information is the Machine Given during Learning?

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

The Cherry Cake

- Definitely some people disagree with Yann ☺
  - Some covered in lecture 10 (RL+LLM)
  - More in DRL course

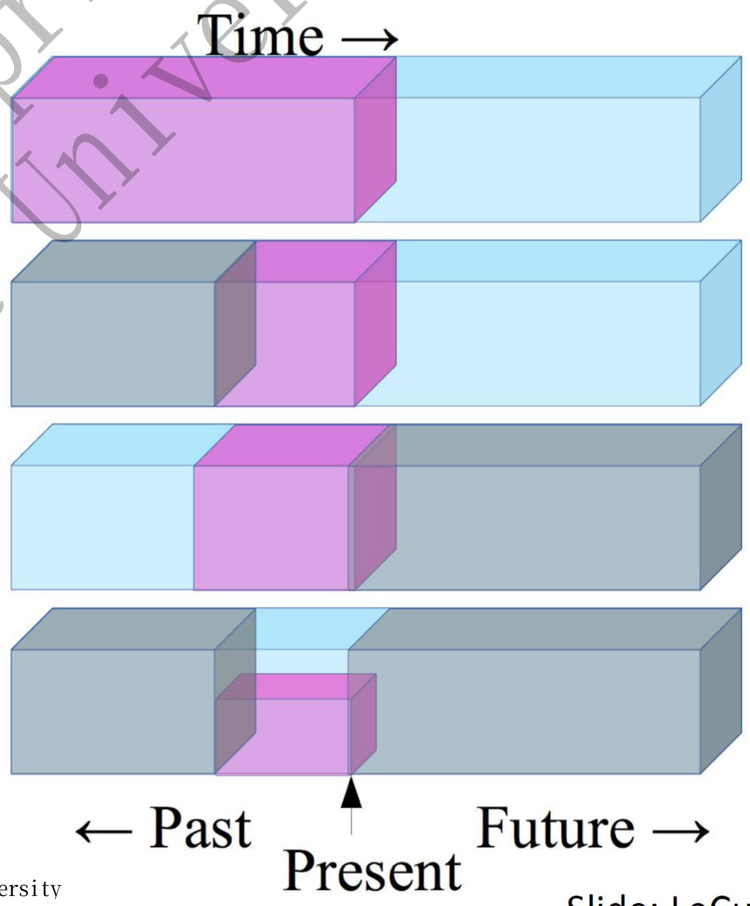


Self-Supervised Learning for Representations

- Self-Supervised Learning
  - Goal: learning representations $z = f(x;\theta)$
    - No labels; unsupervised learning from data
    - *Still want to Learn in a **discriminative** fashion (no density)*
  - Create a virtual supervised learning tasks!
    - Use part of X as virtual "labels"
      - Create a random mask M on X
      - $X \cdot (1 - M) = f(X \cdot M;\theta)$
    - Learning features via supervised training
      - Idea: SL captures generic features for down-stream tasks
      - Fine-tuning for new supervised tasks

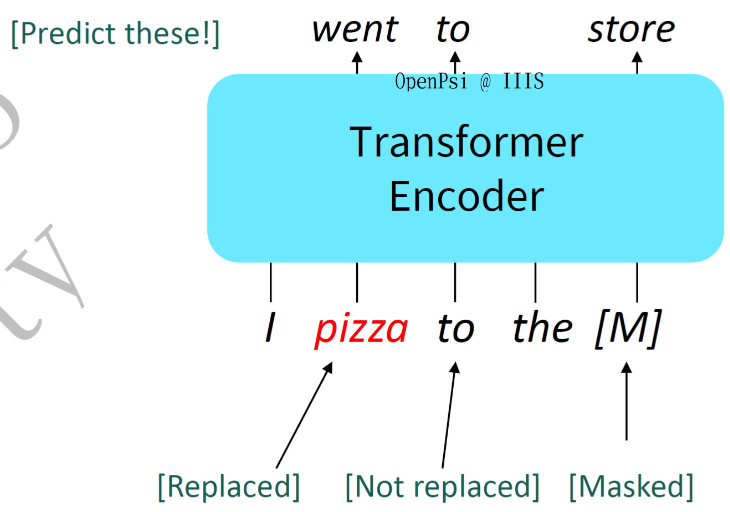
Self-Supervised Learning

- Prediction-Based Self-Supervised Learning
  - Predict any part of the input from any other part.
  - Predict the future from the past.
  - Predict the future from the recent past.
  - Predict the past from the present.
  - Predict the top from the bottom.
  - Predict the occluded from the visible
  - Pretend there is a part of the input you don't know and predict that.

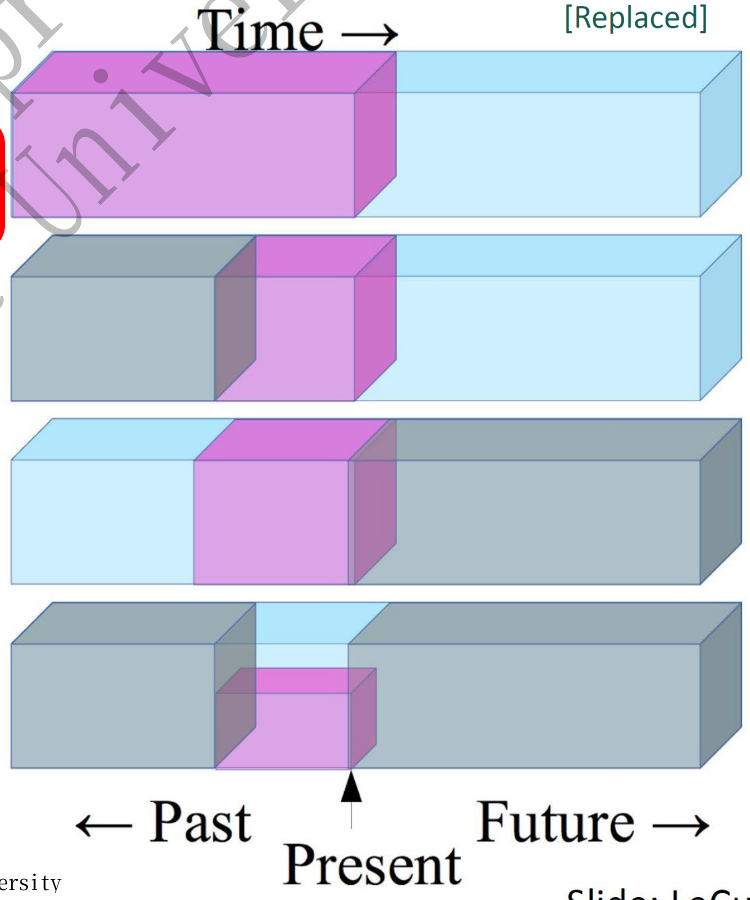


Slide: LeCun

Self-Supervised Learning



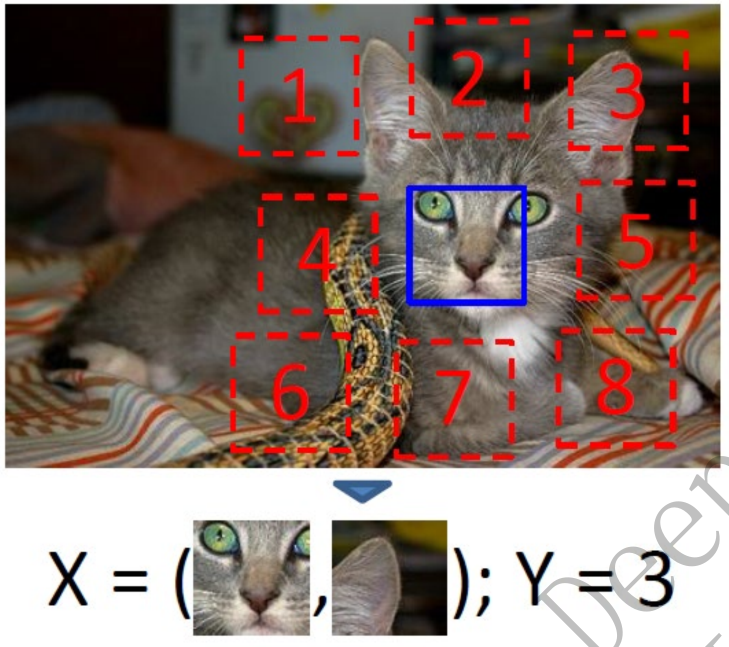
- Prediction-Based Self-Supervised Learning
  - Predict any part of the input from any other part.
  - LM: Predict the future from the past.
  - Predict the future from the recent past.
  - Predict the past from the present.
  - Predict the top from the bottom.
  - BERT: Predict the occluded from the visible
  - Pretend there is a part of the input you don't know and predict that.



Slide: LeCun

Self-Supervised Learning

- Context Prediction (Pathak et al., ICCV 2015)



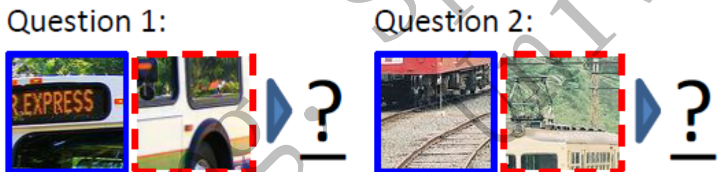
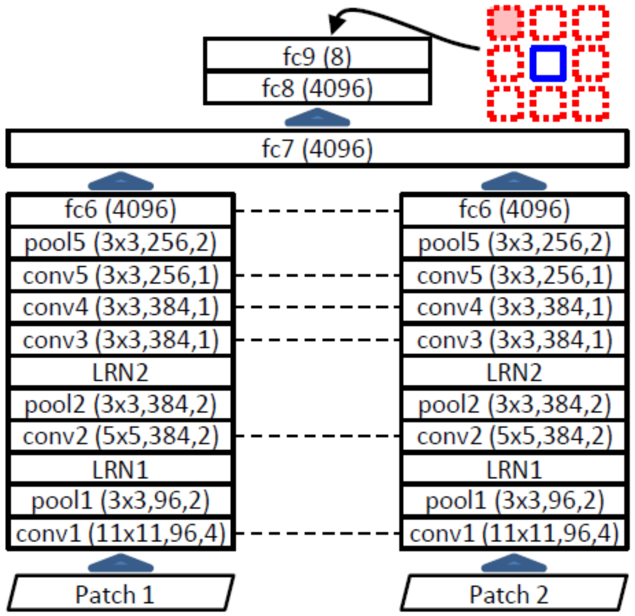
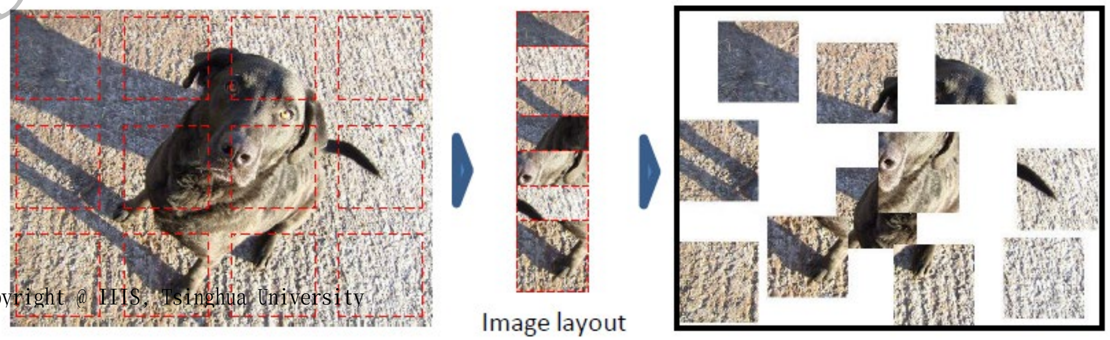


Figure 1. Our task for learning patch representations involves randomly sampling a patch (blue) and then one of eight possible neighbors (red). Can you guess the spatial configuration for the two pairs of patches? Note that the task is much easier once you have recognized the object!

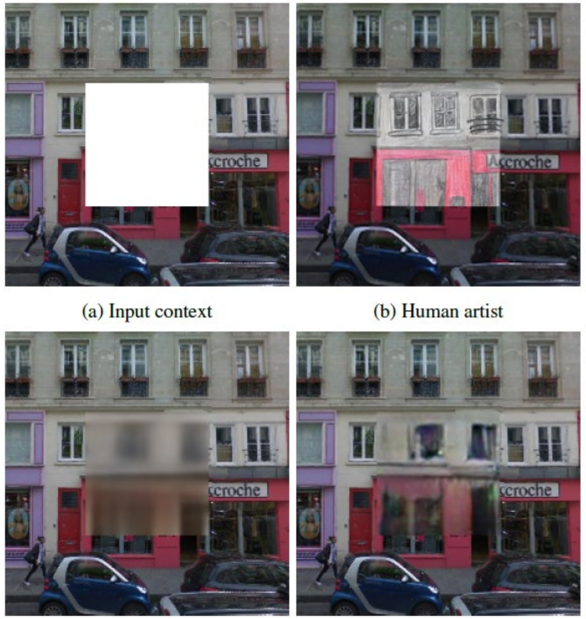
Answer key: Q1: Bottom right Q2: Top center





Self-Supervised Learning

- Feature Learning by Inpainting (Pathak et al, CVPR 2016)
  - AE + random mask + GAN loss



(a) Input context (b) Human artist

(c) Context Encoder ($L2$ loss) (d) Context Encoder ($L2$ + Adversarial loss)

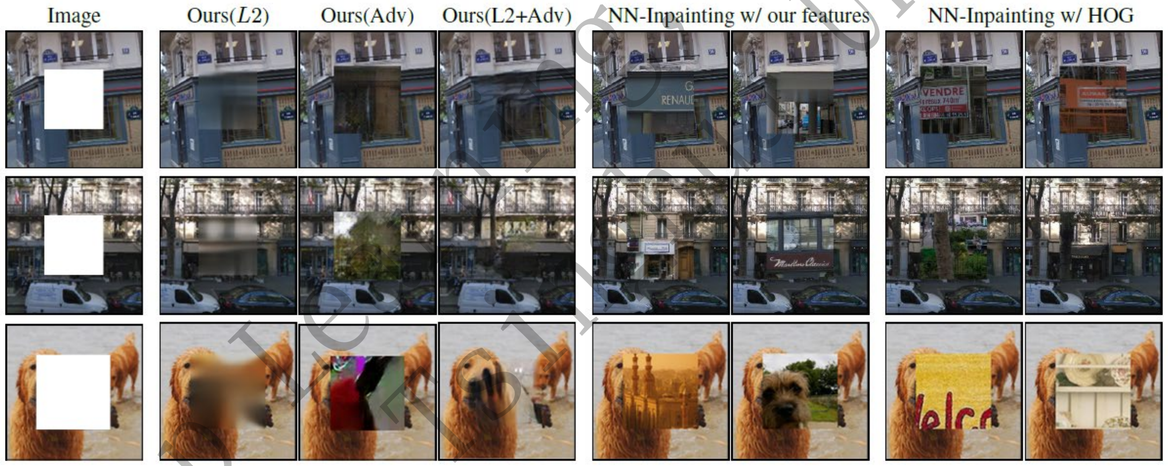


Figure 6: Semantic Inpainting using different methods on *held-out* images. Context Encoder with just L2 are well aligned, but not sharp. Using adversarial loss, results are sharp but not coherent. Joint loss alleviate the weaknesses of each of them. The last two columns are the results if we plug-in the best nearest neighbor (NN) patch in the masked region.

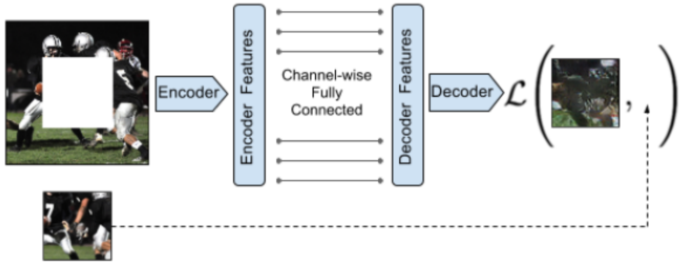
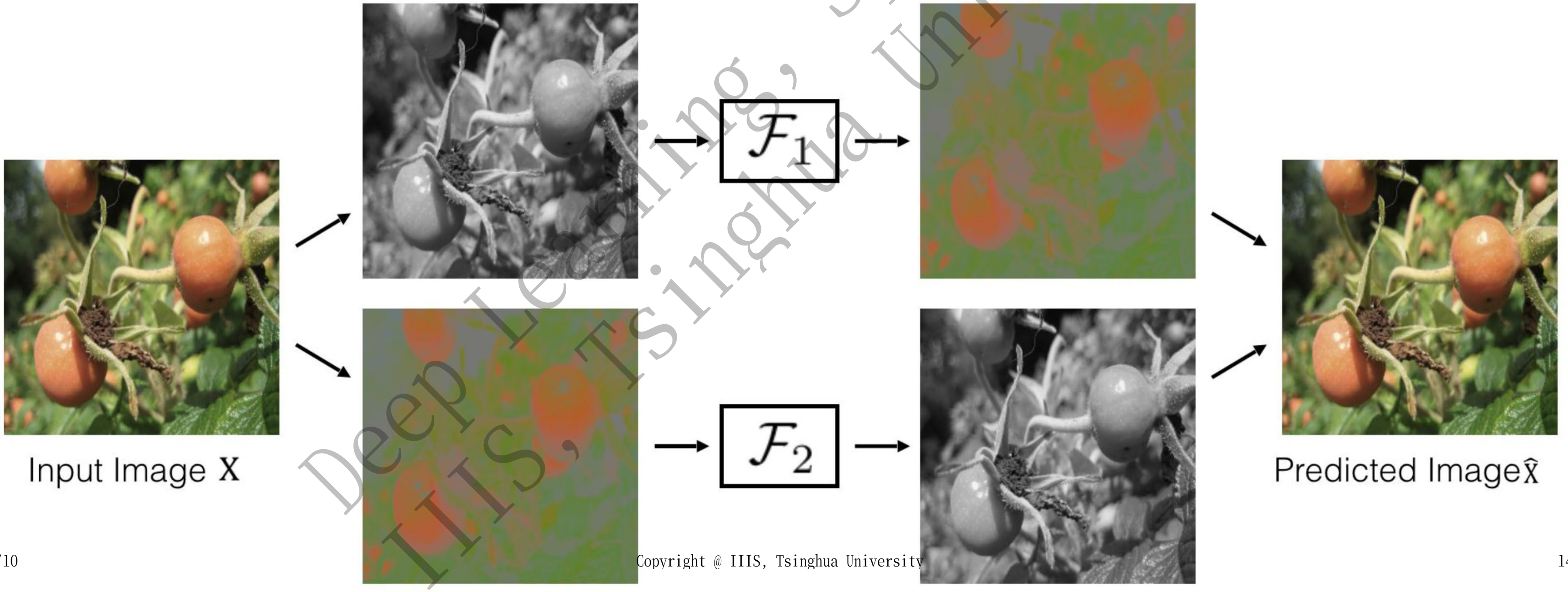


Figure 2: Context Encoder. The context image is passed through the encoder to obtain features which are connected to the decoder using channel-wise fully-connected layer as described in Section 3.1. The decoder then produces the missing regions in the image.

Self-Supervised Learning

- Image Colorization (Richard Zhang, et al, ECCV 2016 and more)



Self-Supervised Learning

- Rotation Prediction (Gidaris et al, ICLR 2018)

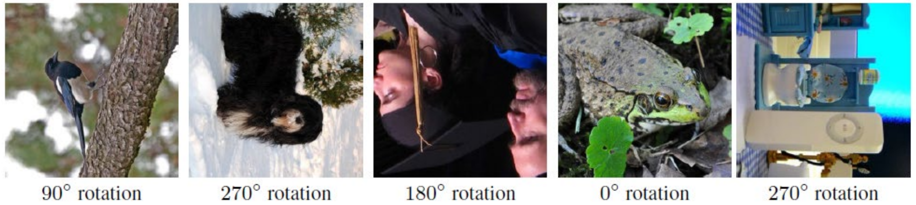
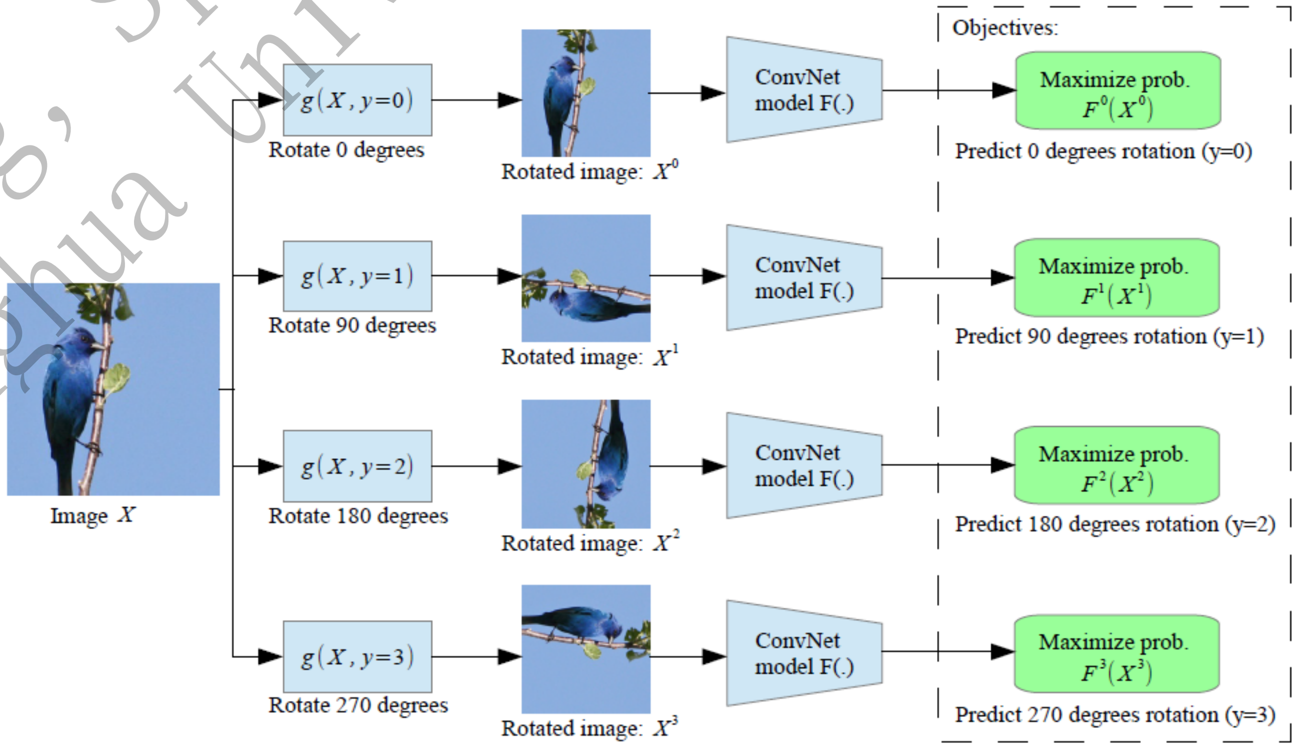


Figure 1: Images rotated by random multiples of 90 degrees (e.g., 0, 90, 180, or 270 degrees). The core intuition of our self-supervised feature learning approach is that if someone is not aware of the concepts of the objects depicted in the images, he cannot recognize the rotation that was applied to them.

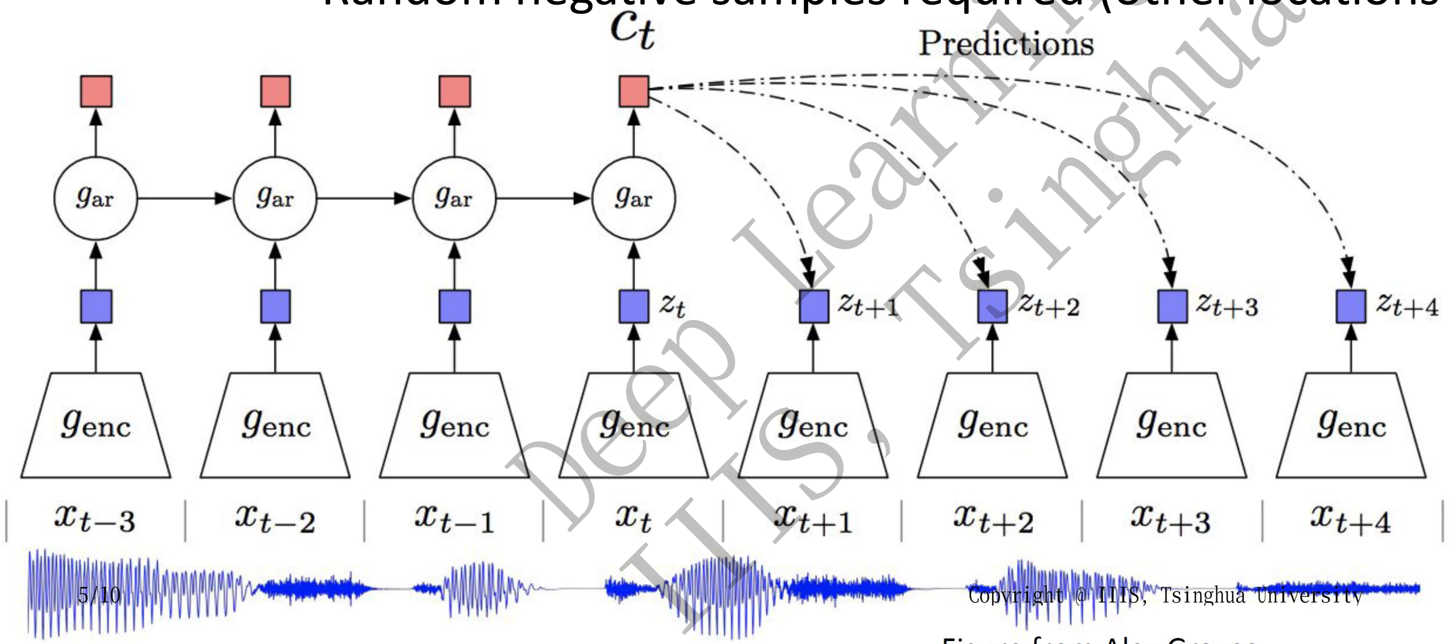
(a) **Attention maps of supervised model**

(b) **Attention maps of our self-supervised model**



Self-Supervised Learning

- Contrastive Predictive Coding (Van den Oord et al, DeepMind, 2018)
  - CPC: Originally proposed on audio data
  - Use context to predict future embeddings
    - Use contrastive loss (avoid trivial solutions)
    - Random negative samples required (other locations / other samples in each mini-batch)



$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

$$\mathcal{L}_N = -\mathop{\mathbb{E}}_{X}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$$

Figure from Alex Graves

Self-Supervised Learning

- Contrastive Predictive Coding (Van den Oord et al, DeepMind, 2018)

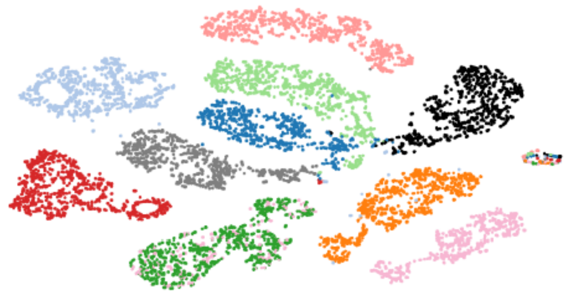


Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

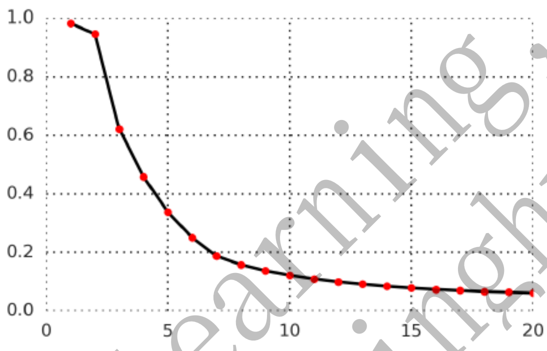


Figure 3: Average accuracy of predicting the positive sample in the contrastive loss for 1 to 20 latent steps in the future of a speech waveform. The model predicts up to 200ms in the future as every step consists of 10ms of audio.

Method	ACC
Phone classification	
Random initialization	27.6
MFCC features	39.7
CPC	64.6
Supervised	74.6
Speaker classification	
Random initialization	1.87
MFCC features	17.6
CPC	97.4
Supervised	98.5

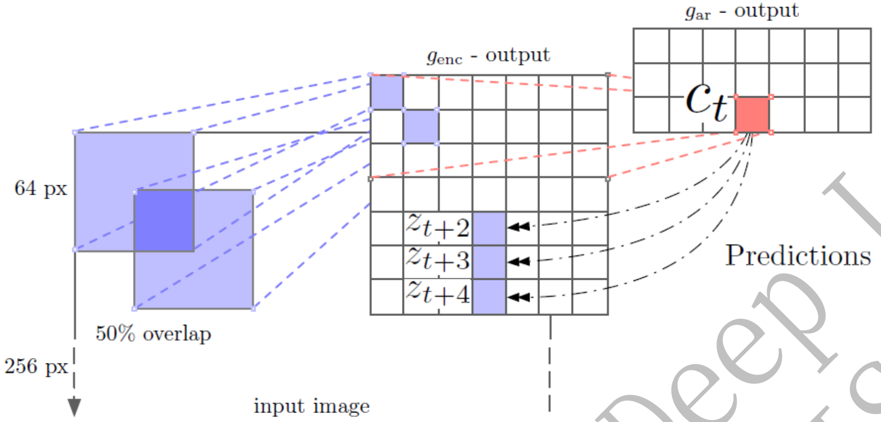
Table 1: LibriSpeech phone and speaker classification results. For phone classification there are 41 possible classes and for speaker classification 251. All models used the same architecture and the same audio input sizes.

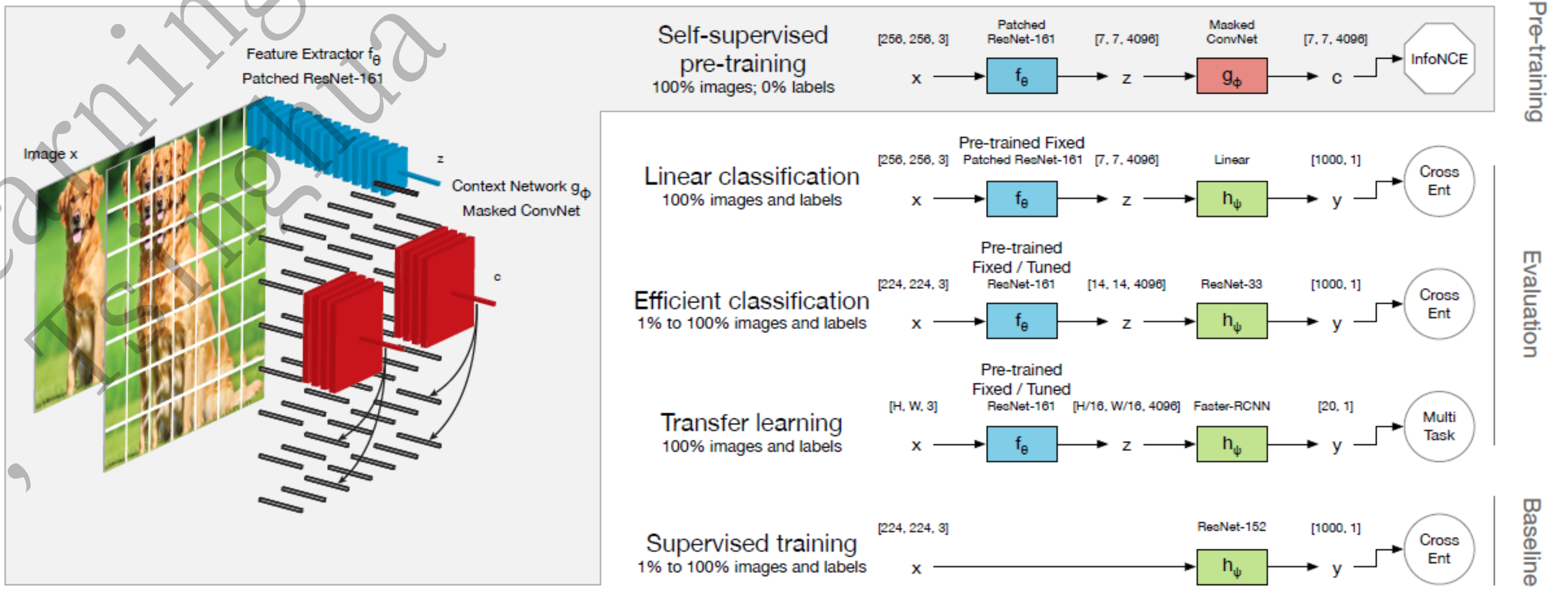
Method	ACC
#steps predicted	
2 steps	28.5
4 steps	57.6
8 steps	63.6
12 steps	64.6
16 steps	63.8
Negative samples from	
Mixed speaker	64.6
Same speaker	65.5
Mixed speaker (excl.)	57.3
Same speaker (excl.)	64.6
Current sequence only	65.2

Table 2: LibriSpeech phone classification ablation experiments. More details can be found in Section 3.1.

Self-Supervised Learning

- Contrastive Predictive Coding (Van den Oord et al, ICML 2020)
  - CPCv2: improved version of CPC on images with large scale training
    - Basic-version CPC on images: PixelCNN (masked convolution)
    - Divide an image into patches; For each context c_t, predict "future" patches below it





Self-Supervised Learning

- Contrastive Predictive Coding (Van den Oord et al, ICML 2020)
  - CPCv2: improved version of CPC on images with large scale training
    - Enhancements: Large-scale training; layer normalization; prediction in 4 directions; more prediction directions; patch-based data augmentations

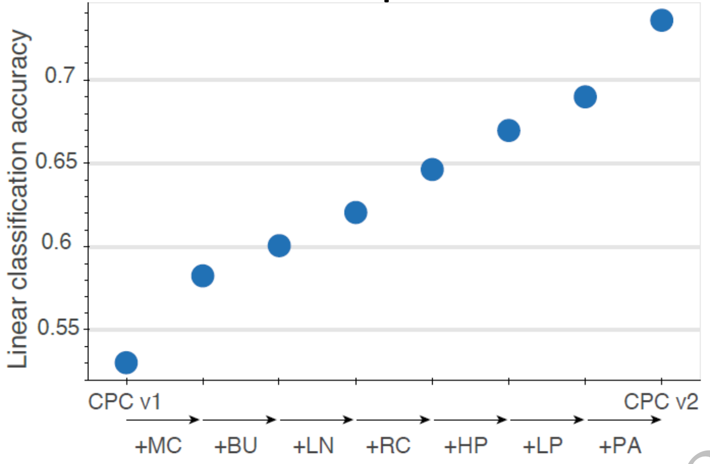
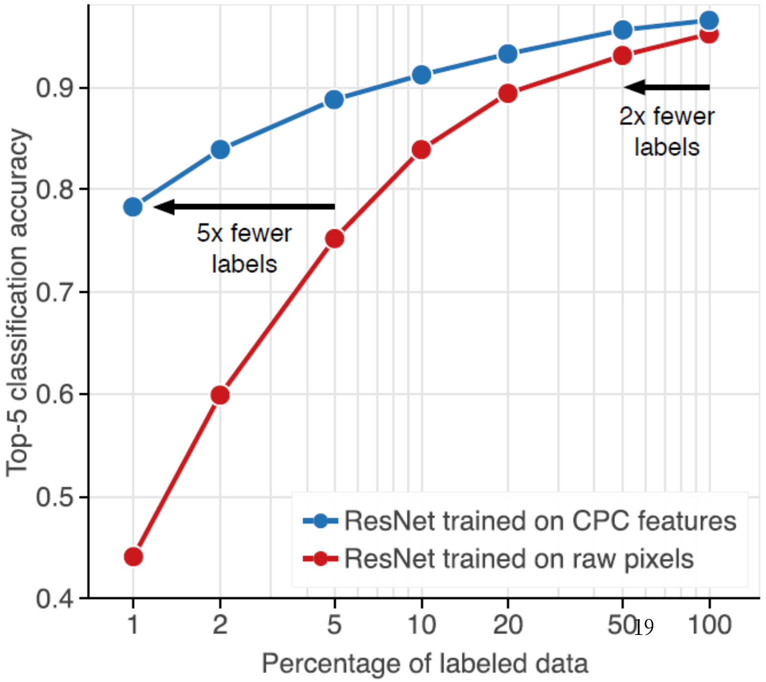


Figure 3. Linear classification performance of new variants of CPC, which incrementally add a series of modifications. MC: model capacity. BU: bottom-up spatial predictions. LN: layer normalization. RC: random color-dropping. HP: horizontal spatial predictions. LP: larger patches. PA: further patch-based augmentation. Note that these accuracies are evaluated on a custom validation set and are therefore not directly comparable to the results we report on the official validation set.

METHOD	PARAMS (M)	TOP-1	TOP-5
Methods using ResNet-50:			
INSTANCE DISCR. [1]	24	54.0	-
LOCAL AGGR. [2]	24	58.8	-
MOCO [3]	24	60.6	-
PIRL [4]	24	63.6	-
CPC V2 - RESNET-50	24	**63.8**	**85.3**
Methods using different architectures:			
MULTI-TASK [5]	28	-	69.3
ROTATION [6]	86	55.4	-
CPC V1 [7]	28	48.7	73.6
BIGBIGAN [8]	86	61.3	81.9
AMDIM [9]	626	68.1	-
CMC [10]	188	68.4	88.2
MOCO [2]	375	68.6	-
CPC V2 - RESNET-161	305	**71.5**	**90.1**



Self-Supervised Learning

- MoCo: Momentum Contrastive Learning (Kaiming et al, CVPR 2020)
  - Get negative samples directly from a buffer (fast negative sampling)
  - Two encoders: f_{θ_q} for query; f_{θ_k} for keys; store key samples in a queue
  - SGD for θ_q; θ_k is updated using ***exponential moving average*** (momentum)

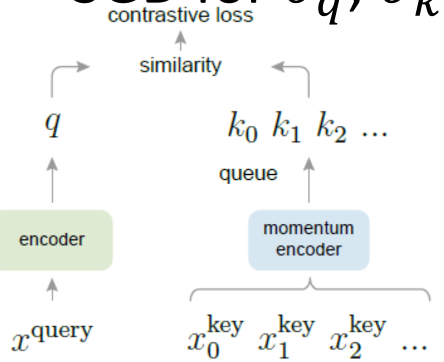
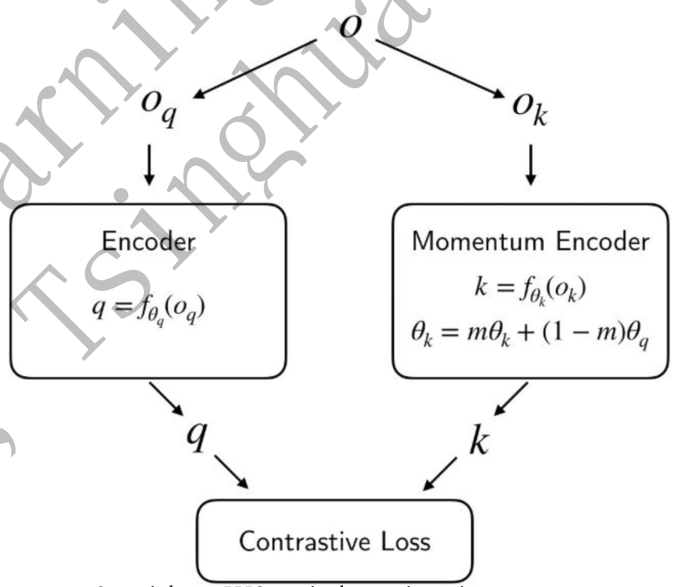


Figure 1. Momentum Contrast (MoCo) trains a visual representation encoder by matching an encoded query q to a dictionary of encoded keys using a contrastive loss. The dictionary keys $\{k_0, k_1, k_2, ...\}$ are defined on-the-fly by a set of data samples. The dictionary is built as a queue, with the current mini-batch enqueued and the oldest mini-batch dequeued, decoupling it from the mini-batch size. The keys are encoded by a slowly progressing encoder, driven by a momentum update with the query encoder. This method enables a large and consistent dictionary for learning visual representations.



$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)}$$

Self-Supervised Learning

- MoCo: Momentum Contrastive Learning (Kaiming et al, CVPR 2020)
  - Why momentum encoder?
    - Ensure the encodings in buffer ***moves slowly*** via momentum
    - Enable a ***consistent*** buffer of negative samples (no need to recompute features)
      - This further ensures the feature extractor updates smoothly

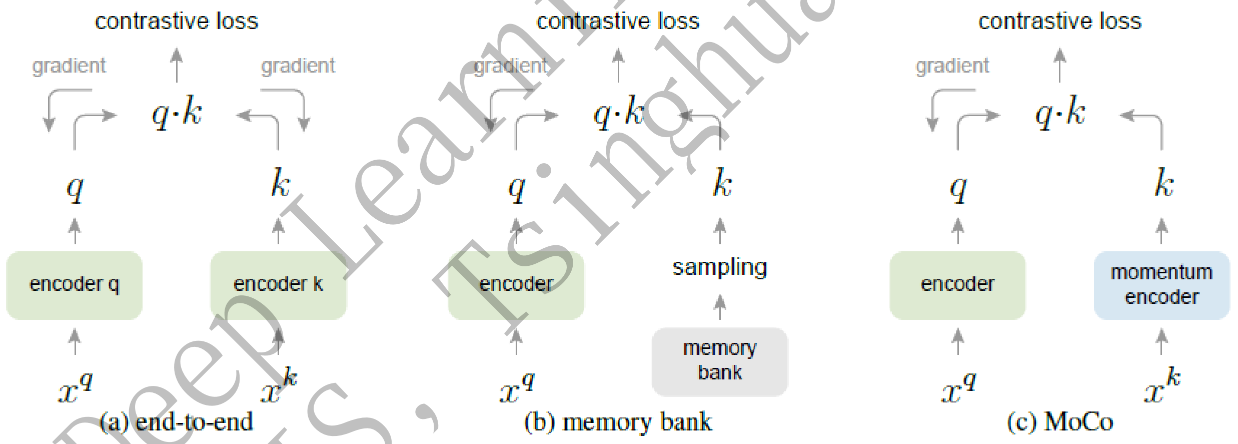


Figure 2. **Conceptual comparison of three contrastive loss mechanisms** (empirical comparisons are in Figure 3 and Table 3). Here we illustrate one pair of query and key. The three mechanisms differ in how the keys are maintained and how the key encoder is updated. **(a)**: The encoders for computing the query and key representations are updated ***end-to-end*** by back-propagation (the two encoders can be different). **(b)**: The key representations are sampled from a ***memory bank*** [61]. **(c)**: ***MoCo*** encodes the new keys on-the-fly by a momentum-updated encoder, and maintains a queue (not illustrated in this figure) of keys.

Self-Supervised Learning

- MoCo: Momentum Contrastive Learning (Kaiming et al, CVPR 2020)

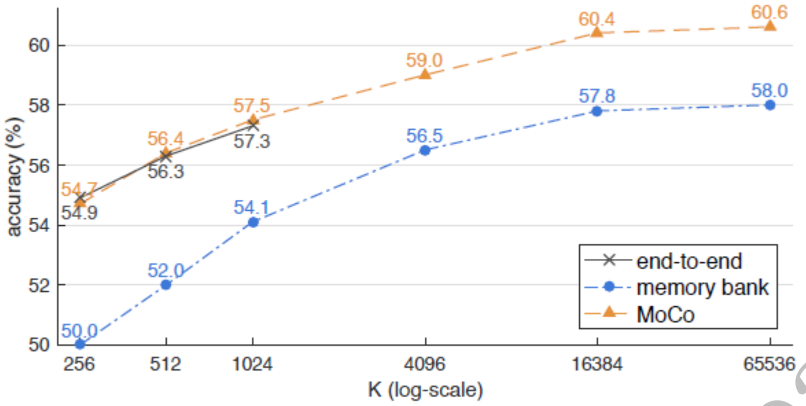
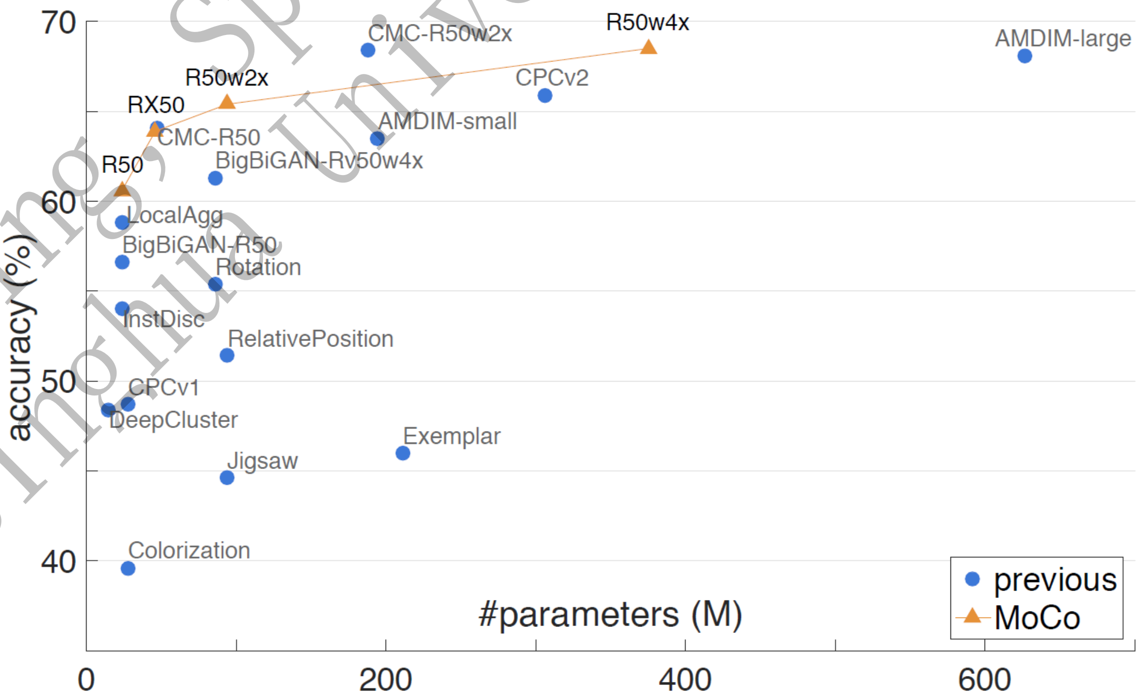
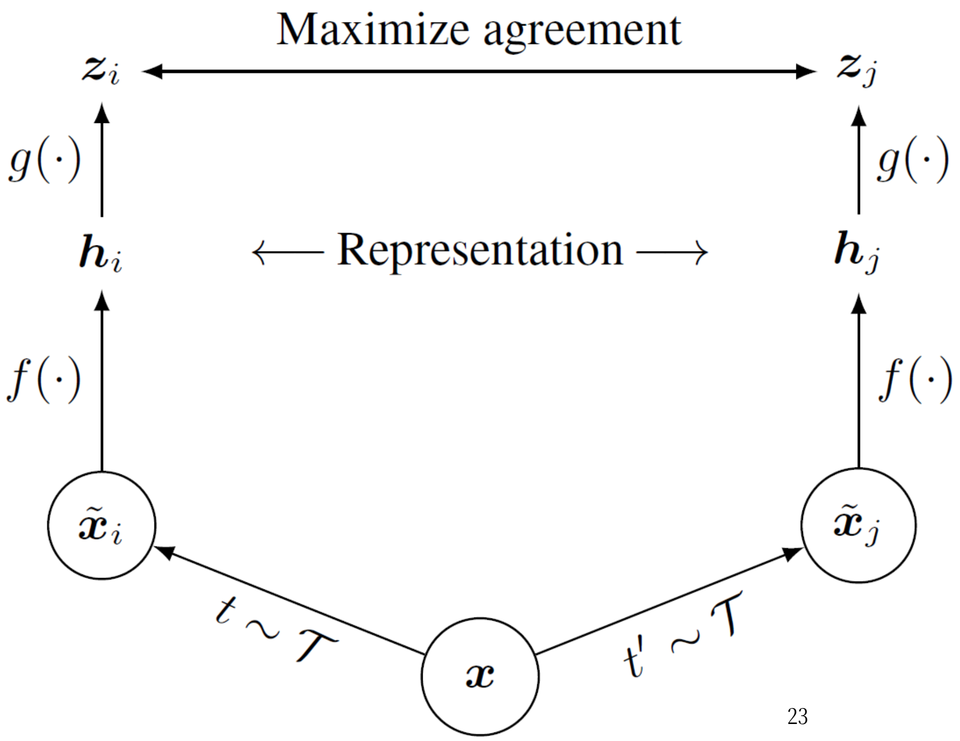


Figure 3. **Comparison of three contrastive loss mechanisms** under the ImageNet linear classification protocol. We adopt the same pretext task (Sec. 3.3) and only vary the contrastive loss mechanism (Figure 2). The number of negatives is K in memory bank and MoCo, and is $K-1$ in end-to-end (offset by one because the positive key is in the same mini-batch). The network is ResNet-50.



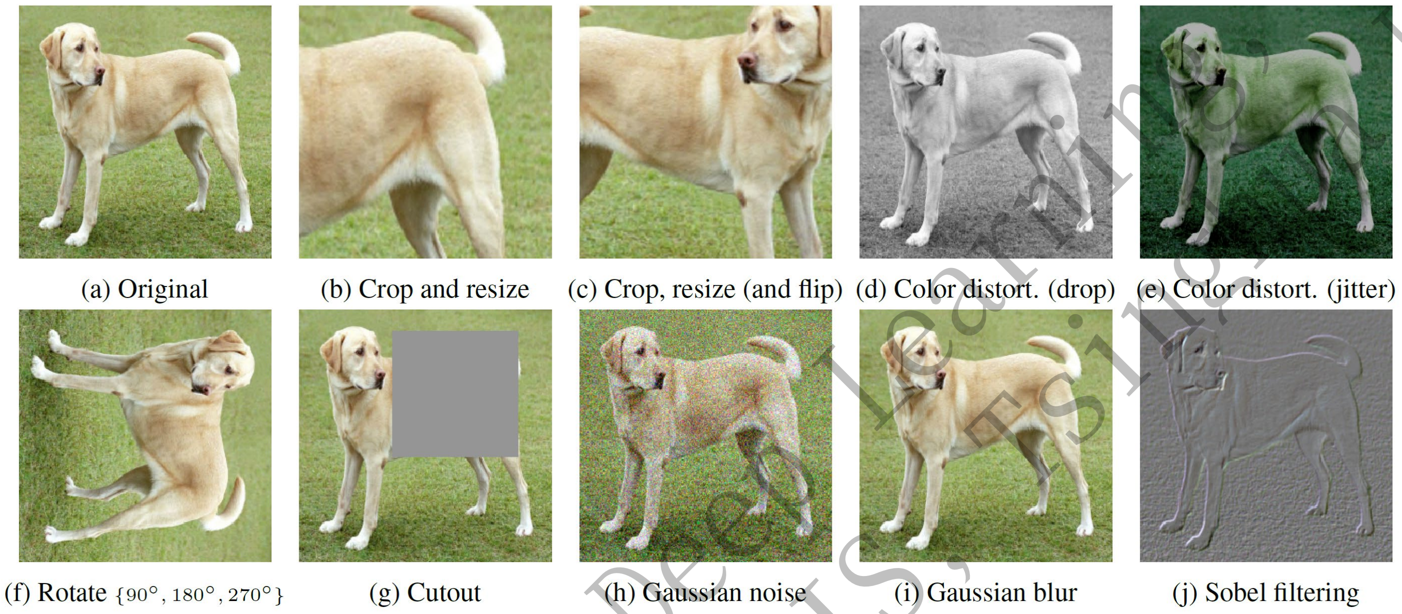
Self-Supervised Learning

- SimCLR (Chen et al, Hinton's group, ICML 2020)
  - A Simple Framework for Contrastive Learning of Visual Representations
    - Predefine a set of transformations
    - For a data, sample two transformations
    - Maximum agreement on representations
  - No explicit negative data sampling
    - Non-paired data in the batch are negative ones
    - For each z_i, and every unpaired z_k in the minibatch
      - Minimize the agreement



Self-Supervised Learning

- SimCLR (Chen et al, Hinton's group, ICM
  - A Simple Framework for Contrastive Lear



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

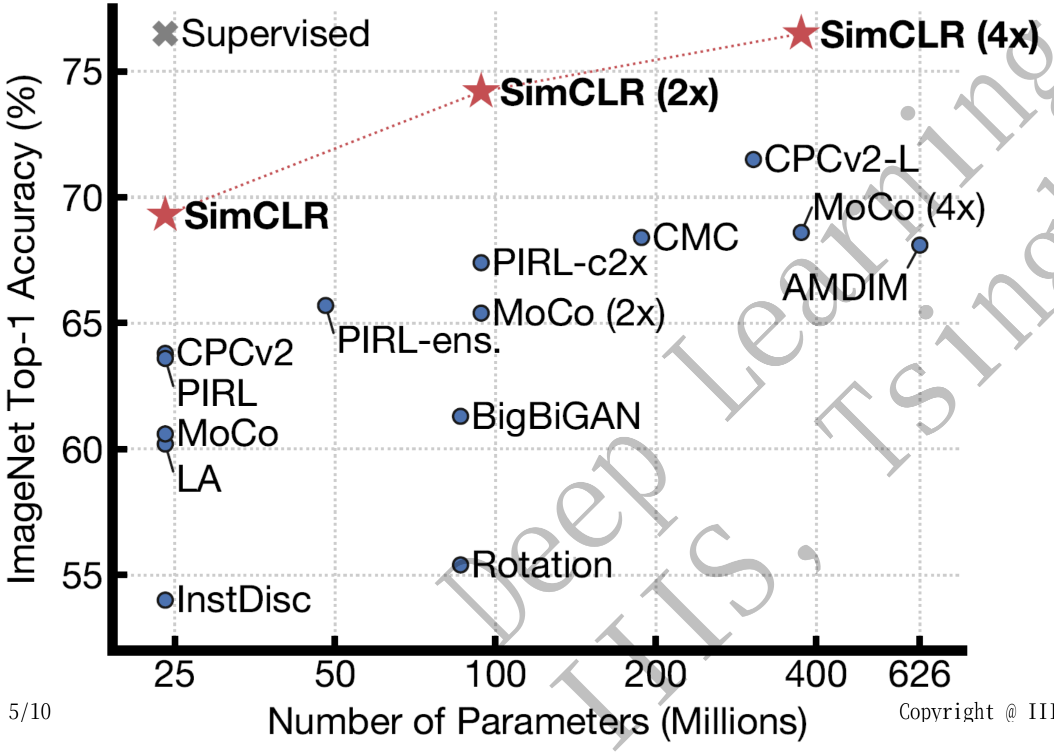
(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of f, g, $\mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i,j)$ **as** $\ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

Self-Supervised Learning

- SimCLR (Chen et al, Hinton's group, ICML 2020)
  - A Simple Framework for Contrastive Learning of Visual Representations

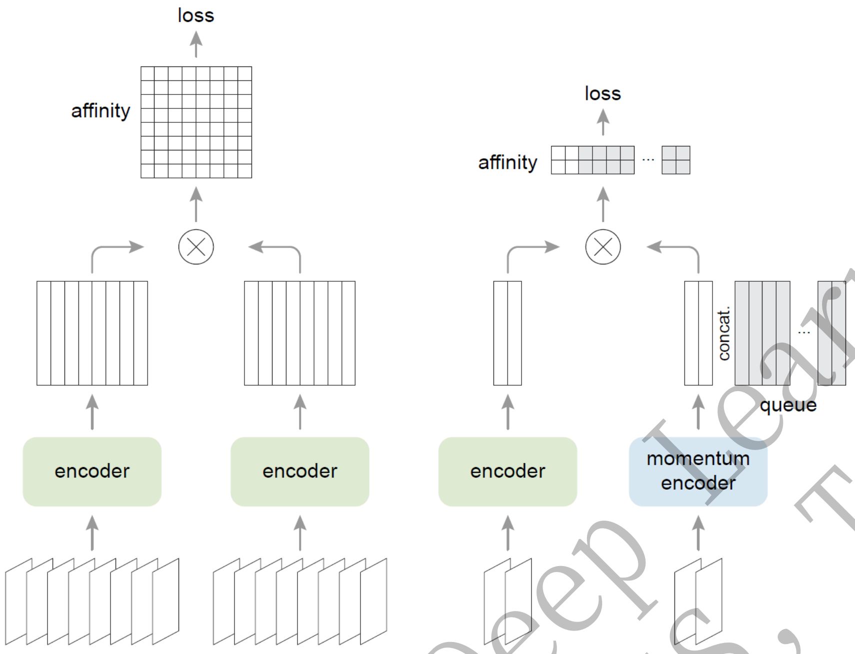


Method	Architecture	Label fraction 1% (Top 5)	Label fraction 10% (Top 5)
Supervised baseline	ResNet-50	48.4	80.4
Methods using other label-propagation:			
Pseudo-label	ResNet-50	51.6	82.4
VAT+Entropy Min.	ResNet-50	47.0	83.4
UDA (w. RandAug)	ResNet-50	-	88.5
FixMatch (w. RandAug)	ResNet-50	-	89.1
S4L (Rot+VAT+En. M.)	ResNet-50 (4×)	-	91.2
Methods using representation learning only:			
InstDisc	ResNet-50	39.2	77.4
BigBiGAN	RevNet-50 (4×)	55.2	78.8
PIRL	ResNet-50	57.2	83.8
CPC v2	ResNet-161(∗)	77.9	91.2
SimCLR (ours)	ResNet-50	75.5	87.8
SimCLR (ours)	ResNet-50 (2×)	83.0	91.2
SimCLR (ours)	ResNet-50 (4×)	**85.8**	**92.6**

Table 7. ImageNet accuracy of models trained with few labels.

Self-Supervised Learning

- MoCo-v2 (Xinlei Chen, Kaiming He, et al, 2020)
  - Larger batch size + More data augmentations + MLP projection head



(a) end-to-end (b) Momentum Contrast

Figure 1. A **batching** perspective of two optimization mechanisms for contrastive learning. Images are encoded into a representation space, in which pairwise affinities are computed.

case	unsup. pre-train MLP	aug+	cos	epochs	batch	ImageNet acc.
MoCo v1 [6]				200	256	60.6
SimCLR [2]	✓	✓	✓	200	256	61.9
SimCLR [2]	✓	✓	✓	200	8192	66.6
MoCo v2	✓	✓	✓	200	256	**67.5**
*results of **longer** unsupervised training follow:*						
SimCLR [2]	✓	✓	✓	1000	4096	69.3
MoCo v2	✓	✓	✓	800	256	**71.1**

Table 2. **MoCo *vs.* SimCLR**: ImageNet linear classifier accuracy (**ResNet-50, 1-crop 224×224**), trained on features from unsupervised pre-training. "aug+" in SimCLR includes blur and stronger color distortion. SimCLR ablations are from Fig. 9 in [2] (we thank the authors for providing the numerical results).

mechanism	batch	memory / GPU	time / 200-ep.
MoCo	256	**5.0G**	**53 hrs**
end-to-end	256	7.4G	65 hrs
end-to-end	4096	93.0G†	n/a

Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. †: based on our estimation.

Self-Supervised Learning

- MoCo-v3 (Xinlei Chen, Saining Xie, Kaiming He, 2021)
  - Built for ViT & stability enhancement for self-supervised learning on ViT
  - No sample queue; Use other mini-batch samples as negative samples

framework	model	params	acc. (%)
linear probing:			
iGPT [9]	iGPT-L	1362M	69.0
iGPT [9]	iGPT-XL	6801M	72.0
MoCo v3	ViT-B	86M	76.7
MoCo v3	ViT-L	304M	77.6
MoCo v3	ViT-H	632M	78.1
MoCo v3	ViT-BN-H	632M	79.1
MoCo v3	ViT-BN-L/7	304M	**81.0**
end-to-end fine-tuning:			
masked patch pred. [16]	ViT-B	86M	79.9†
MoCo v3	ViT-B	86M	83.2
MoCo v3	ViT-L	304M	**84.1**

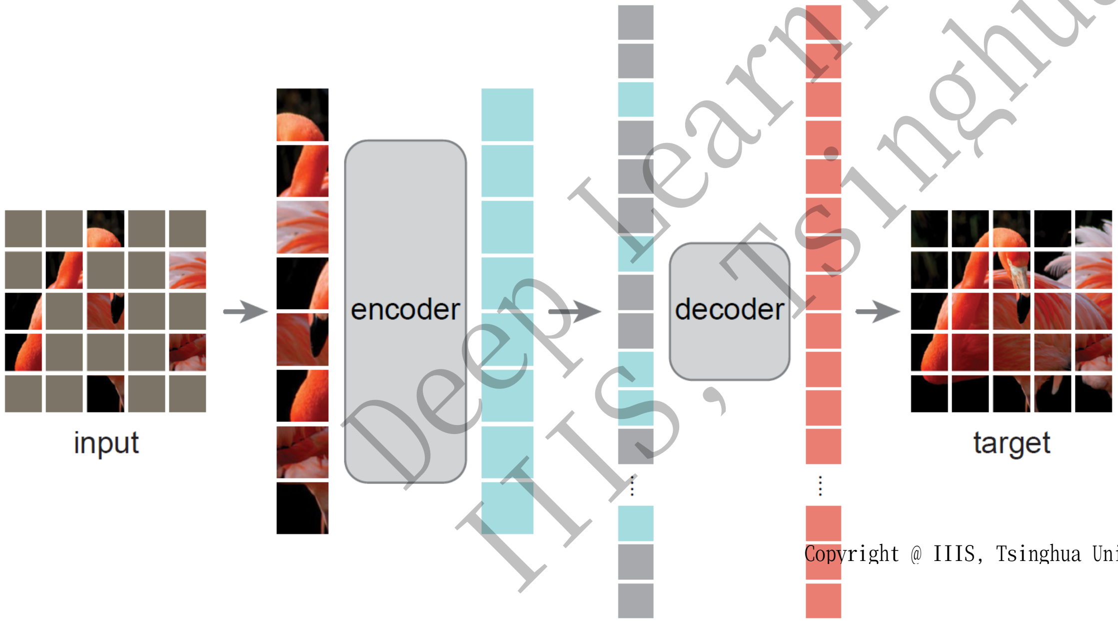
Table 1. **State-of-the-art Self-supervised Transformers in**

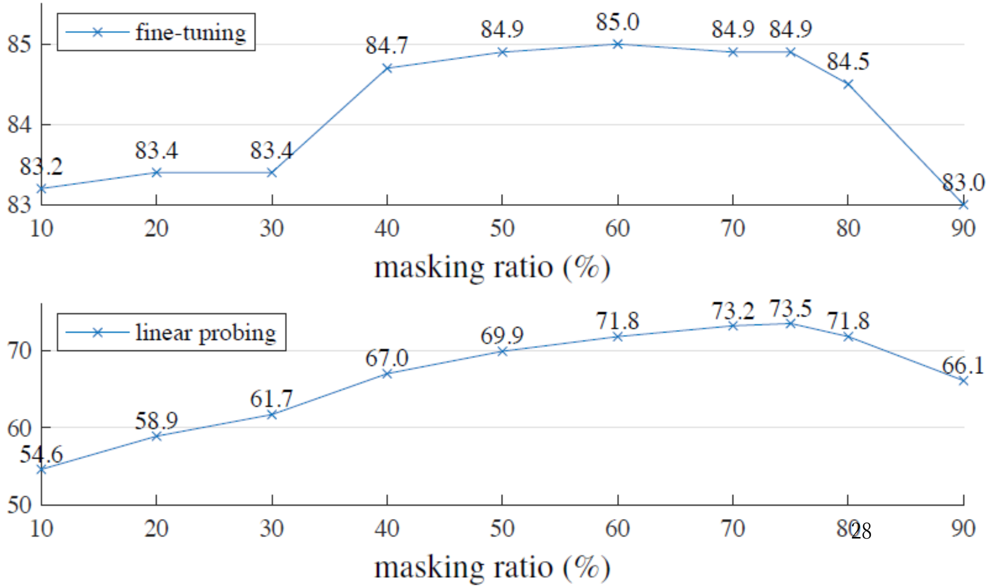
Algorithm 1 MoCo v3: PyTorch-like Pseudocode

```
# f_q: encoder: backbone + proj mlp + pred mlp
# f_k: momentum encoder: backbone + proj mlp
# m: momentum coefficient
# tau: temperature

for x in loader: # load a minibatch x with N samples
    x1, x2 = aug(x), aug(x) # augmentation
    q1, q2 = f_q(x1), f_q(x2) # queries: [N, C] each
    k1, k2 = f_k(x1), f_k(x2) # keys: [N, C] each

    loss = ctr(q1, k2) + ctr(q2, k1) # symmetrized
    loss.backward()

    update(f_q) # optimizer update: f_q
    f_k = m*f_k + (1-m)*f_q # momentum update: f_k

# contrastive loss
def ctr(q, k):
    logits = mm(q, k.t()) # [N, N] pairs
    labels = range(N) # positives are in diagonal
    loss = CrossEntropyLoss(logits/tau, labels)
    return 2 * tau * loss
```

Self-Supervised Learning

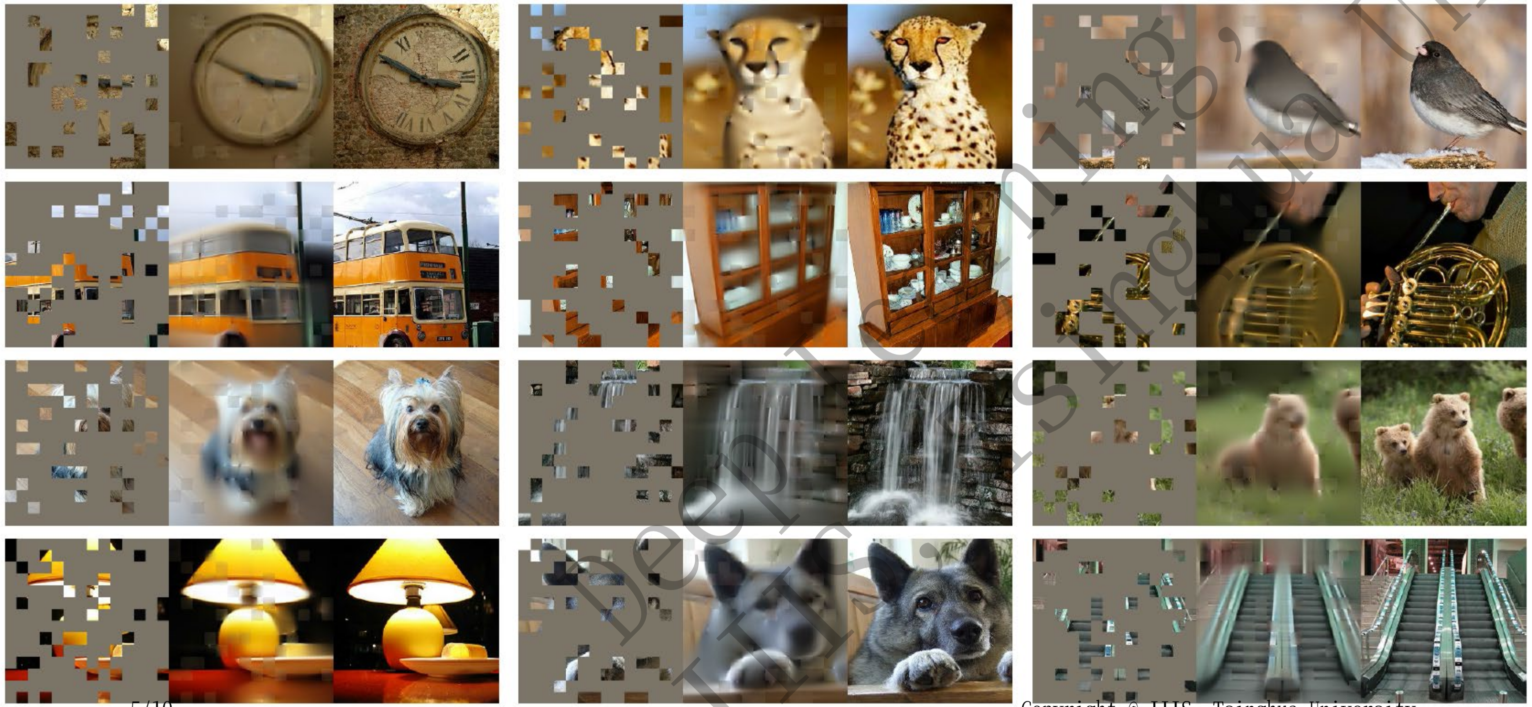
- Masked Autoencoders (MAE; Kaiming He, et al, 2021)
  - Key idea: use auto-encoder to predict (75%) masked patches
  - Encoder: ViT; only operates on visible patches
  - Decoder: shallow ViT (~10% of encoder); only used for training
  - Loss: predict pixel values for the missing patches





Self-Supervised Learning

- Masked Autoencoders (MAE; Kaiming He, et al, 2021)
  - Key idea: use auto-encoder to predict (75%) masked patches



method	pre-train data	ViT-B	ViT-L	ViT-H	ViT-H_{448}
scratch, our impl.	-	82.3	82.6	83.1	-
DINO [5]	IN1K	82.8	-	-	-
MoCo v3 [9]	IN1K	83.2	84.1	-	-
BEiT [2]	IN1K+DALLE	83.2	85.2	-	-
MAE	IN1K	83.6	85.9	86.9	**87.8**

Table 3. **Comparisons with previous results on ImageNet-**

		AP^{box}		AP^{mask}	
method	pre-train data	ViT-B	ViT-L	ViT-B	ViT-L
supervised	IN1K w/ labels	47.9	49.3	42.9	43.9
MoCo v3	IN1K	47.9	49.3	42.7	44.0
BEiT	IN1K+DALLE	49.8	**53.3**	44.4	47.1
MAE	IN1K	**50.3**	**53.3**	**44.9**	**47.2**

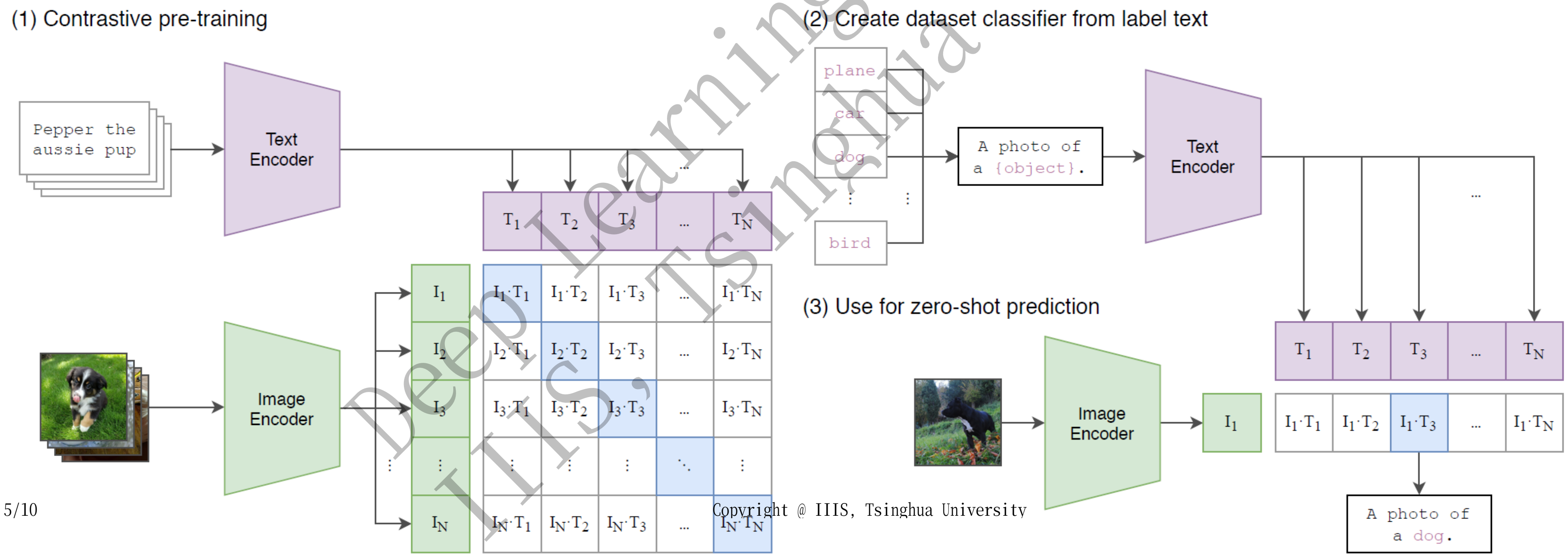
Table 4. **COCO object detection and segmentation** using a ViT

method	pre-train data	ViT-B	ViT-L
supervised	IN1K w/ labels	47.4	49.9
MoCo v3	IN1K	47.3	49.1
BEiT	IN1K+DALLE	47.1	53.3
MAE	IN1K	**48.1**	**53.6**

Table 5. **ADE20K semantic segmentation** (mIoU) using Uper-

Multi-Modal Contrastive Learning

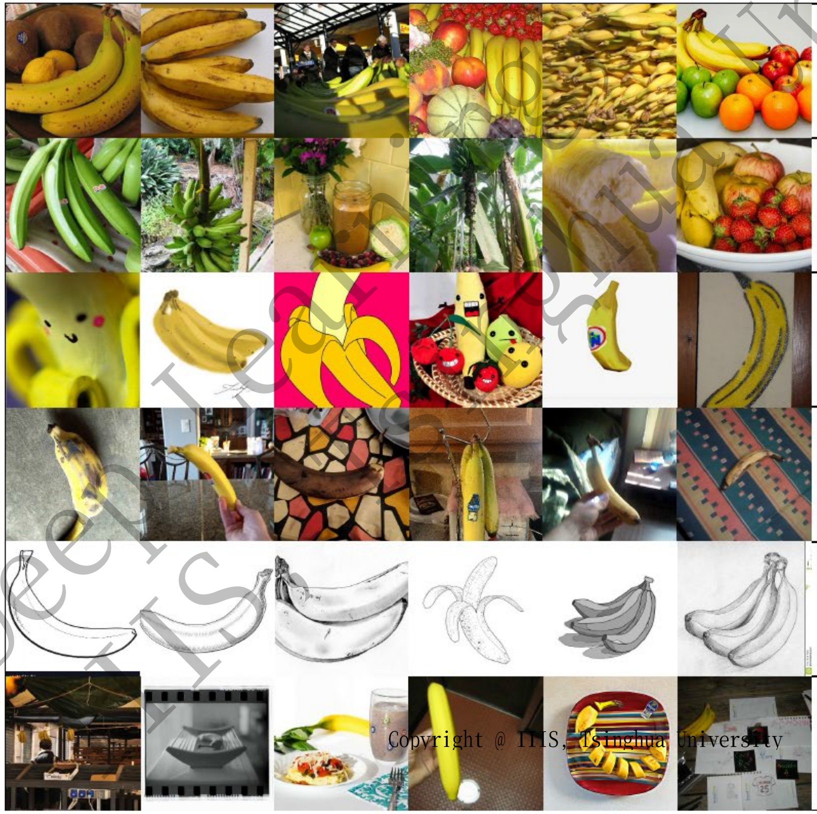
- CLIP: aligned representation for text and images (OpenAI, 2021)
  - 400M paired text-image data; Aligned representation space
    - Released model: https://github.com/OpenAI/CLIP
  - Contrastive learning on paired text and image representations



Multi-Modal Contrastive Learning

- CLIP: aligned representation for text and images (OpenAI, 2021)
  - CLIP captures strongly semantic information

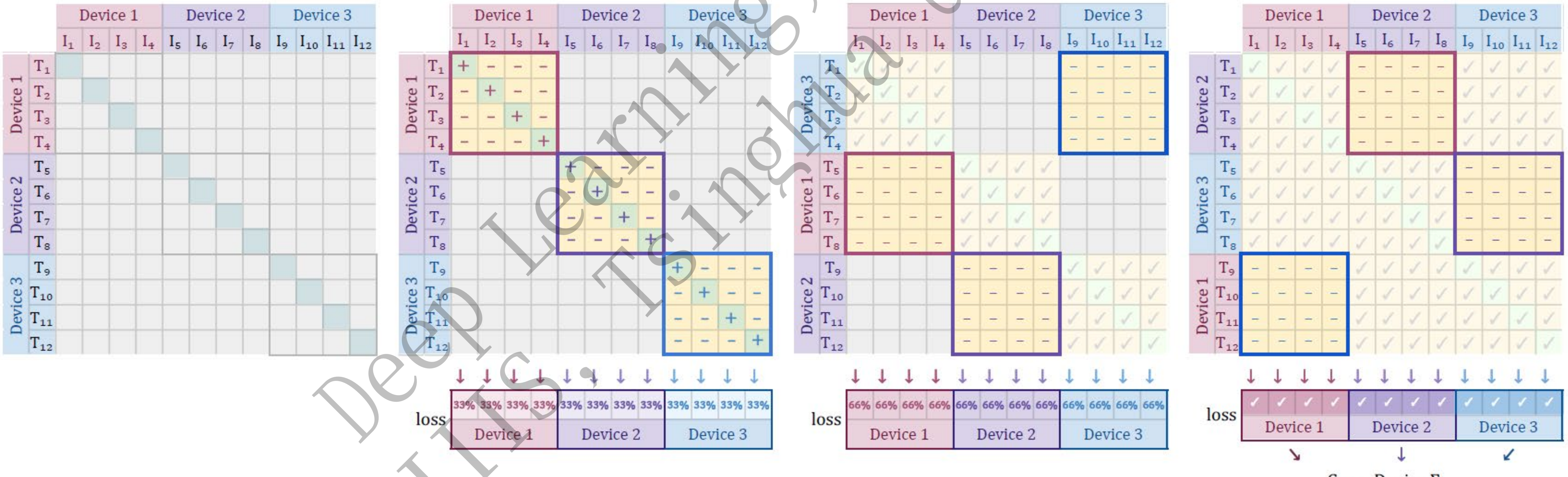
Dataset Examples



Dataset	ImageNet ResNet101	Zero-Shot CLIP	Δ Score
ImageNet	**76.2**	**76.2**	0%
ImageNetV2	64.3	**70.1**	+5.8%
ImageNet-R	37.7	**88.9**	+51.2%
ObjectNet	32.6	**72.3**	+39.7%
ImageNet Sketch	25.2	**60.2**	+35.0%
ImageNet-A	2.7	**77.1**	+74.4%

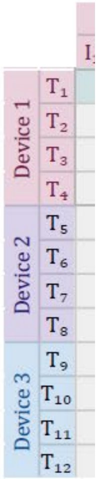
Multi-Modal Contrastive Learning

- SigLIP: Sigmoid Loss for Language Image Pre-Training (Google, ICCV 2023)
    - Change softmax operator in CLIP to sigmoid loss + large scaling training



Multi-Modal Contrastive Learning

- SigLIP: Sigmoid Loss for Language Image Pre-Training (Google, ICCV 2023)
  - Change softmax operator in CLIP to sigmoid loss + large scaling training



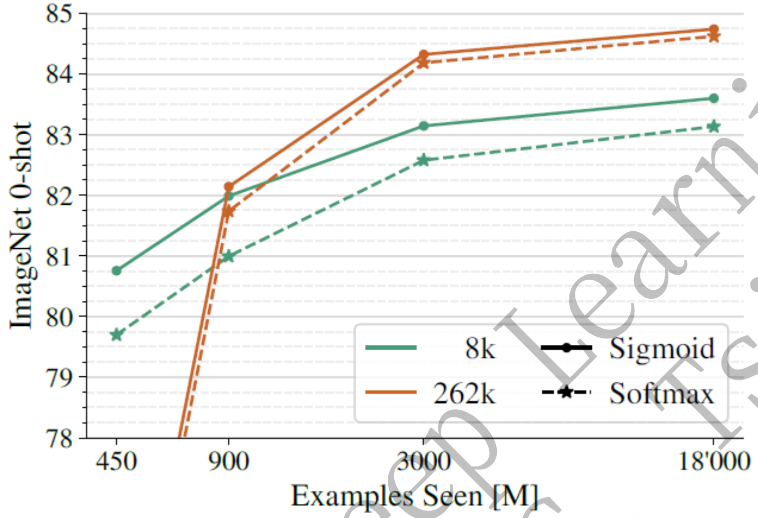
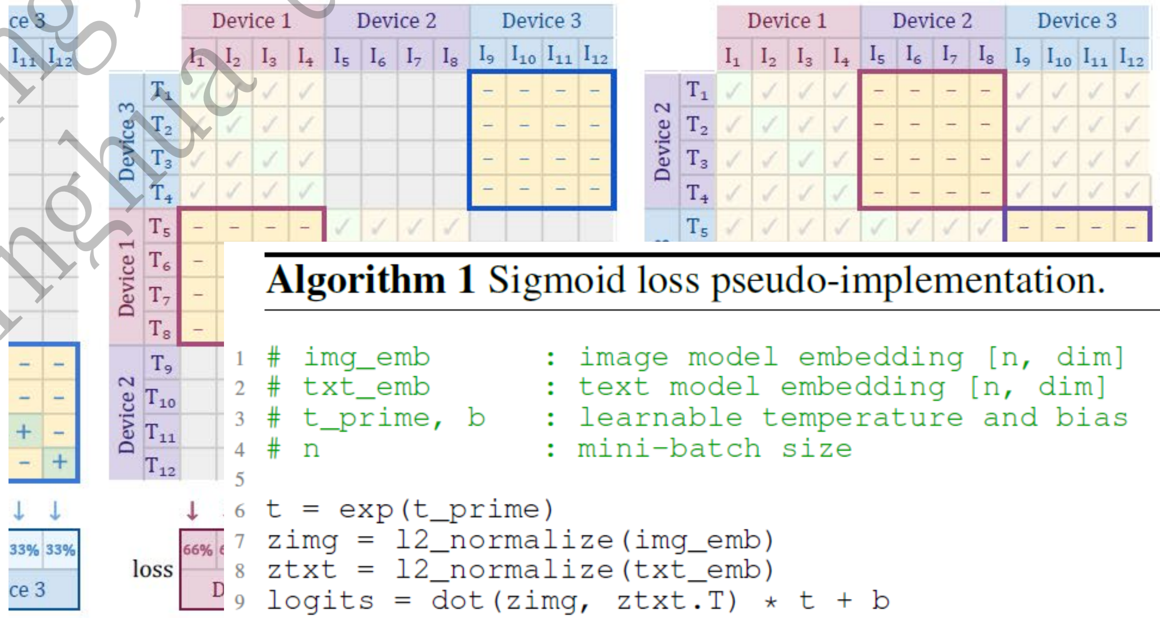


Figure 3: **SigLiT ImageNet 0-shot transfer results with different training durations.** Large batch size results in a big performance boost, but needs a sufficiently long schedule to ramp up, as for short schedules, very large batch size results in a small number of gradient update steps.

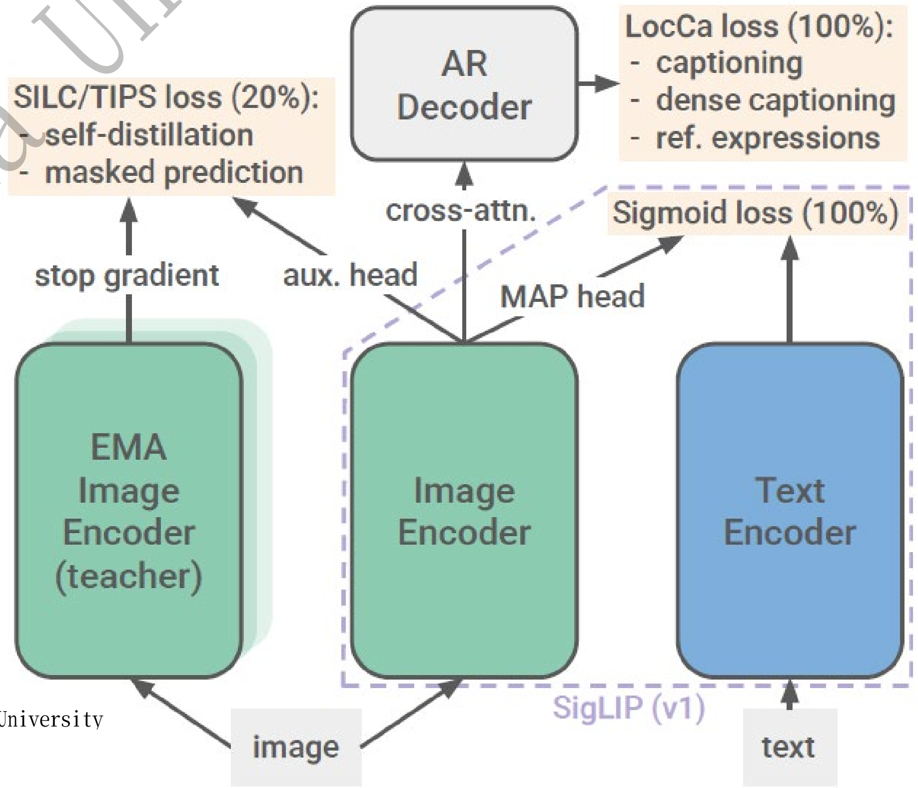


Algorithm 1 Sigmoid loss pseudo-implementation.

```
# img_emb        : image model embedding [n, dim]
# txt_emb        : text model embedding [n, dim]
# t_prime, b     : learnable temperature and bias
# n              : mini-batch size

t = exp(t_prime)
zimg = l2_normalize(img_emb)
ztxt = l2_normalize(txt_emb)
logits = dot(zimg, ztxt.T) * t + b
labels = 2 * eye(n) - ones(n) # -1 with diagonal 1
l = -sum(log_sigmoid(labels * logits)) / n
```

Multi-Modal Contrastive Learning

- SigLIP v2: Improved Multilingual Vision-Language Encoders (Google Deepmind, 2025)
  - SigLIP combined with captioning-based pretraining, self-supervised losses and online data curation



Multi-Modal Contrastive Learning

- SigLIP v2: Improved Multilingual Vision-Language Encoders (Google Deepmin
  - SigLIP (… d losses and online

			Segmentation ↑		Depth ↓		Normals ↓	
Model	ViT	Res.	PASCAL	ADE20k	NYUv2	NAVI	NYUv2	NAVI
CLIP [50]	L/14	224	74.5	39.0	0.553	0.073	**24.3**	25.5
OpenCLIP [27]	G/14	224	71.4	39.3	0.541	–	–	–
SigLIP [71]	So/14	224	72.0	37.6	0.576	0.083	25.9	26.0
SigLIP 2	So/14	224	**77.1**	**41.8**	**0.493**	**0.067**	24.9	**25.4**
SigLIP [71]	So/14	384	73.8	40.8	0.563	0.069	24.1	25.4
SigLIP 2	So/14	384	**78.1**	**45.4**	**0.466**	**0.064**	**23.0**	**25.0**

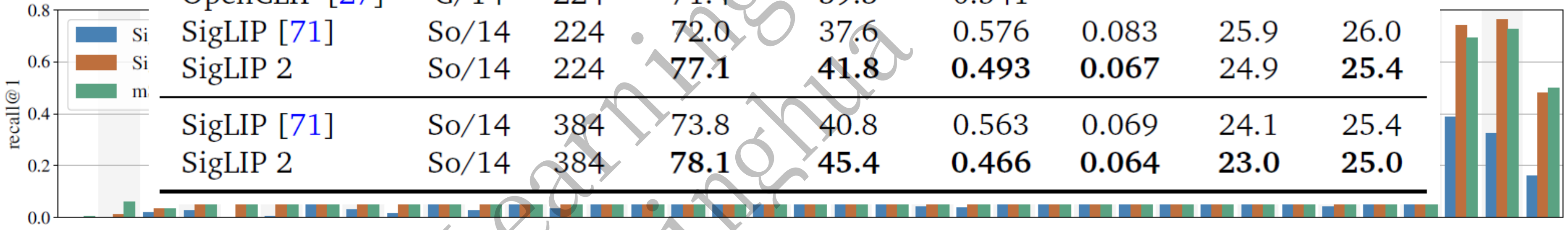
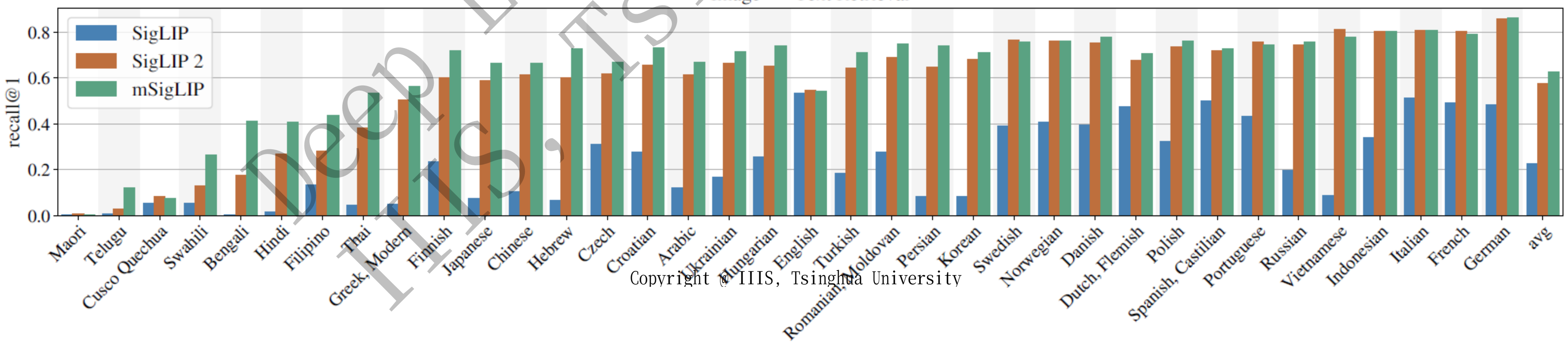


Image → Text Retrieval



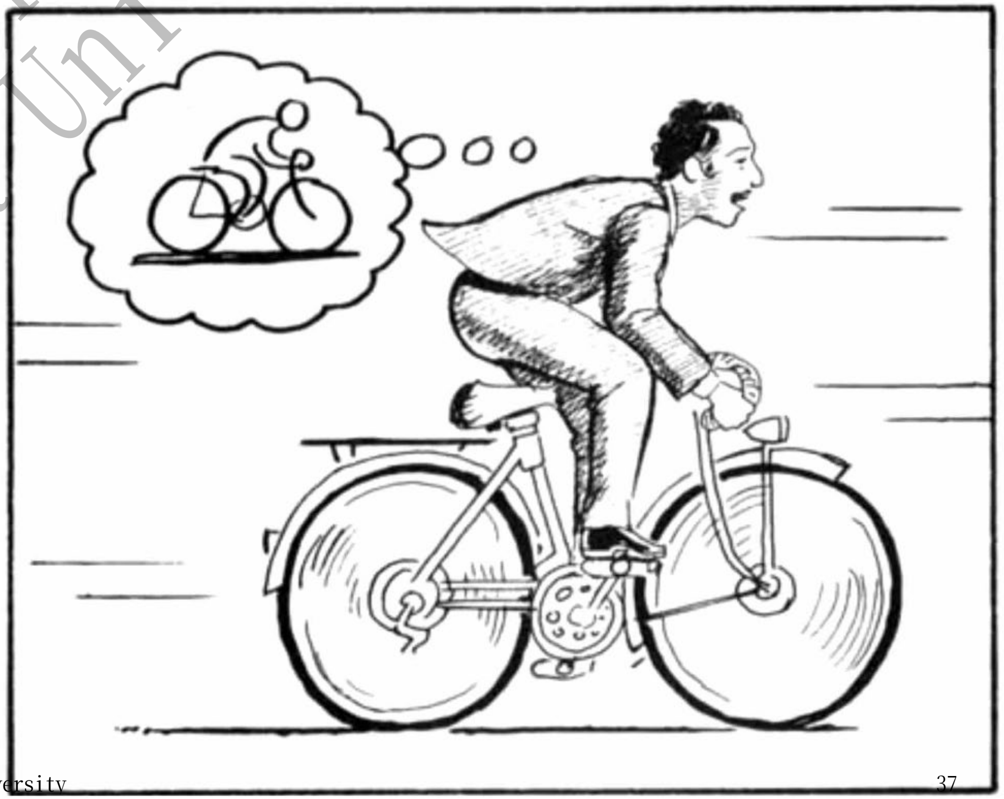
Multi-Modal Contrastive Learning

- SigLIP v2: Improved Multilingual Vision-Language Encoders (Google Deepmind, 2025)
  - SigLIP combined with captioning-based pretraining, self-supervised losses and online data curation **+ more diverse high-quality data**

			Segmentation ↑		Depth ↓		Normals ↓	
Model	ViT	Res.	PASCAL	ADE20k	NYUv2	NAVI	NYUv2	NAVI
CLIP [50]	L/14	224	74.5	39.0	0.553	0.073	**24.3**	25.5
OpenCLIP [27]	G/14	224	71.4	39.3	0.541	–	–	–
SigLIP [71]	So/14	224	72.0	37.6	0.576	0.083	25.9	26.0
SigLIP 2	So/14	224	**77.1**	**41.8**	**0.493**	**0.067**	24.9	**25.4**
SigLIP [71]	So/14	384	73.8	40.8	0.563	0.069	24.1	25.4
SigLIP 2	So/14	384	**78.1**	**45.4**	**0.466**	**0.064**	**23.0**	**25.0**

Predictive Modeling with Actions

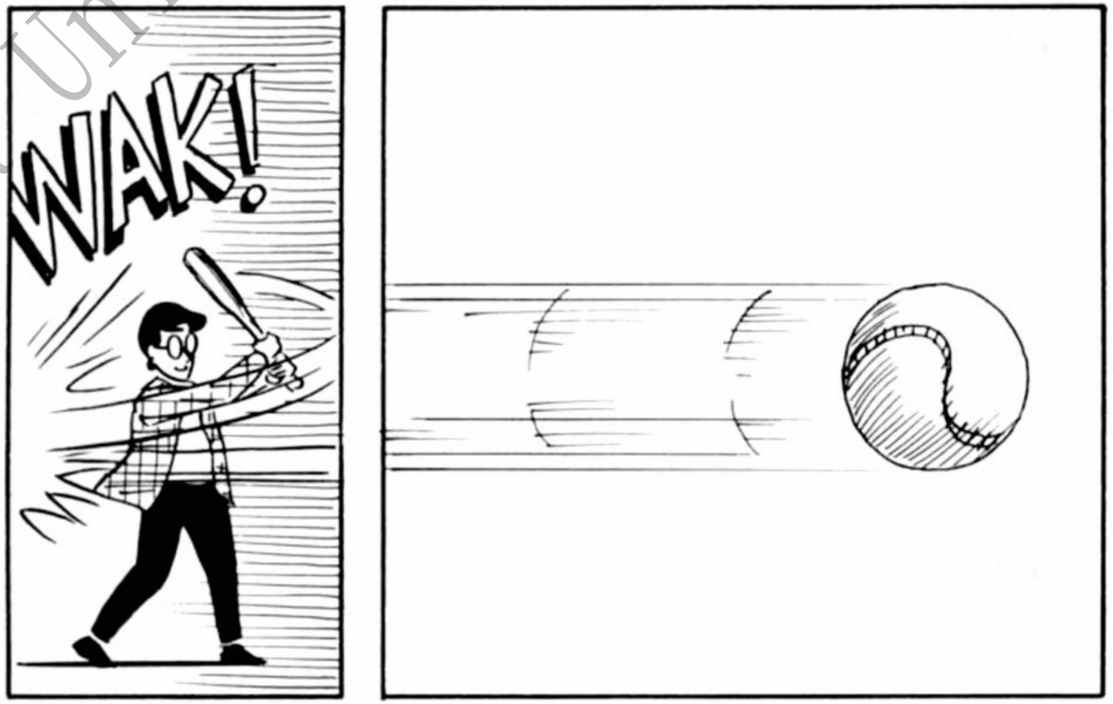
- Can we build intelligence from massive data in the world?
  - A model on video data
  - Prediction with actions
- The world model (Ha et al. 2018)
  - Learn latent representations from video
    - A VAE model over images
    - $x_t \rightarrow z_t$
  - Prediction based on actions
    - A transition model over latent variables
    - $f(z_t, a_t) \rightarrow z_{t+1}$
    - $a_t = C(z_t)$ can be learned or given



A World Model, from Scott McCloud's *Understanding Comics*. [1, 2]

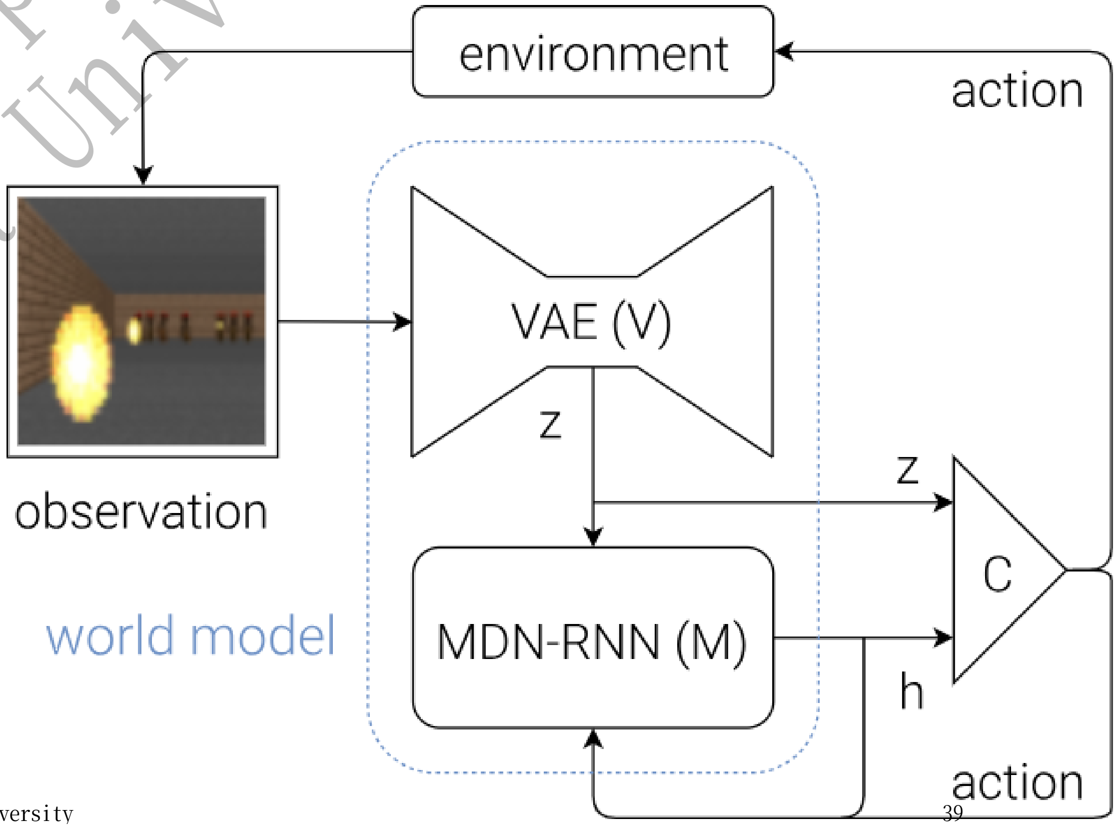
Predictive Modeling with Actions

- Can we build intelligence from massive data in the world?
  - A model on video data
  - Prediction with actions
- The world model (Ha et al. 2018)
  - Learn latent representations from video
    - A VAE model over images
    - $x_t \to z_t$
  - Prediction based on actions
    - A transition model over latent variables
    - $f(z_t, a_t) \to z_{t+1}$
    - $a_t = C(z_t)$ can be learned or given



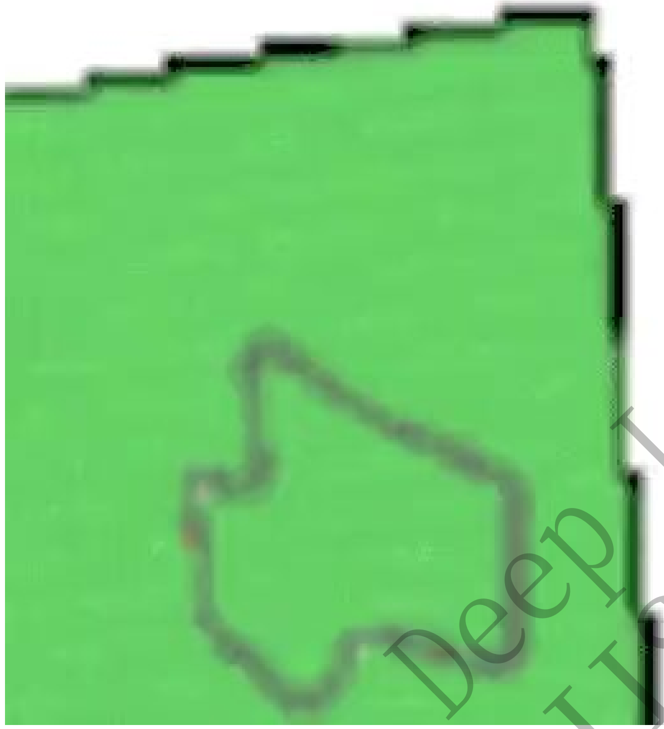
Predictive Modeling with Actions

- Can we build intelligence from massive data in the world?
  - A model on video data
  - Prediction with actions
- The world model (Ha et al. 2018)
  - Learn latent representations from video
    - A VAE model over images
    - $x_t \to z_t$
  - Prediction based on actions
    - A transition model over latent variables
    - $f(z_t, a_t) \to z_{t+1}$
    - $a_t = C(z_t)$ can be learned or given

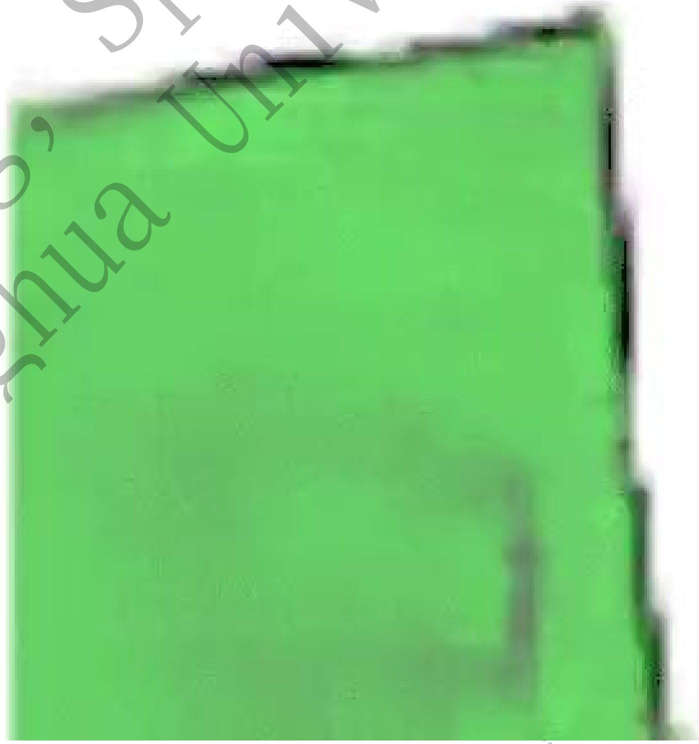


Predictive Modeling with Actions

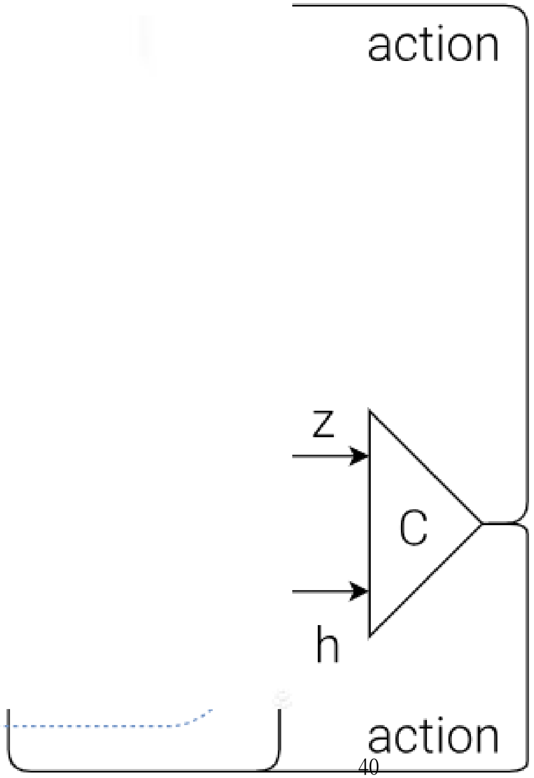
-
-



Ground-Truth Car Racing



Recovered from world model latent

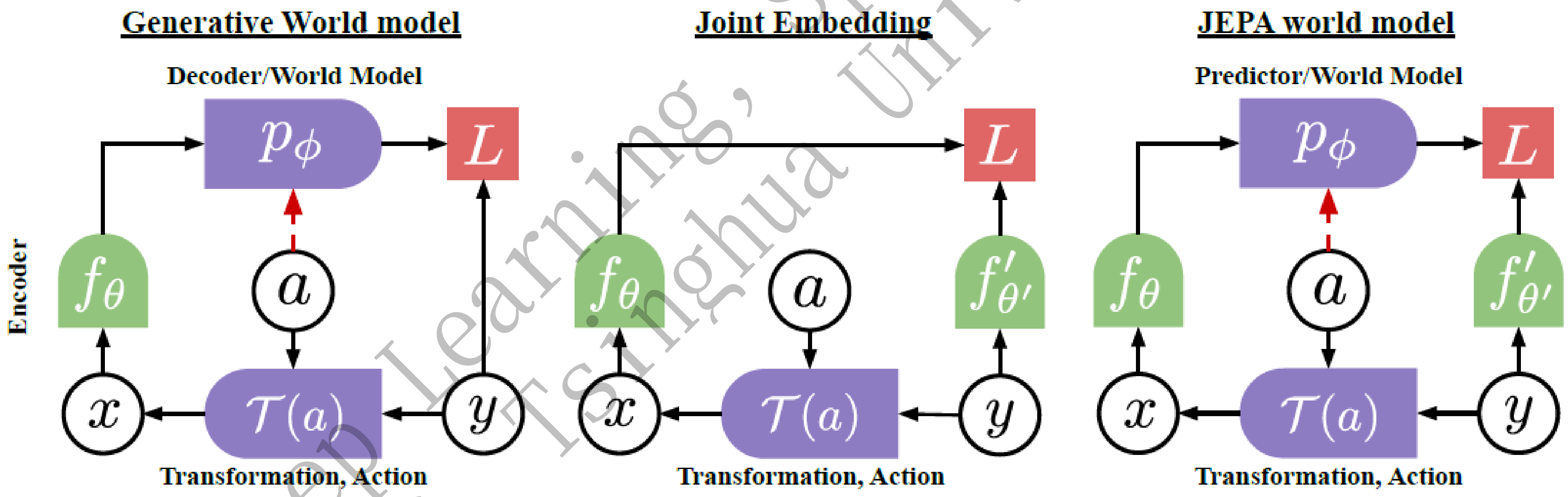


Predictive Modeling with Actions

- VPT: Learning to Act by Watching Videos (OpenAI, 2022)

Predictive Modeling with Actions

- The Latent World Models (LeCun et al, 2024) (https://arxiv.org/abs/2403.00504)
  - Direct MoCo-style representation learning over latent variables



	Generative World model	Joint Embedding	JEPA world model
Unconditional	Denoising Autoencoders Variational Autoencoders	Siamese, SimCLR, VICReg, DINO	BYOL, SimSiam
Conditional	Generative World Models Masked Image Modeling	N/A	Latent world models, I-JEPA, Equivariant SSL, **IWM (Ours)**

Predictive Modeling with Actions

- The Latent World Models (LeCun et al, 2024) (https://arxiv.org/abs/2403.00504)
  - Direct MoCo-style representation learning over latent variables

Table 4 Finetuning evaluations on ImageNet-1k. We evaluate prediction based methods by finetuning their encoder, by keeping the encoder frozen and finetuning their predictive world model or by finetuning both. Finetuning the world model is highly effective with IWM when it exhibits an equivariant behavior. This behavior is absent or less clear with other methods, showing the importance of a strong world model.

Method	Epochs	No predictor	Frozen encoder, tuned predictor		End to end
		Encoder	Random Init.	Pretrained	
MAE	300	82.7	82.4	82.7 (+0.3)	82.3
	1600	**83.6**	**83.0**	83.1 (+0.1)	83.3
I-JEPA	300	83.0	79.1	80.0 (+0.9)	82.0
$\text{IWM}^{\text{Inv}}_{12,384}$	300	83.3	80.5	81.3 (+0.8)	82.7
$\text{IWM}^{\text{Equi}}_{18,384}$	300	82.9	81.5	**83.3** (+1.8)	**84.4**

Predictive Modeling with Actions

- The Latent World Models (LeCun et al, 2024) (https://arxiv.org/abs/2403.00504)
  - Direct MoCo-style representation learning over latent variables

Table 4 Finetuning evaluations on ImageNet-1k. We evaluate prediction based methods by finetuning their encoder, by keeping the encoder frozen and finetuning their predictive world model or by finetuning both. Finetuning the world model is highly effective with IWM when it exhibits an equivariant behavior. This behavior is absent or less clear with other methods, showing the importance of a strong world model.

Method	Epochs	No predictor	Frozen encoder, tuned predictor		End to end
		Encoder	Random Init	Pretrained	
MAE	300	82.7	82.4	82.7 (+0.3)	82.3
	1600	**83.6**	**83.0**	83.1 (+0.1)	83.3
I-JEPA	300	83.0	79.1	80.0 (+0.9)	82.0
$\text{IWM}^{\text{Inv}}_{12,384}$	300	83.3	80.5	81.3 (+0.8)	82.7
$\text{IWM}^{\text{Equi}}_{18,384}$	300	82.9	81.5	**83.3** (+1.8)	**84.4**

What if we train a latent world model with massive video data?

Predictive Modeling with Actions

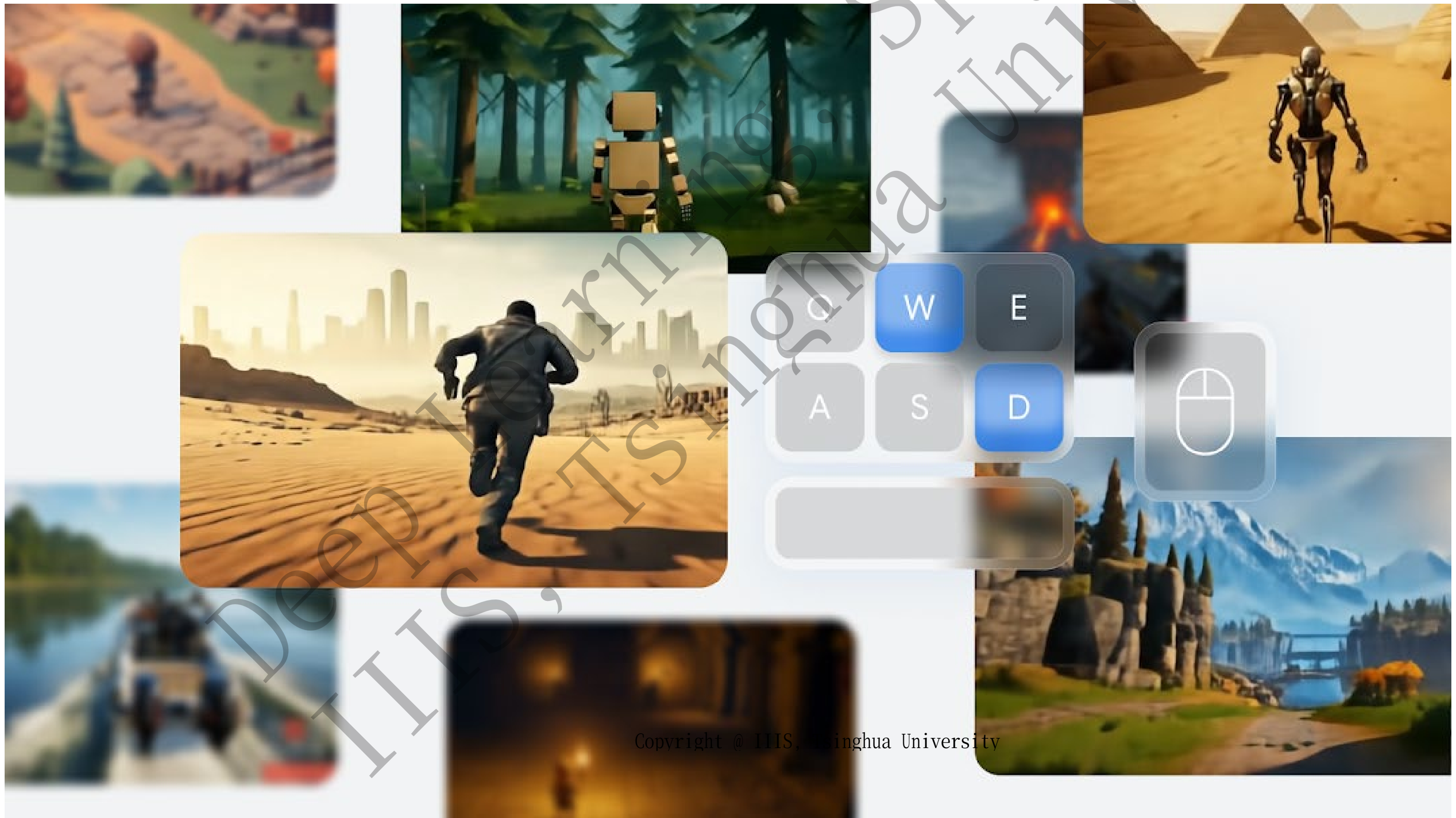
- Genie: Generative Interactive Environments (DeepMind, 2024)
    - Convert a text/image into an actionable world

Predictive Modeling with Actions

- Genie: Generative Interactive Environments (DeepMind, 2024)
  - Convert a text/image into an actionable world

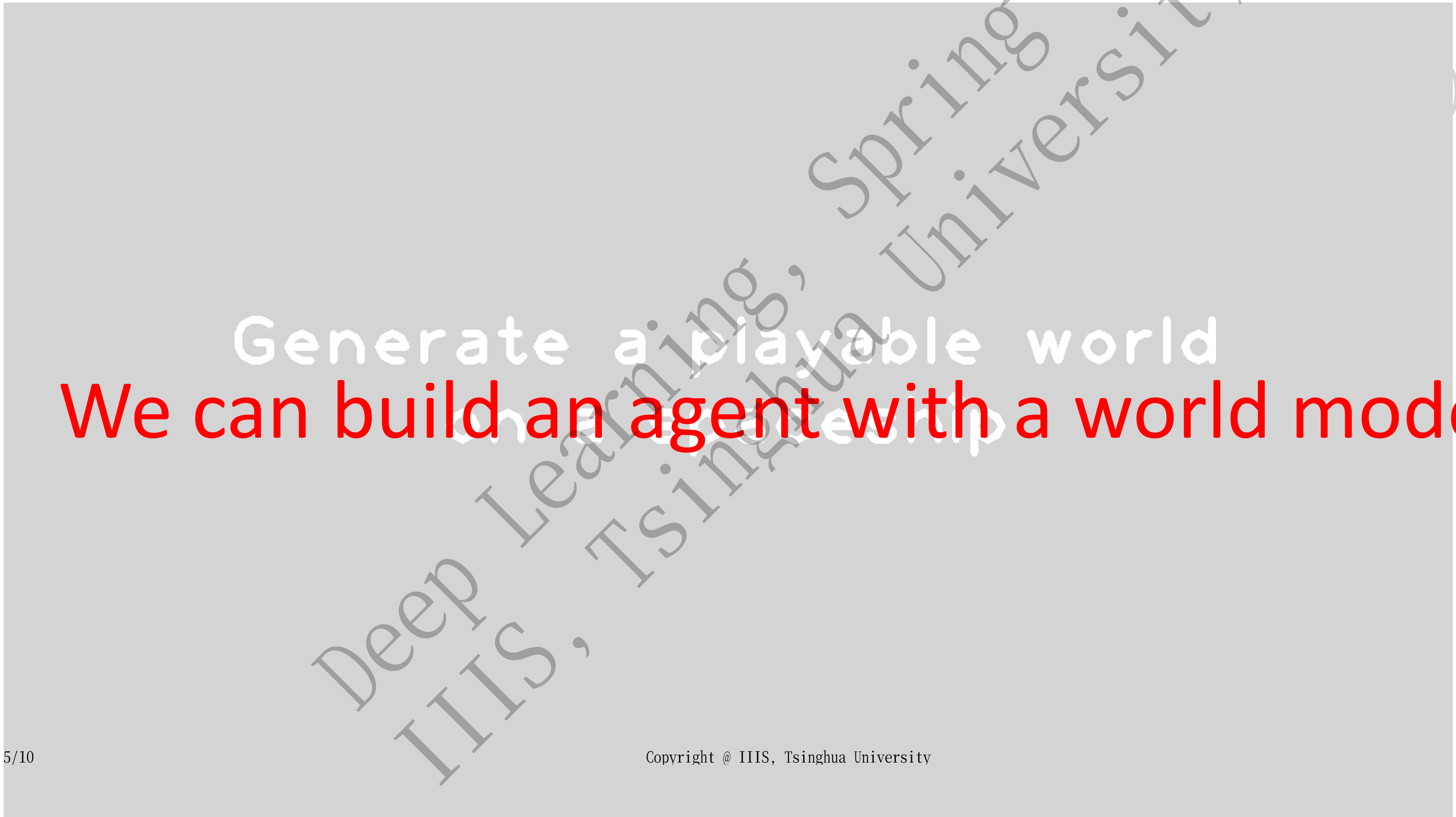
Predictive Modeling with Actions

- Genie 2: A large-scale foundation world model (DeepMind, 2024)
  - Large-scale training + consistent actions → 3D scenes + 10~20s videos



Predictive Modeling with Actions

Predictive Modeling with Actions



We can build an agent with a world model!

Predictive Modeling with Actions

- Dreamer v3: Mastering diverse control tasks through world models (Deepmind, 2025)
  - Large-scale world model learning + model-based reinforcement learning

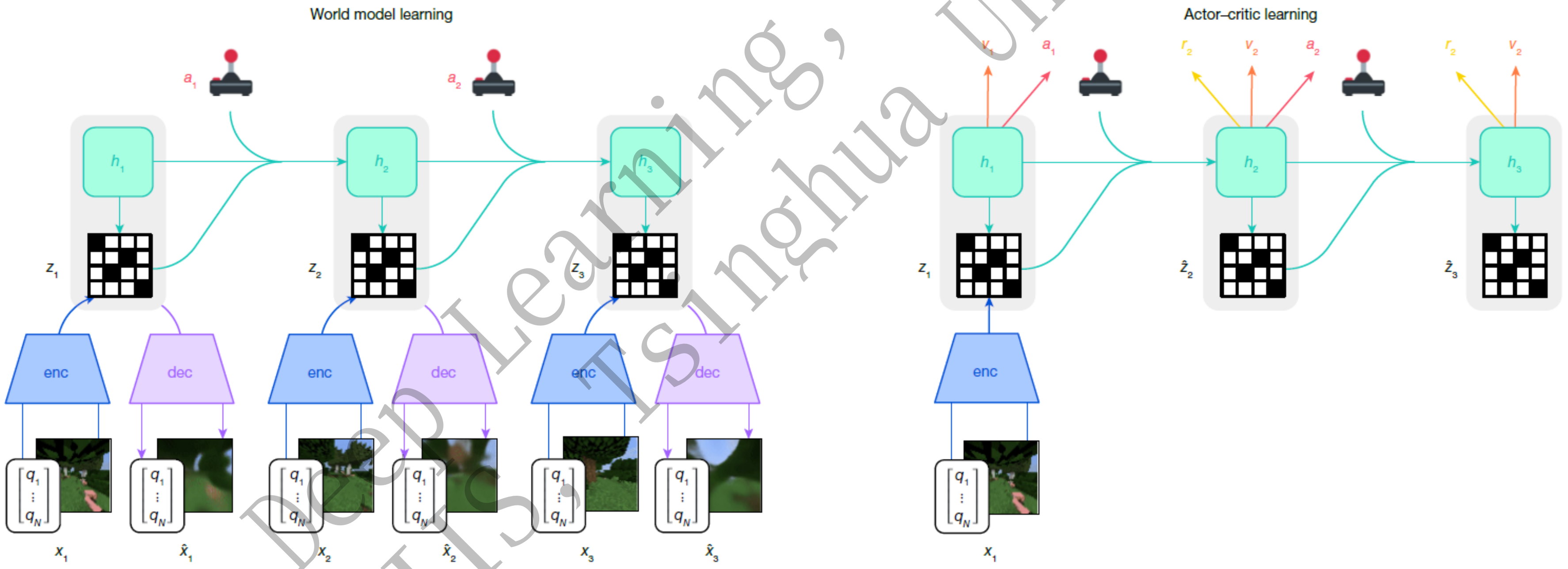
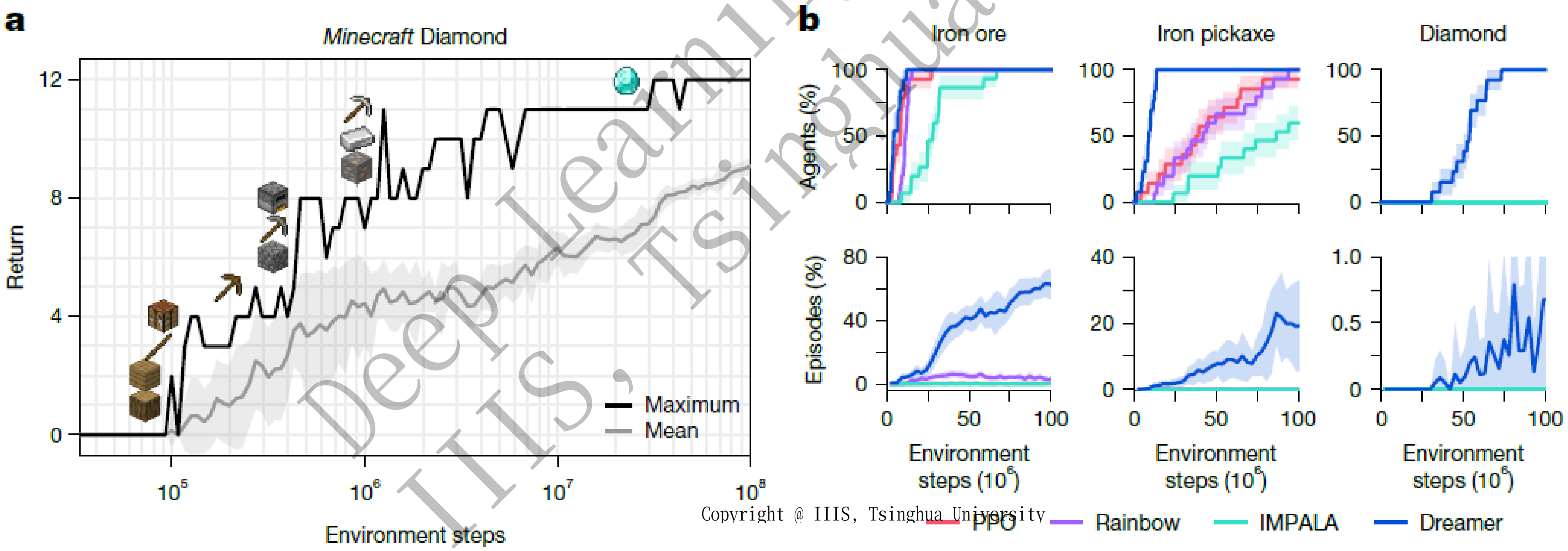


Fig. 1 | Training process of Dreamer. The world model encodes sensory inputs x_t using the encoder (enc) into discrete representations z_t that are predicted by a sequence model with recurrent state h_t given actions a_t. The inputs are reconstructed as $\hat{x}_t$ using the decoder (dec) to shape the representations. The actor and critic predict actions a_t and values v_t and learn from trajectories of abstract representations $\hat{z}_t$ and rewards r_t predicted by the world model.

Predictive Modeling with Actions

- Dreamer v3: Mastering diverse control tasks through world models (Deepmind, 2025)
  - Large-scale world model learning + model-based reinforcement learning
  - First project to discover diamond in Minecraft without human supervision



Summary

- Unsupervised Learning
  - Leverage the massive unlabeled data
  - Only X available → Generative models
- Self-supervised Learning
  - Create virtual supervision using portion of X
    - Prediction-based SL
      - Rotation, color, patches; predict futures (CPC);
      - Random mask + reconstruction (BERT, MAE)
    - Contrastive Learning
      - CPC, MoCo, SimCLR: maximize agreement between transformations; need negative samples
      - CLIP/SigLIP: cross-modal representation learning
- World Model
  - Predictive video models with actions

Today’s Topic

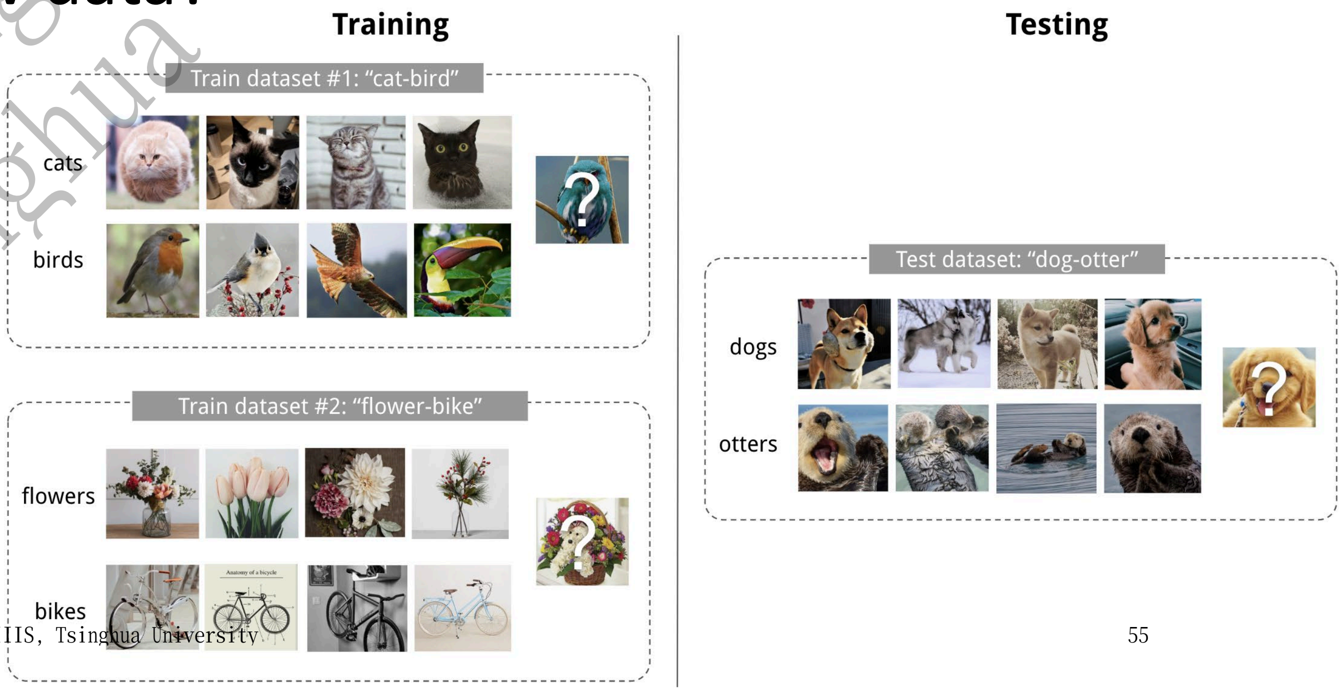
- Unsupervised Learning and Self-supervised Learning
- Learning to Efficiently Learn Neural Networks
  - Aka. Meta-Learning, Learning to Learn
- Reinforcement Learning and Human-AI Collaboration
  - Some interesting projects from Prof. Wu’s group

Today’s Topic

- Unsupervised Learning and Self-supervised Learning
- **Learning to Efficiently Learn Neural Networks**
  - **Aka. Meta-Learning, Learning to Learn**
- Reinforcement Learning and Human-AI Collaboration
  - Some interesting projects from Prof. Wu’s group

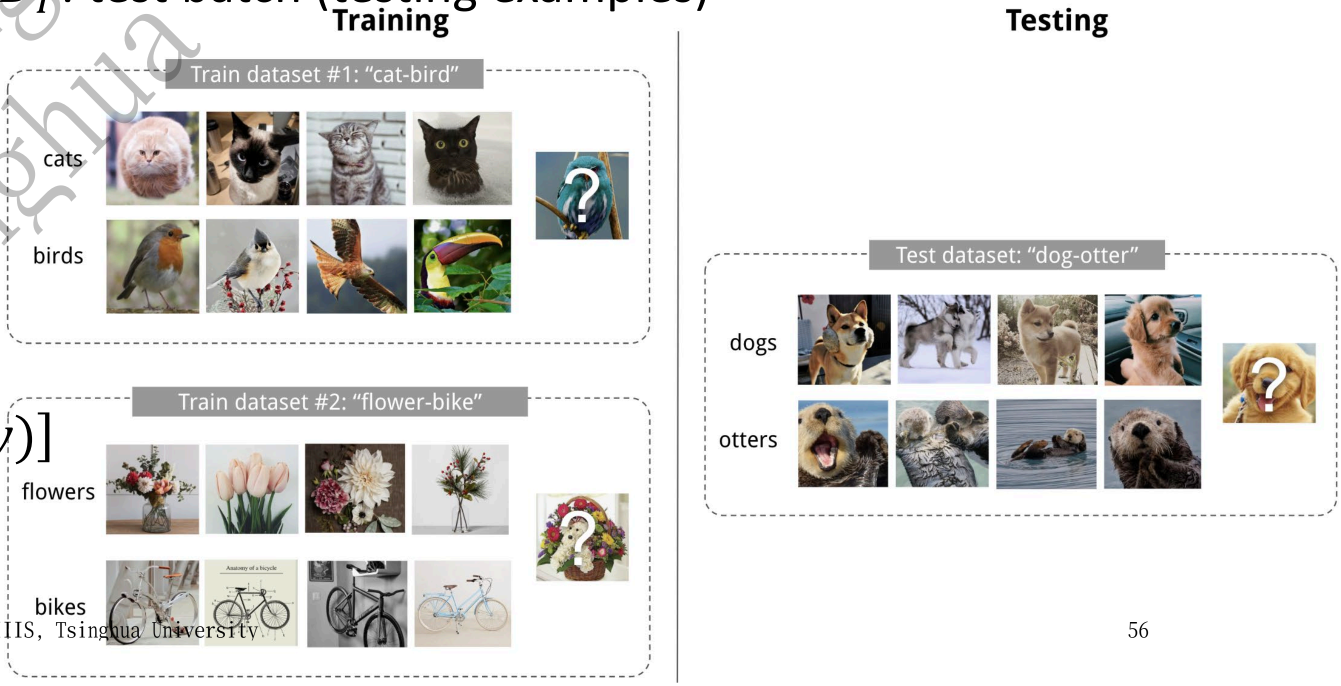
Few-Shot Learning

- Unsupervised Learning
  - Deep learning requires lots of data-label pairs
  - Unsupervised/Self-supervised Learning to produce supervision
- What if you really just have a few data?
  - Few-shot learning
  - Training
    - A collection of "tasks"
    - Task: a few samples and targets
  - Testing
    - A few new labeled samples
    - Prediction



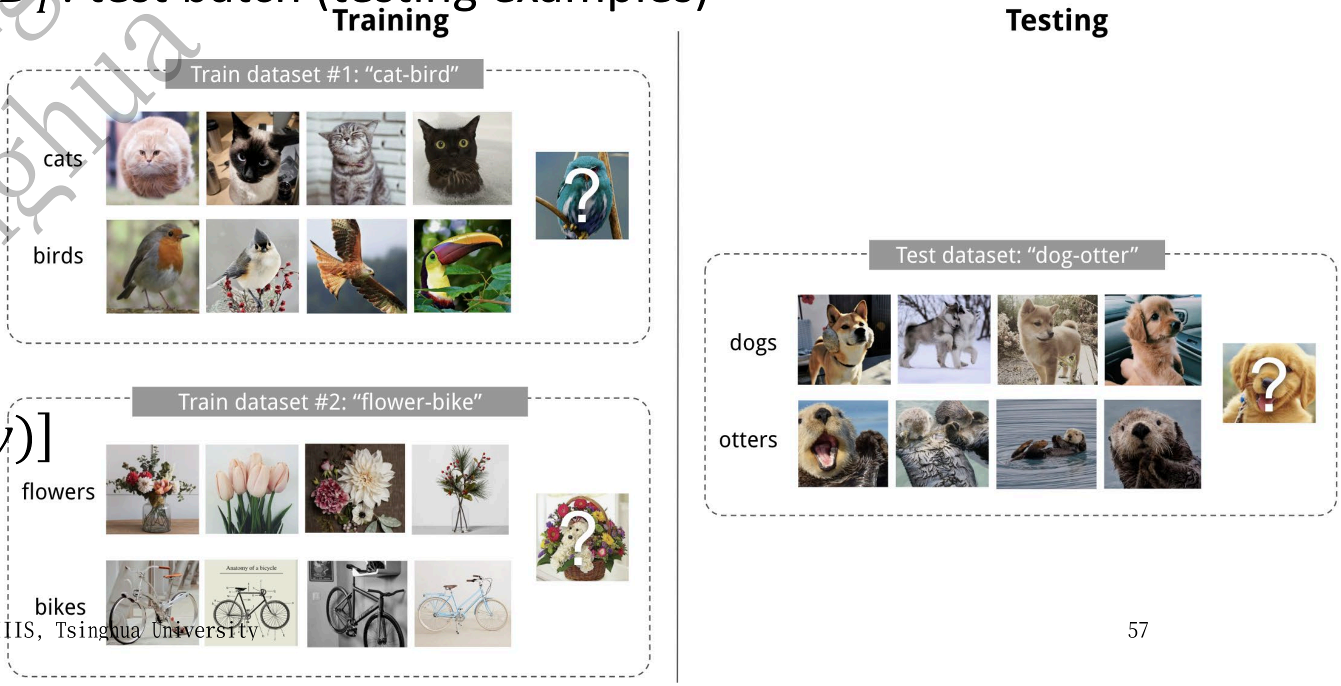
Few-Shot Learning

- Formulation
  - Training data $\mathcal{D}$: a collection of task T
    - $T: \{S_T, B_T\} = \left\{\left(x_i^S, y_i^S\right)_i, \left(x_j^B, y_j^B\right)_j\right\}$
    - S_T: support set (training examples), B_T: test batch (testing examples)
    - $|S_T|$ is typically small
  - Target: few-shot learner
    - $y = f_\theta(x|S_T)$ for $x \in B$
    - Predict based on support set
  - Training
    - $L(T, \theta) = E_{(x,y)\in B_T}[Div(f_\theta(x|S_T), y)]$
    - $\theta^\star = \arg\min_\theta E_{T\in\mathcal{D}}[L(T, \theta)]$



Meta Learning

- Formulation
  - Training data $\mathcal{D}$: a collection of task T
    - $T: \{S_T, B_T\} = \left\{\left(x_i^S, y_i^S\right)_i, \left(x_j^B, y_j^B\right)_j\right\}$
    - S_T: support set (training examples), B_T: test batch (testing examples)
    - $|S_T|$ is typically small
  - Target: meta learner
    - $y = f_\theta(x|S_T)$
    - Predict based on support set
  - Meta Training
    - $L(T, \theta) = E_{(x,y)\in B_T}[Div(f_\theta(x|S_T), y)]$
    - $\theta^\star = \arg\min_\theta E_{T\in\mathcal{D}}[L(T, \theta)]$



Meta Learning

- Standard Learning
  - Learner: $y = f_\theta(x)$
    - Once θ is learned, the output of $f(x)$ is never changed
  - Training:
    - A single task of massive data pair $T = \{(x_i, y_i)\}$, SGD is used for training
- Meta-Learning
  - Meta-Leaner: $y = f_\theta(x|S_T)$
    - f_θ can further adapt with additional samples in S_T
  - Meta-Training:
    - A large collection of tasks is provided $\mathcal{D} = \{T_i\}$
    - Use SGD to learn meta-learner on each T_i (outer loop)
    - Meta-learner $f_\theta(x|S)$ must adapt quickly with S_{T_i} (inner loop)

Meta Learning

- Standard Learning
  - Learner: $y = f_\theta(x)$
    - Once θ is learned, the output of $f(x)$ is never changed
  - Training:
    - A single task of massive data pair $T = \{(x_i, y_i)\}$, SGD is used for training
- Meta-Learning
  - Meta-Leaner: $y = f_\theta(x|S_T)$
    - f_θ can further adapt with additional samples in S_T
  - Meta-Training:
    - A large collection of tasks is provided $\mathcal{D} = \{T_i\}$
    - Use SGD to learn meta-learner on each T_i (outer loop)
    - Meta-learner $f_\theta(x|S)$ must adapt quickly with S_{T_i} (inner loop)

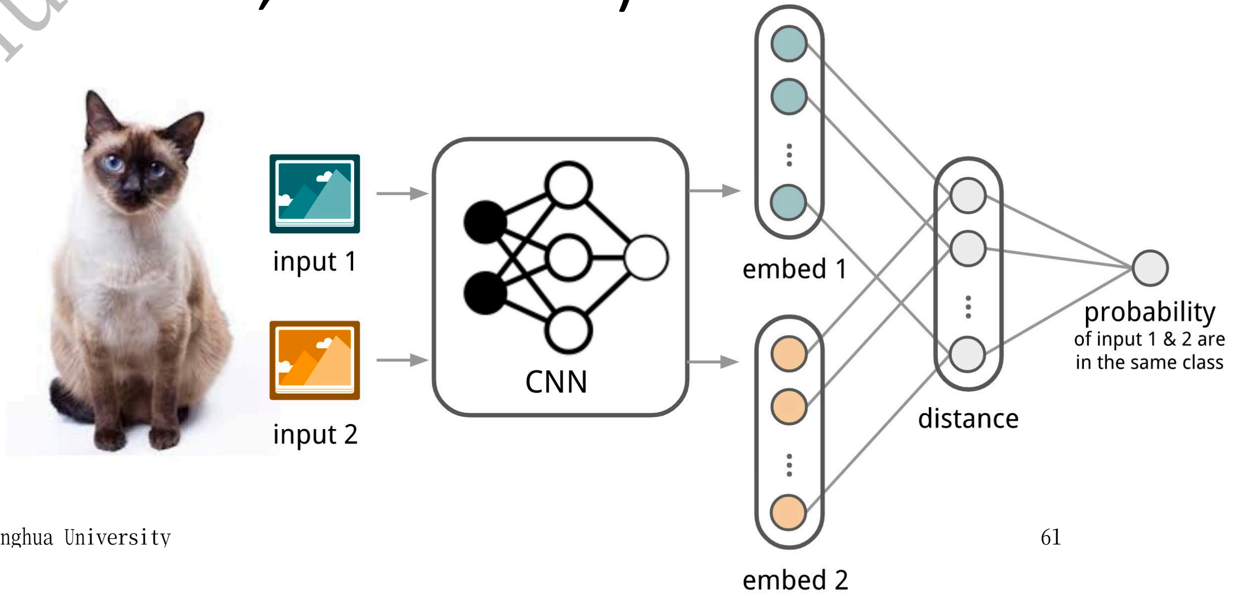
Goal: A neural module that can quickly adapt

Representation of Meta-Learner

- Metric Learning
  - Idea: use a nearest neighbor classifier as $f_\theta(x|S)$
    - $k^\star = \arg\max_i D\left(x, x_i^S\right), \quad f_\theta(x|S) = y_{k^\star}$
  - Goal: learn a good similarity metric $D(x, x')$

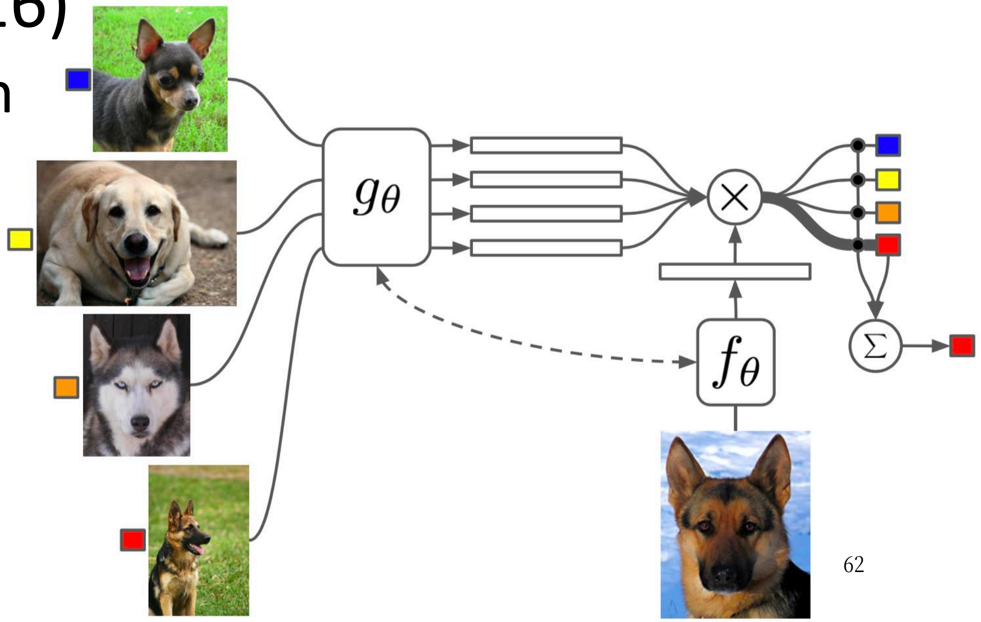
Representation of Meta-Learner

- Metric Learning
  - Idea: use a nearest neighbor classifier as $f_\theta(x|S)$
    - $k^\star = \arg\max_i D\left(x, x_i^S\right), \quad f_\theta(x|S) = y_{k^\star}$
  - Goal: learn a good similarity metric $D(x, x')$
- Siamese Neural Network (Koch, Ruslan et al, ICML 2015)
  - $f(x, y) \rightarrow \sigma(W|\phi(x) - \phi(y)|)$
  - Predict the probability of same class
  - Few-shot learning
    - x_i^S: support example for class i
    - $k = \arg\max_i f\left(x, x_i^S\right)$



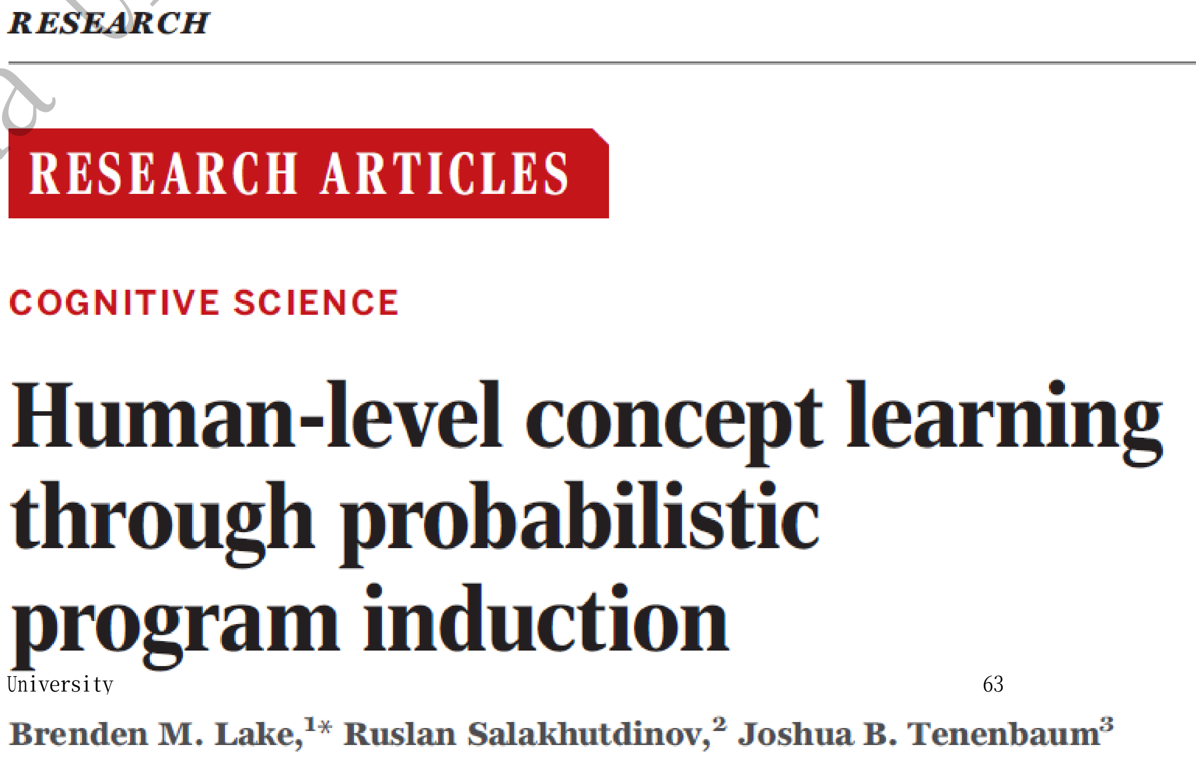
Representation of Meta-Learner

- Metric Learning
  - Idea: use a nearest neighbor classifier as $f_\theta(x|S)$
    - $k^\star = \arg\max_i D\left(x, x_i^S\right), \quad f_\theta(x|S) = y_{k^\star}$
  - Goal: learn a good similarity metric $D(x, x')$
- Matching Network (Vinyals et al, NIPS 2016)
  - Main idea: soft nearest neighbor by attention
    - $g_\theta\left(x_i^S\right)$: support embeddings
    - $f_\theta(x)$: query embedding
    - $\alpha_i = \text{softmax}\left(g_i^T f\right)$ & $y = \sum_i \alpha_i y_i$
  - Enhancement
    - $f_\theta = f_\theta(x, S) = LSTM_\theta\ (S, x)$
    - $g_\theta = g_\theta\left(x_i^S, S\right) = LSTM_\theta\left(S, x_i^S\right)$



Representation of Meta-Learner

- Bayesian Inference
  - Posterior via Bayes Rule gives a perfect few-shot learning method
    - Gibbs sampling, MCMC, Variational Inference, etc
  - Meta-Training: $P_\theta(x, y)$
    - Learning a Bayesian model
    - It should allow simple Bayesian Inference
      - NO SGD!!! We only have a few data!
  - Adaptation: $P_\theta(y|x, S^T)$
    - Posterior sampling via Bayes rule
- Probabilistic Programming!
  - Universal Bayesian modeling tool
  - Black-Box Inference


RESEARCH

RESEARCH ARTICLES

COGNITIVE SCIENCE

Human-level concept learning through probabilistic program induction

Brenden M. Lake,[1*] Ruslan Salakhutdinov,[2] Joshua B. Tenenbaum[3]

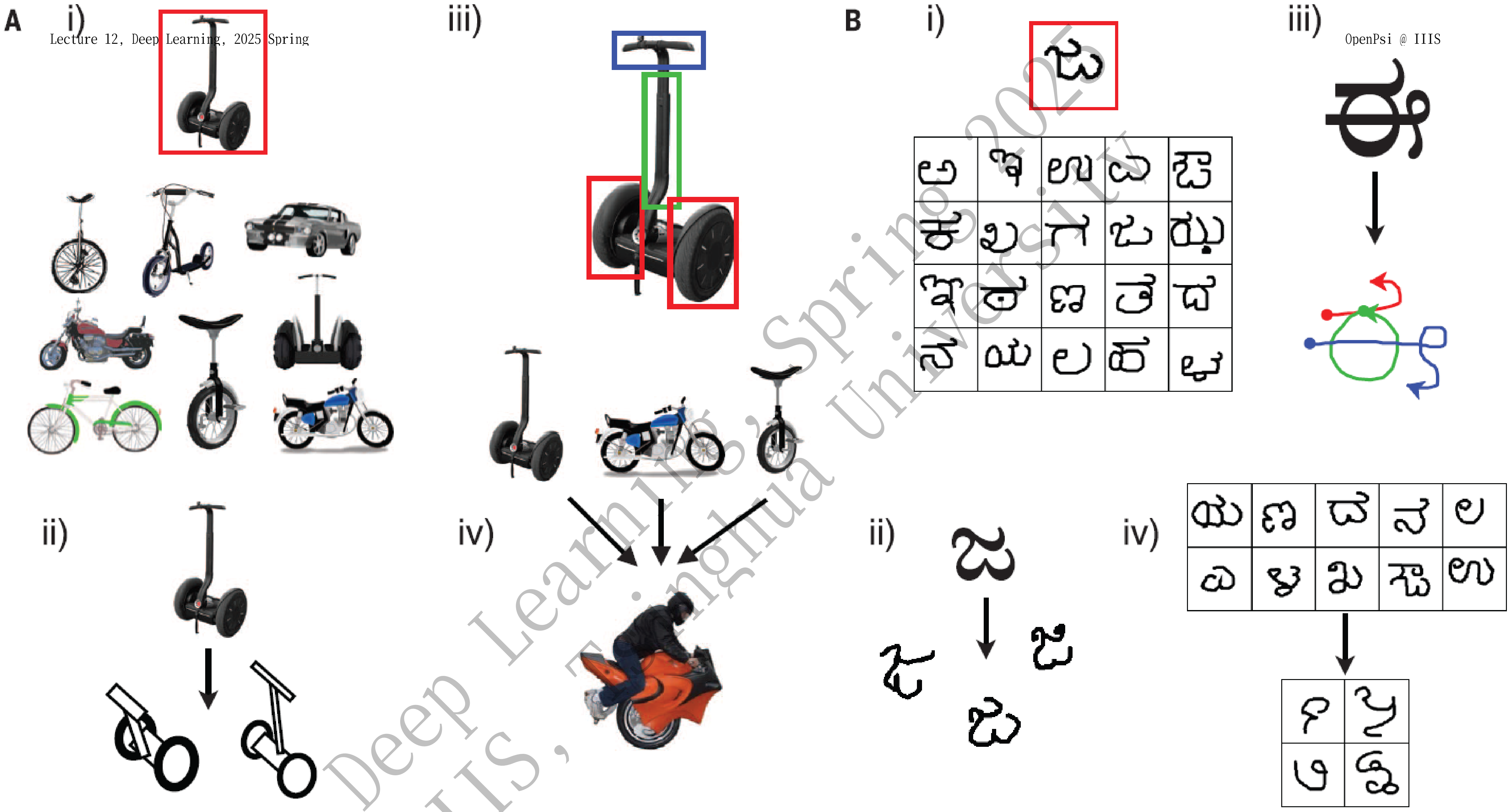
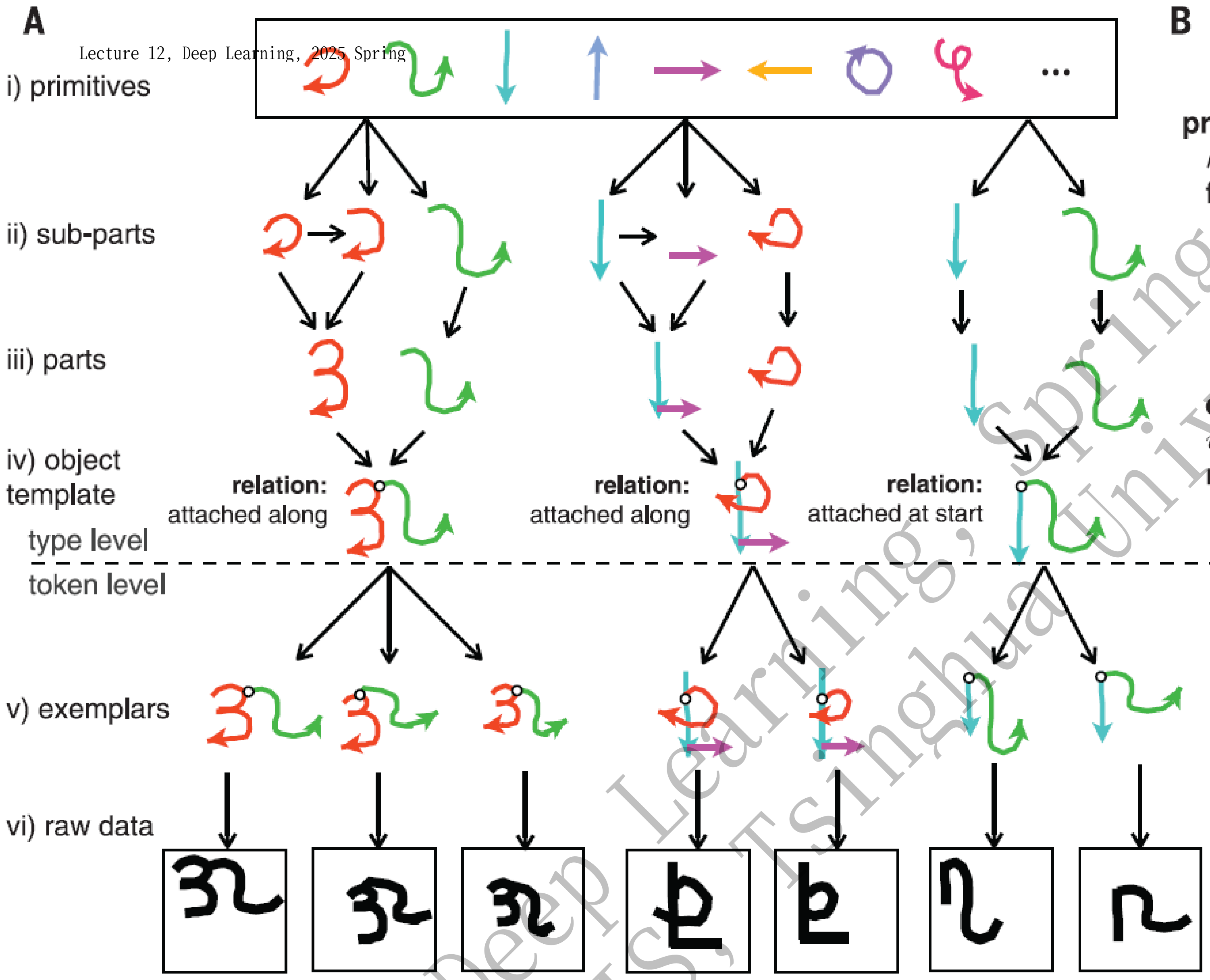


Fig. 1. People can learn rich concepts from limited data. (**A** and **B**) A single example of a new concept (red boxes) can be enough information to support the (i) classification of new examples, (ii) generation of new examples, (iii) parsing an object into parts and relations (parts segmented by color), and (iv) generation of new concepts from related concepts. [Image credit for (A), iv, bottom: With permission from Glenn Roberts and Motorcycle Mojo Magazine]

A



B

procedure GenerateType
 $\kappa \leftarrow P(\kappa)$ ▷ Sample number of parts
 for $i = 1 \ldots \kappa$ **do**
  $n_i \leftarrow P(n_i|\kappa)$ ▷ Sample number of sub-parts
  for $j = 1 \ldots n_i$ **do**
   $s_{ij} \leftarrow P(s_{ij}|s_{i(j-1)})$ ▷ Sample sub-part sequence
  end for
  $R_i \leftarrow P(R_i|S_1, ..., S_{i-1})$ ▷ Sample relation
 end for
 $\psi \leftarrow \{\kappa, R, S\}$
 return @GenerateToken(ψ) ▷ Return program

procedure GenerateToken(ψ)
 for $i = 1 \ldots \kappa$ **do**
  $S_i^{(m)} \leftarrow P(S_i^{(m)}|S_i)$ ▷ Add motor variance
  $L_i^{(m)} \leftarrow P(L_i^{(m)}|R_i, T_1^{(m)}, ..., T_{i-1}^{(m)})$
   ▷ Sample part's start location
  $T_i^{(m)} \leftarrow f(L_i^{(m)}, S_i^{(m)})$ ▷ Compose a part's trajectory
 end for
 $A^{(m)} \leftarrow P(A^{(m)})$ ▷ Sample affine transform
 $I^{(m)} \leftarrow P(I^{(m)}|T^{(m)}, A^{(m)})$ ▷ Sample image
 return $I^{(m)}$

Fig. 3. A generative model of handwritten characters. (**A**) New types are generated by choosing primitive actions (color coded) from a library (i), combining these subparts (ii) to make parts (iii), and combining parts with relations to define simple programs (iv). New tokens are generated by running these programs (v), which are then rendered as raw data (vi). (**B**) Pseudocode for generating new types ψ and new token images $I^{(m)}$ for $m = 1, ..., M$. The function $f(\cdot, \cdot)$ transforms a subpart sequence and start location into a trajectory.

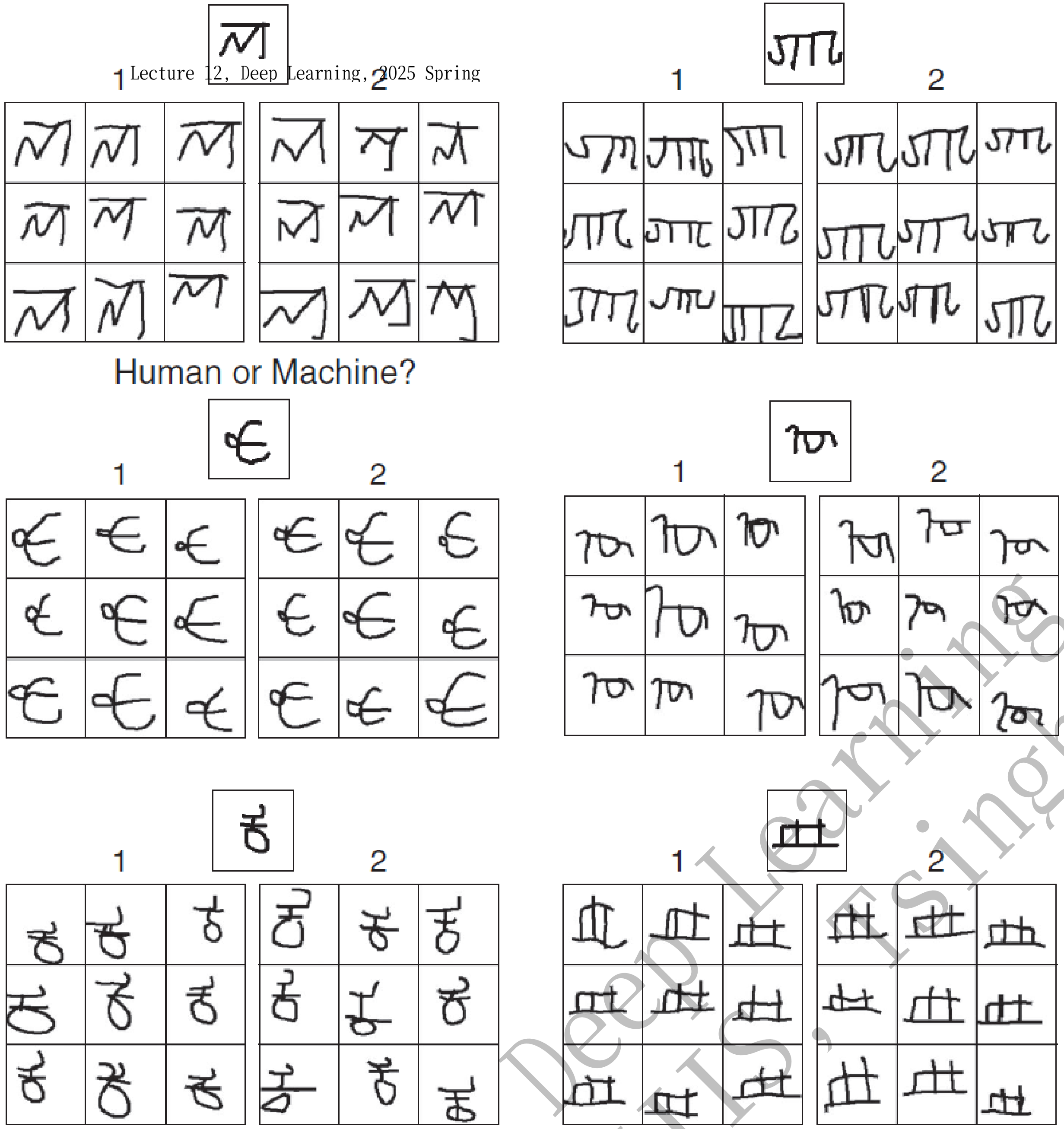
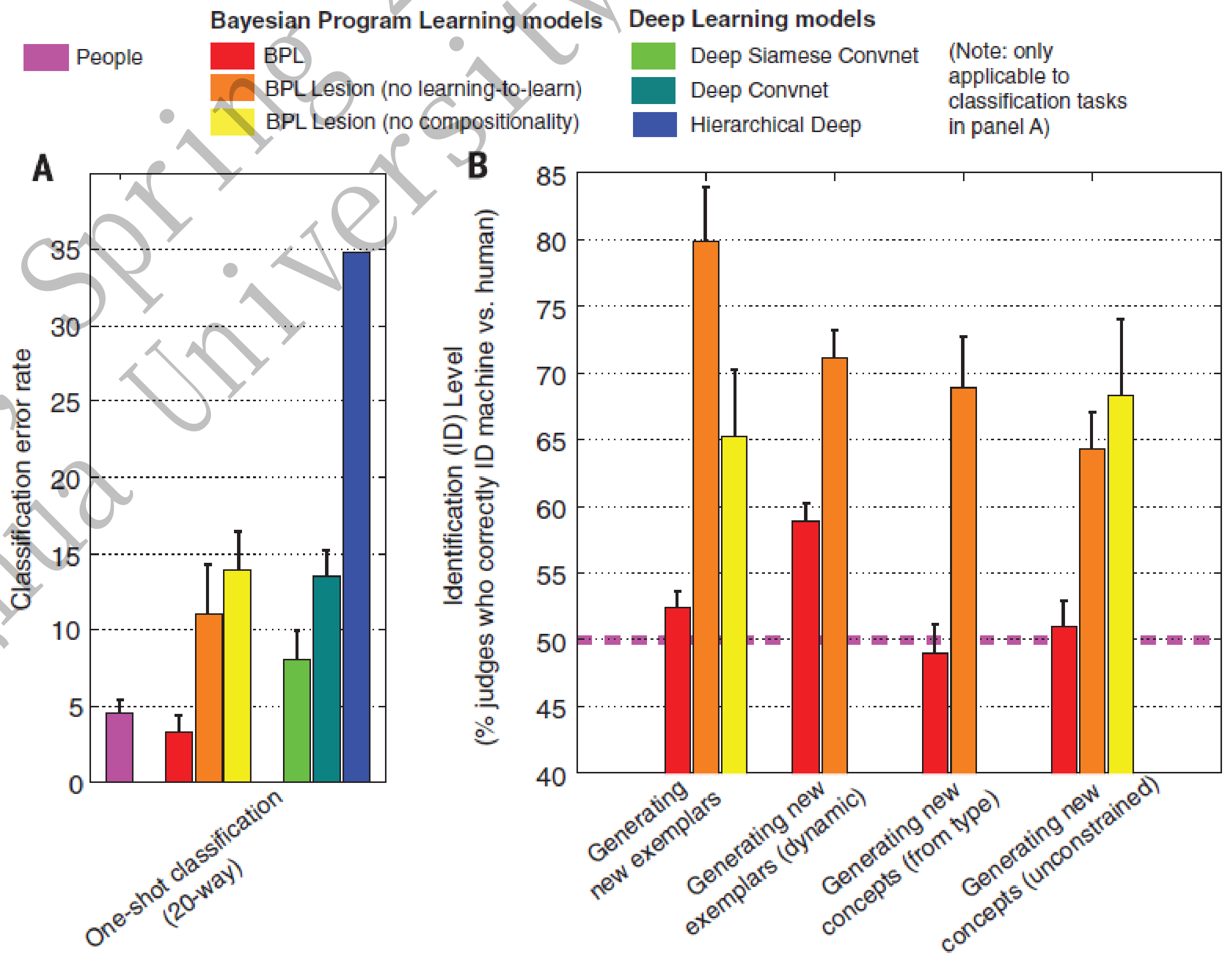


Fig. 5. Generating new exemplars. Humans and machines were given an image of a novel character (top) and asked to produce new exemplars. The nine-character grids in each pair that were generated by a machine are (by row) 1, 2; 2, 1; 1, 1.

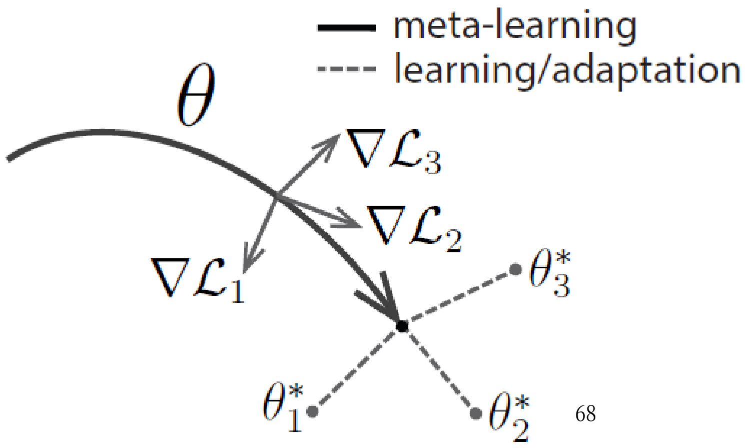


Representation of Meta-Learner

- Gradient Descent
  - Given new few-shot training data S, we fine-tune θ using SGD
  - $f_\theta(x|S) = g_{\theta^\star}(x)$
    - $\theta^\star = SGD\left(\theta, \eta, \left\{x_i^S, y_i^S\right\}\right)$
    - $\theta^{k+1} = \theta^k - \eta \cdot \nabla L\left(S, \theta^k\right)$
    - $\theta^0 = \theta$
  - Issue: typically, SGD converges in a large number of iterations
    - Overfitting?
      - We only have a few samples but want to fine-tune a deep net
    - Meta-training?
      - What are we going to learn?

Representation of Meta-Learner

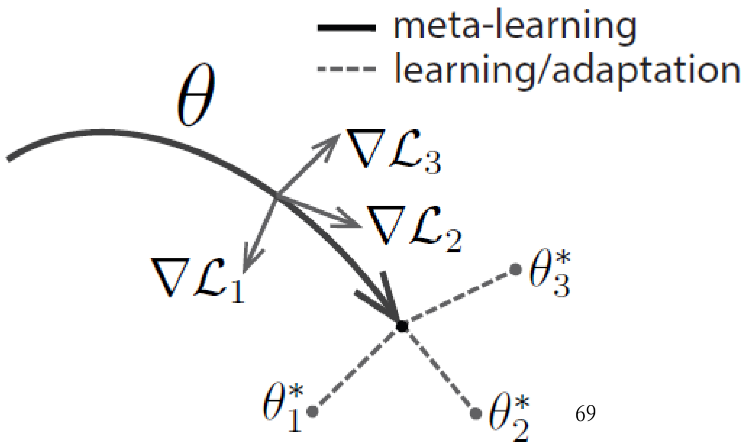
- Gradient Descent
  - Given new few-shot training data S, we fine-tune θ using SGD
- Model-Agnostic Meta-Learning (MAML, Finn et al, ICML 2017)
  - Goal: learn a good parameter initialization θ
    - Such that θ is close to optimal $\theta_i^\star$ via gradient steps in each test task T_i
  - Meta-Training
    - $\theta^\star = \arg\min_\theta L(B_i, SGD(\theta, S_i))$
  - Meta-Testing on $T = (S, B)$
    - $\theta' = SGD(\theta^\star, S)$
    - Evaluation: $L(B, g_{\theta'})$



Representation of Meta-Learner

- Gradient Descent
  - Given new few-shot training data S, we fine-tune θ using SGD
- Model-Agnostic Meta-Learning (MAML, Finn et al, ICML 2017)
  - Goal: learn a good parameter initialization θ
    - Such that θ is close to optimal $\theta_i^\star$ via gradient steps in each test task T_i
  - Meta-Training
    - $\theta^\star = \arg\min_\theta L(B_i, SGD(\theta, S_i))$
  - Meta-Testing on $T = (S, B)$
    - $\theta' = SGD(\theta^\star, S)$
    - Evaluation: $L(B, g_{\theta'})$

Backpropagation over SGD process??

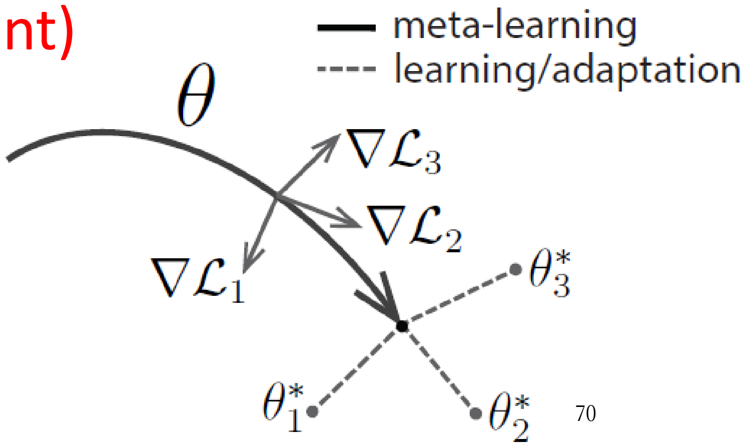


Representation of Meta-Learner

- Gradient Descent
  - Given new few-shot training data S, we fine-tune θ using SGD
- Model-Agnostic Meta-Learning (MAML, Finn et al, ICML 2017)
  - Goal: learn a good parameter initialization θ
    - Such that θ is close to optimal $\theta_i^\star$ via gradient steps in each test task T_i
  - Meta-Training
    - $\theta^\star = \arg\min_\theta L(B_i, \theta - \eta\nabla L(S_i, \theta))$

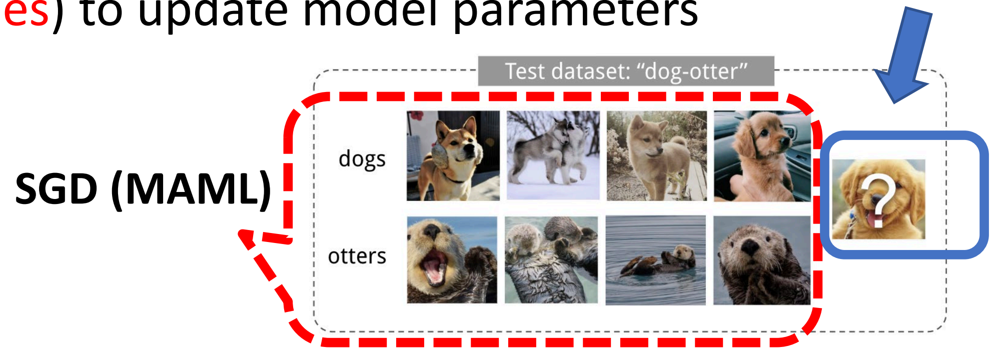
1-step GD approximation
(gradient of gradient)

  - Meta-Testing on $T = (S, B)$
    - $\theta' = SGD(\theta^\star, S)$
    - Evaluation: $L(B, g_{\theta'})$



Representation of Meta-Learner

- Gradient Descent
  - Given new few-shot training data S, we fine-tune θ using SGD
  - $f_\theta(x|S) = g_{\theta^\star}(x)$
    - $\theta^\star = SGD\left(\theta, \eta, \{x_i^S, y_i^S\}\right)$
    - $\theta^{k+1} = \theta^k - \eta \cdot \nabla L\left(S, \theta^k\right)$
    - $\theta^0 = \theta$
  - MAML: learn good initializations for fast SGD adaptation
    - Adaptation: use SGD on S (meta-training samples) to update model parameters
    - Can we even adapt on B (meta-test samples)?
      - We do not have label on test data…
      - **Self-supervised learning!**
        - Improve feature representation

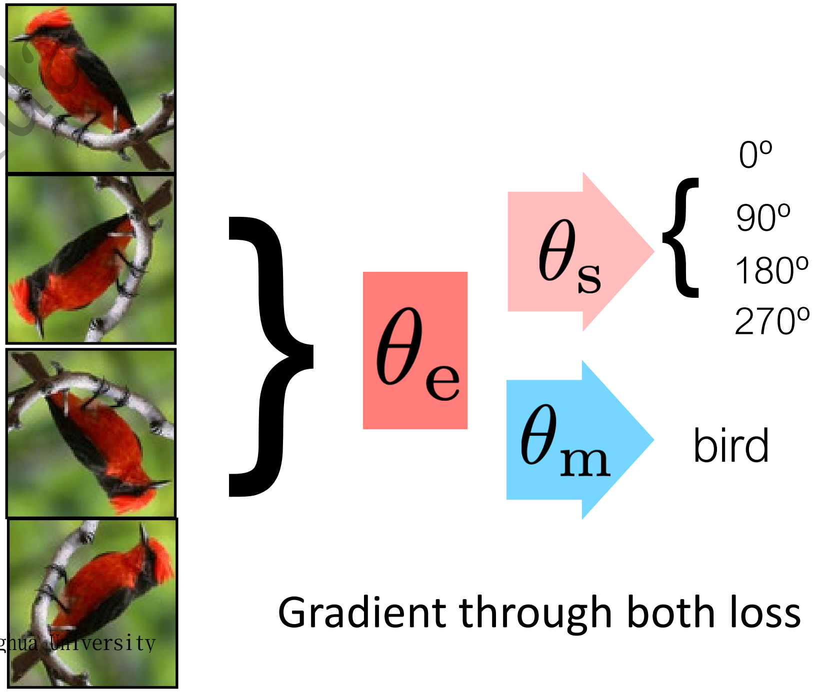


Representation of Meta-Learner

- Test-Time Training (Sun et al, ICML 2020)
  - $L(X, \theta_e, \theta_m, \theta_s) = L_m(X, \theta_e, \theta_m) + L_s(X, \theta_e, \theta_s)$
    - L_m: labeled loss; L_s: self-supervised loss
    - $f(x; \theta_e)$ to extract features for down-stream tasks

training

$$\min_{\theta_\mathrm{e}, \theta_\mathrm{s}, \theta_\mathrm{m}} \mathbb{E}_P \left[\begin{array}{c} \ell_\mathrm{m}(x, y; \theta_\mathrm{e}, \theta_\mathrm{m}) \\ + \ell_s(x, y_\mathrm{s}; \theta_e, \theta_s) \end{array} \right]$$

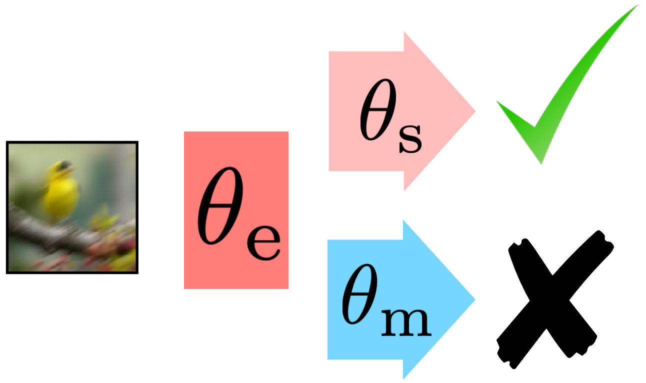


Gradient through both loss heads (train)

Representation of Meta-Learner

- Test-Time Training (Sun et al, ICML 2020)
  - $L(X, \theta_e, \theta_m, \theta_s) = L_m(X, \theta_e, \theta_m) + L_s(X, \theta_e, \theta_s)$
    - L_m: labeled loss (only inference); L_s: self-supervised loss
    - $f(x; \theta_e)$ to extract features for down-stream tasks

testing

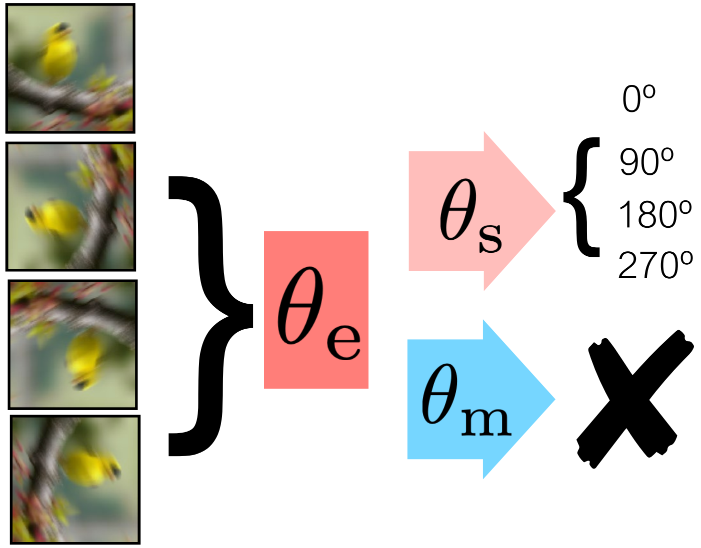


Representation of Meta-Learner

- Test-Time Training (Sun et al, ICML 2020)
  - $L(X,\theta_e,\theta_m,\theta_s) = L_m(X,\theta_e,\theta_m) + L_s(X,\theta_e,\theta_s)$
    - L_m: labeled loss (only inference); L_s: self-supervised loss
    - $f(x;\theta_e)$ to extract features for down-stream tasks

testing

$$\min_{\theta_e,\theta_s} \mathbb{E}_Q\left[\ell_s(x,y_s;\theta_e,\theta_s)\right]$$



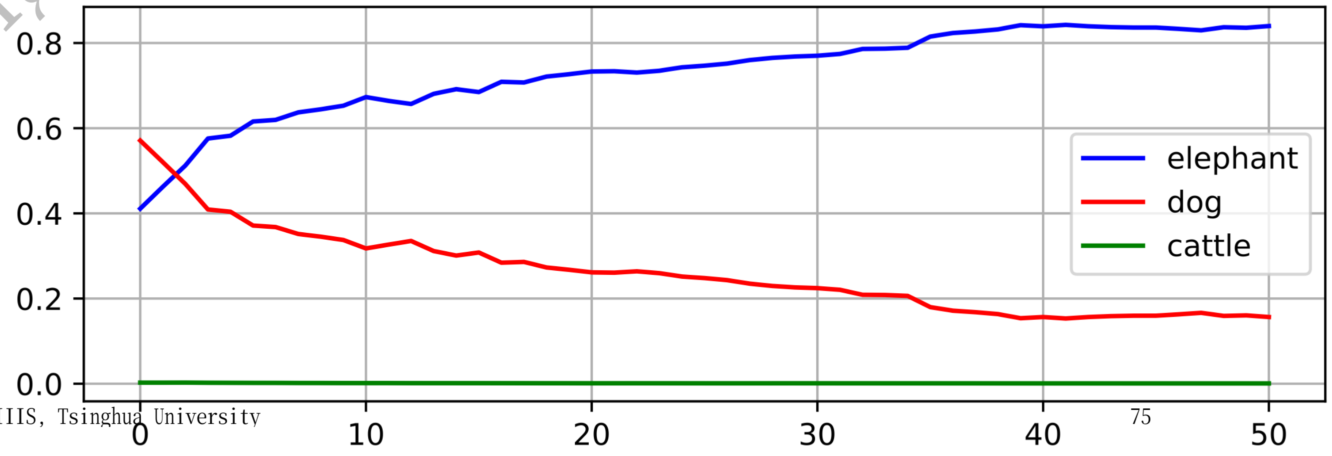
Gradient through only SL head (test-time)

Representation of Meta-Learner

- Test-Time Training (Sun et al, ICI
  - $L(X, \theta_e, \theta_m, \theta_s) = L_m(X, \theta_e, \theta_m)$
    - L_m: labeled loss (only inference); L_s
    - $f(x; \theta_e)$ to extract features for dow

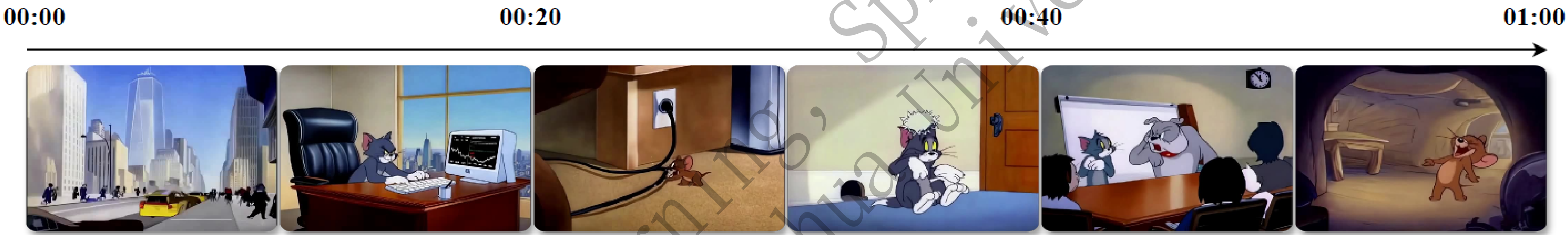
testing

$$\min_{\theta_e, \theta_s} \mathbb{E}_Q \left[\ell_s(x, y_s; \theta_e, \theta_s) \right]$$

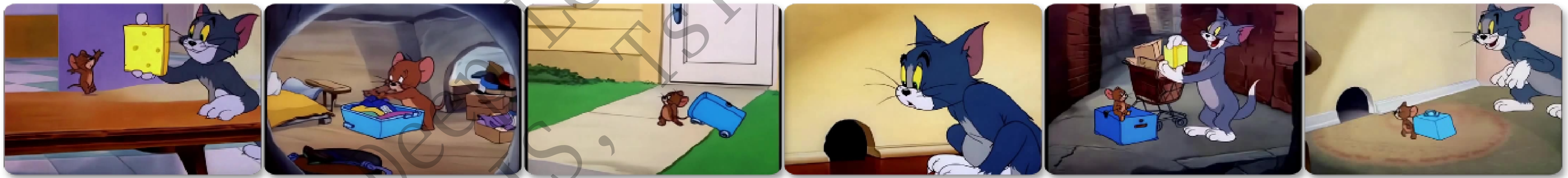


Representation of Meta-Learner

- One-Minute Video Generation with Test-Time Training (CVPR 2025)



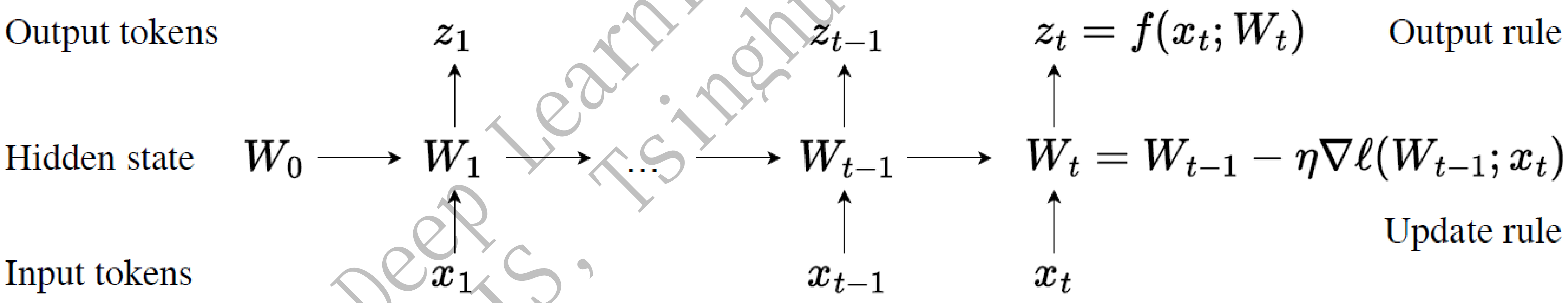
On a sunny morning in New York, Tom, a blue-gray cat carrying a briefcase, arrives at his office in the World Trade Center. As he settles in, his computer suddenly shuts down – Jerry, a mischievous brown mouse, has chewed the cable. A chase ensues, ending with Tom crashing into the wall as Jerry escapes into his mousehole. Determined, Tom bursts through an office door, accidentally interrupting a meeting led by Spike, an irritated bulldog, who angrily sends him away. Safe in his cozy mousehole, Jerry laughs at the chaos.



Jerry happily eats cheese in a tidy kitchen until Tom playfully takes it away, teasing him. Annoyed, Jerry packs his belongings and leaves home, dragging a small suitcase behind him. Later, Tom notices Jerry's absence, feels sad, and follows Jerry's tiny footprints all the way to San Francisco. Jerry sits disheartened in an alleyway, where Tom finds him, gently offering cheese as an apology. Jerry forgives Tom, accepts the cheese, and the two return home together, their friendship restored.

Representation of Meta-Learner

- One-Minute Video Generation with Test-Time Training (CVPR 2025)
  - General test-time training by introducing a TTT layer
  - Given a predefined prediction loss $l(W_{t-1}; x_t)$
  - A TTT layer updates the hidden state with using the gradient ∇L



Representation of Meta-Learner

- One-Minute Video Generation with Test-Time Training (CVPR 2025)
  - General test-time training by introducing a TTT layer
  - TTT is fine-tuned for a video model with local attention

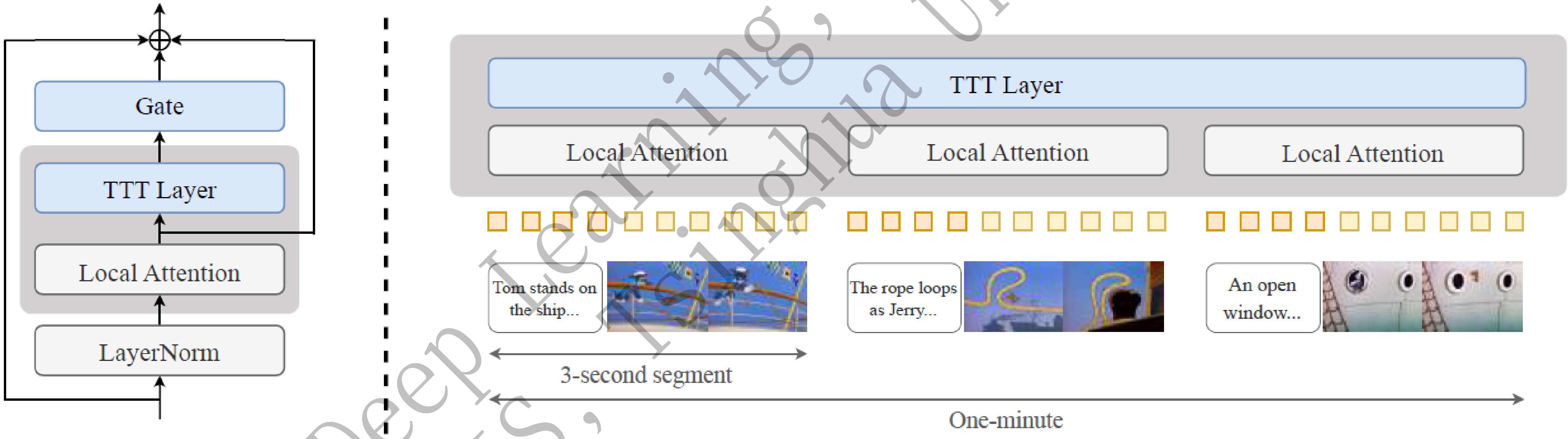


Figure 3. Overview of our approach. **Left**: Our modified architecture adds a TTT layer with a learnable gate after each attention layer. See Subsection 3.1. **Right**: Our overall pipeline creates input sequences composed of 3-second segments. This structure enables us to apply self-attention layers locally over segments and TTT layers globally over the entire sequence. See Subsection 3.2.

Representation of Meta-Learner

- One-Minute Video Generation with Test-Time Training (CVPR 2025)
  - General test-time training by introducing a TTT layer
  - TTT is fine-tuned for a video model with local attention
  - Finetuning with 7 hours of Tom-Jerry video (56 hours of 256 H100)

Representation of Meta-Learner

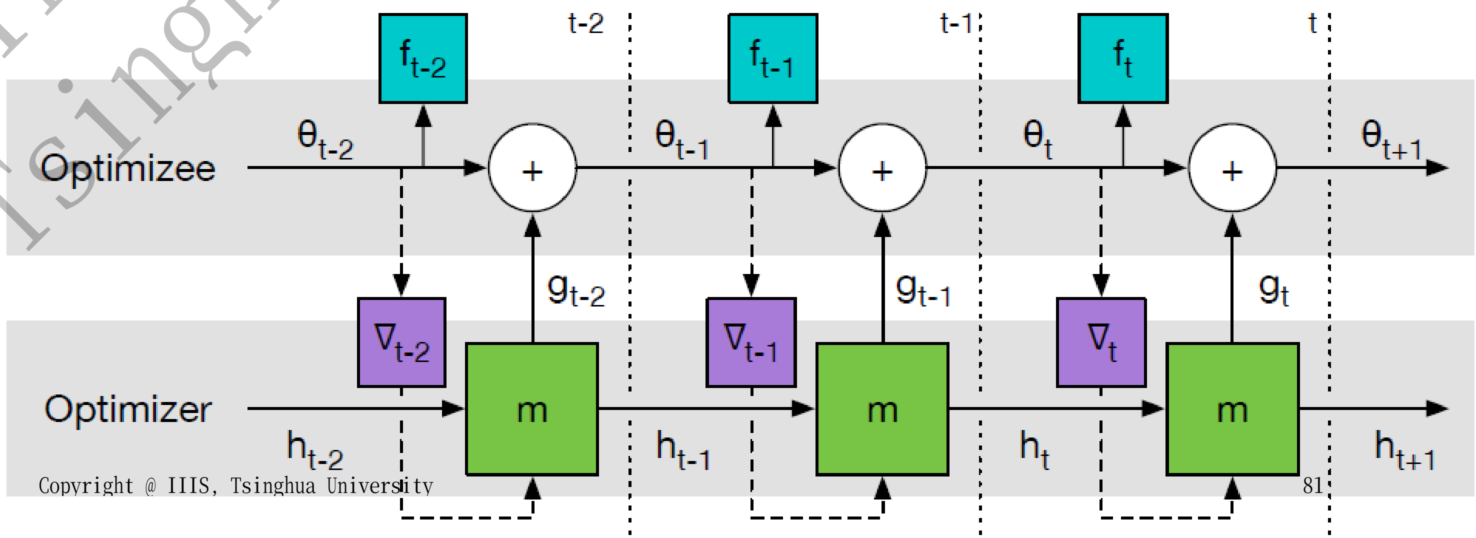
- Gradient Descent
  - Given new training data S, we fine-tune θ using SGD
  - $f_\theta(x|S) = g_{\theta^\star}(x)$
    - $\theta^\star = SGD\left(\theta, \eta, \{x_i^S, y_i^S\}\right)$
    - $\theta^{k+1} = \theta^k - \eta \cdot \nabla L\left(S, \theta^k\right)$
    - $\theta^0 = \theta$
  - MAML: learn good initializations for fast SGD adaptation
    - Adaptation: use SGD to update model parameters
  - Test-time-training
    - Use self-supervision to even adapt with unlabeled test data
- Can we learn an model to output model parameters instead of SGD?
  - i.e., learning an "SGD" algorithm

Learning to Learn

- Learning to learn by gradient descent by gradient descent (Marcin Andrychowicz et al, DeepMind, NIPS 2016)
  - Learning an LSTM optimizer $m_\phi(\nabla_\theta)$ to produce a increment on θ
  - $\theta^{k+1} = \theta + m_\phi(\nabla_\theta)$ where $\nabla_\theta = \frac{\partial L(X,Y;\theta)}{\partial \theta}$
  - Meta-Loss: $L(\phi; \theta^0, X, Y)$
    - Gradient over gradient
    - Recursive gradient
  - Gradient preprocessing

$$\nabla^k \rightarrow \begin{cases} \left(\frac{\log(|\nabla|)}{p}, \operatorname{sgn}(\nabla)\right) & \text{if } |\nabla| \geq e^{-p} \\ (-1, e^p \nabla) & \text{otherwise} \end{cases}$$

$p = 10$



Learning to Learn

- Learning to learn by gradient descent by gradient descent (Marcin Andrychowicz et al, DeepMind, NIPS 2016)
  - Learning an LSTM optimizer $m_\phi(\nabla_\theta)$ to produce a increment on θ

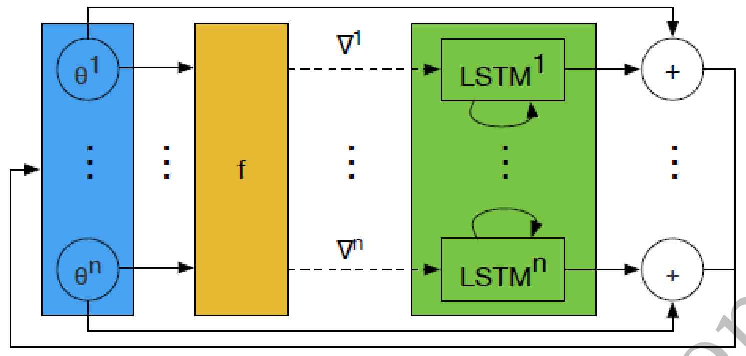


Figure 3: One step of an LSTM optimizer. All LSTMs have shared parameters, but separate hidden states.

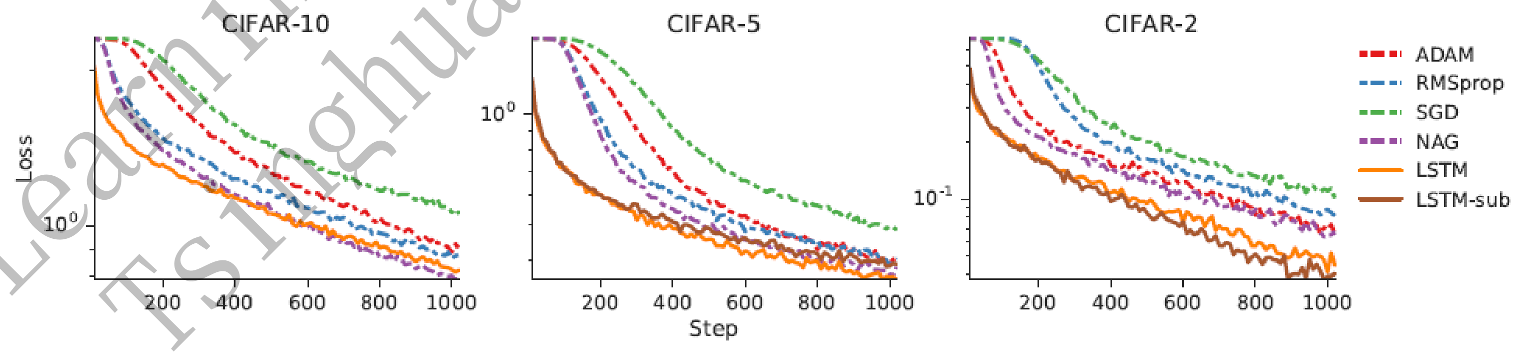
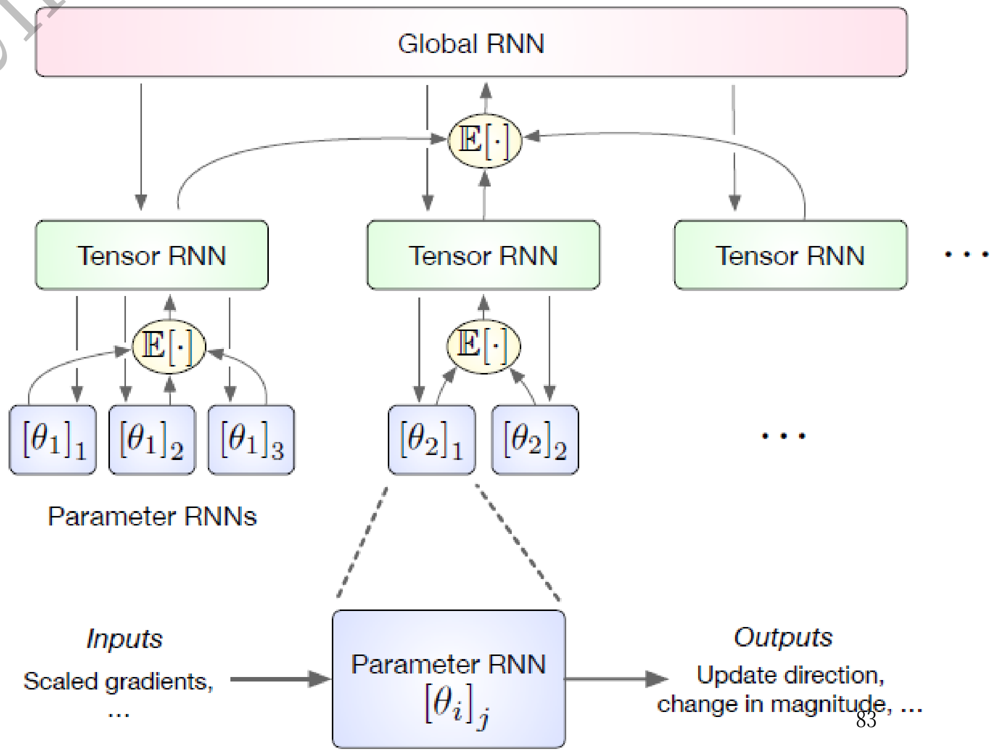


Figure 7: Optimization performance on the CIFAR-10 dataset and subsets. Shown on the left is the LSTM optimizer versus various baselines trained on CIFAR-10 and tested on a held-out test set. The two plots on the right are the performance of these optimizers on subsets of the CIFAR labels. The additional optimizer *LSTM-sub* has been trained only on the heldout labels and is hence transferring to a completely novel dataset.

Learning to Learn

- More advances on learning to optimize
  - Learned Optimizers that Scale and Generalize (Google Brain, ICML 2017)
    - Work with ResNet
    - Hierarchical RNN architectures
    - Non-trivial scaling & momentum
    - Using a lot of engineered optimization features



Learning to Learn

- More advances on learning to optimize
  - Learned Optimizers that Scale and Generalize (Google Brain, ICML 2017)
  - Neural Optimizer Search with Reinforcement Learning (Quoc Le at al, ICML 2017, Google Brain)
    - Use RL to learn symbolic optimizer on CIFAR10
    - Work well on NLP tasks and ImageNet

Optimizer	Train perplexity	Test BLEU
Adam	1.49	24.5
PowerSign	**1.39**	**25.0**

Optimizer	Top-1 Accuracy	Top-5 Accuracy
RMSProp	73.5	91.5
PowerSign-cd	**73.9**	**91.9**
AddSign-ld	73.8	91.6

Table 3. Performance of our PowerSign and AddSign optimizers against RMSProp on a state-of-the art MobileNet baseline (Zoph et al., 2017). All optimizers are applied with cosine learning rate decay.

Optimizer	Best Test	Final Test
SGD	93.0	92.3
Momentum	93.0	92.2
Adam	92.6	92.3
RMSProp	92.3	91.6
PowerSign	93.0	92.4
PowerSign-ld	93.6	93.4
PowerSign-cd	93.7	93.1
PowerSign-rd$_{10}$	94.2	92.6
PowerSign-rd$_{20}$	94.4	92.0
AddSign	93.0	92.6
AddSign-ld	93.5	92.0
AddSign-cd	93.6	92.4
AddSign-rd$_{10}$	94.2	94.0
AddSign-rd$_{20}$	**94.4**	**94.3**

Learning to Learn

- More advances on learning to c
  - Learned Optimizers that Scale ar
  - Neural Optimizer Search with Re 2017, Google Brain)
    - Use RL to learn symbolic optimizer
    - Work well on NLP tasks and Image

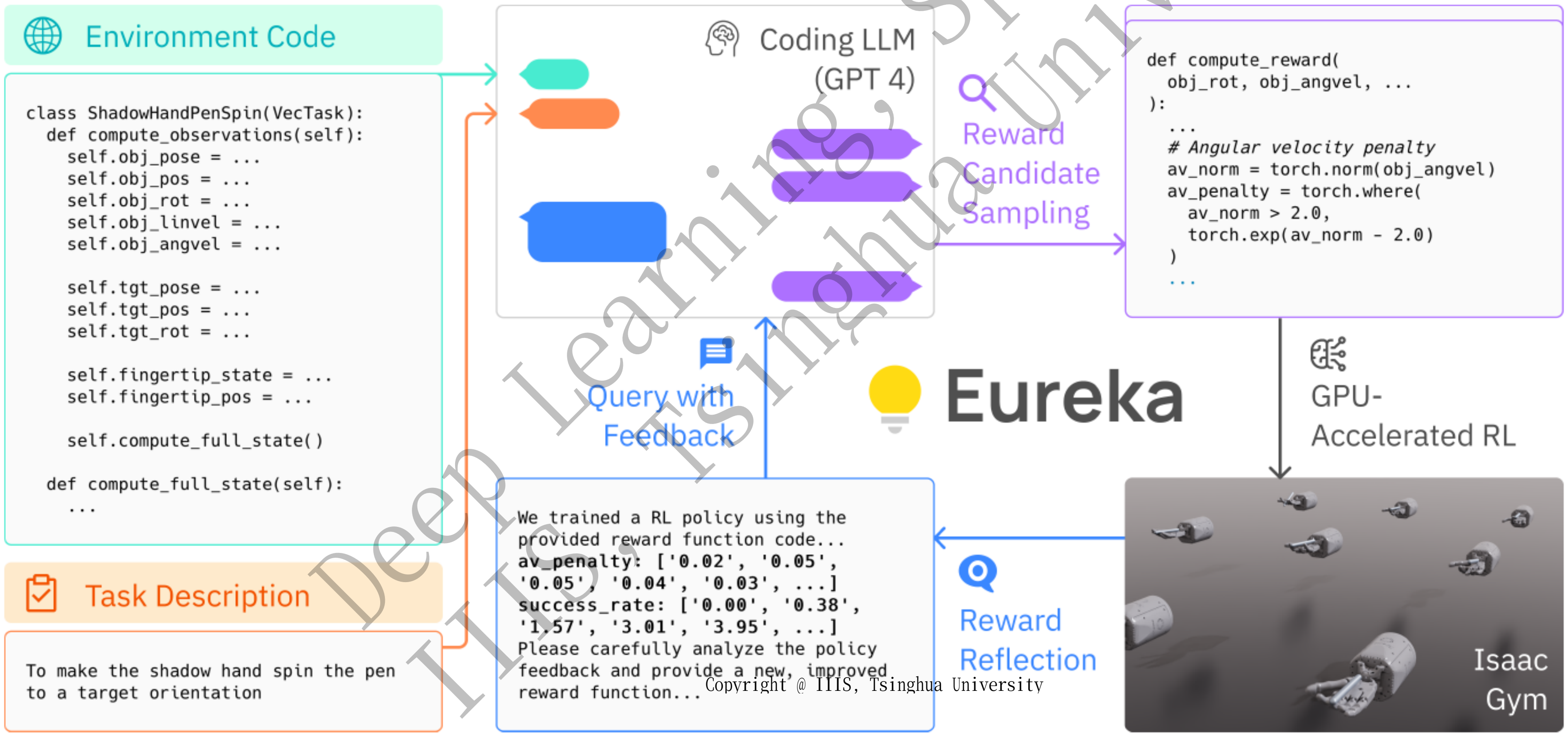
Optimizer	Final Val	Final Test	Best Val	Best Test
SGD	92.0	91.8	92.9	91.9
Momentum	92.7	92.1	93.1	92.3
Adam	90.4	90.1	91.8	90.7
RMSProp	90.7	90.3	91.4	90.3
$\hat{m} * (1+\epsilon) * sigmoid(10^{-4}w)$	90.6	90.6	93.1	92.2
$\text{sign}(m) * \sqrt{\lvert g \rvert}$	92.2	91.8	92.9	92.2
$\text{sign}(g) * \text{sign}(m) * \hat{m}$	91.2	91.0	92.4	91.3
$\text{sign}(m) * \sqrt{\lvert g \rvert}$	91.7	91.1	92.3	91.6
$(1+\text{sign}(g) * \text{sign}(m)) * \text{sign}(g)$	91.3	90.4	91.9	91.1
$(1+\text{sign}(g) * \text{sign}(m)) * \text{sign}(m)$	91.0	90.6	92.0	90.8
$\text{sign}(g) * \sqrt{\lvert \hat{m} \rvert}$	90.7	90.6	91.5	90.6
$\sqrt{\lvert g \rvert} * \hat{m}$	92.0	90.9	93.6	93.1
$\sqrt{\lvert g \rvert} * g$	92.6	91.9	93.2	92.3
$(1+\text{sign}(g) * \text{sign}(m)) * \hat{m}$	91.8	91.3	92.6	91.8
$(1+\text{sign}(g) * \text{sign}(m)) * \text{RMSProp}$	92.0	92.1	92.9	92.4
$(1+\text{sign}(g) * \text{sign}(m)) * \text{Adam}$	91.2	91.2	92.2	91.9
$[e^{\text{sign}(g)*\text{sign}(m)} + \text{clip}(g, 10^{-4})] * g$	92.5	92.4	93.8	93.1
$\text{clip}(\hat{m}, 10^{-4}) * e^{\hat{v}}$	93.5	92.5	93.8	92.7
$\hat{m} * e^{\hat{v}}$	93.1	92.4	93.8	92.6
$g * e^{\text{sign}(g)*\text{sign}(m)}$	93.1	92.8	93.8	92.8
$\text{drop}(g, 0.3) * e^{\text{sign}(g)*\text{sign}(m)}$	92.7	92.2	93.6	92.7
$\hat{m} * e^{g^2}$	93.1	92.5	93.6	92.4
$\text{drop}(\hat{m}, 0.1)/(e^{g^2} + \epsilon)$	92.6	92.4	93.5	93.0
$\text{drop}(g, 0.1) * e^{\text{sign}(g)*\text{sign}(m)}$	92.8	92.4	93.5	92.2
$\text{clip}(\text{RMSProp}, 10^{-5}) + \text{drop}(\hat{m}, 0.3)$	90.8	90.8	91.4	90.9
$\text{Adam} * e^{\text{sign}(g)*\text{sign}(m)}$	92.6	92.0	93.4	92.0
$\text{Adam} * e^{\hat{m}}$	92.9	92.8	93.3	92.7
$g + \text{drop}(\hat{m}, 0.3)$	93.4	92.9	93.7	92.9
$\text{drop}(\hat{m}, 0.1) * e^{g^3}$	92.8	92.7	93.7	92.8
$g - \text{clip}(g^2, 10^{-4})$	93.4	92.8	93.7	92.8
$e^g - e^{\hat{m}}$	93.2	92.5	93.5	93.1
$\text{drop}(\hat{m}, 0.3) * e^{10^{-3}w}$	93.2	93.0	93.5	93.2

Learning to Learn

- More advances on learning to optimize
  - Learned Optimizers that Scale and Generalize (Google Brain, ICML 2017)
  - Neural Optimizer Search with Reinforcement Learning (Quoc Le at al, ICML 2017, Google Brain)
    - Use RL to learn symbolic optimizer on CIFAR10
    - Work well on NLP tasks and ImageNet
- Even learning a loss function for RL!
  - RL^2 (OpenAI, 2017): learning an LSTM policy updater (instead of SGD)
  - Evolved Policy Gradient (OpenAI, 2018): evolution algorithm to learn a neural loss function for RL to run SGD (instead of policy gradient)
  - Meta-Gradient Reinforcement Learning (DeepMind 2020): discover a new symbolic Q-learning objective

Use LLM as a Pretrained Meta-Learner

- Eureka: Human-Level Reward Design via LLMs (ICLR 2024, Nvidia)
  - Give the code and execution result to LLM, and the LLM will make it work!

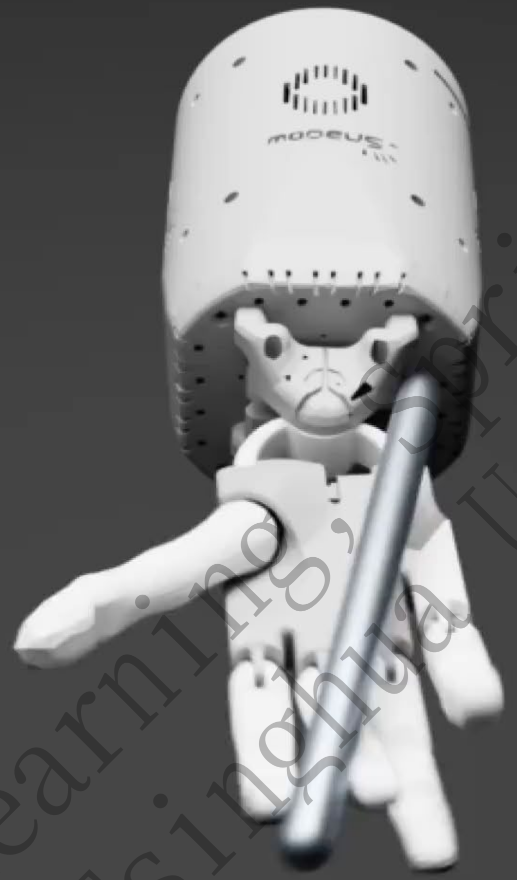


```
def compute_reward(object_rot, goal_rot, object_angvel, object_pos, fingertip_pos):
    # Rotation reward
    rot_diff = torch.abs(torch.sum(object_rot * goal_rot, dim=1) - 1) / 2
-   rotation_reward_temp = 20.0
+   rotation_reward_temp = 30.0
    rotation_reward = torch.exp(-rotation_reward_temp * rot_diff)

    # Distance reward
+   min_distance_temp = 10.0
    min_distance = torch.min(torch.norm(fingertip_pos - object_pos[:, None], dim=2), dim=1).values
-   distance_reward = min_distance
+   uncapped_distance_reward = torch.exp(-min_distance_temp * min_distance)
+   distance_reward = torch.clamp(uncapped_distance_reward, 0.0, 1.0)

-   total_reward = rotation_reward + distance_reward
+   # Angular velocity penalty
+   angvel_norm = torch.norm(object_angvel, dim=1)
+   angvel_threshold = 0.5
+   angvel_penalty_temp = 5.0
+   angular_velocity_penalty = torch.where(angvel_norm > angvel_threshold,
+       torch.exp(-angvel_penalty_temp * (angvel_norm - angvel_threshold)), torch.zeros_like(angvel_norm))
+
+   total_reward = 0.5 * rotation_reward + 0.3 * distance_reward - 0.2 * angular_velocity_penalty

    reward_components = {
        "rotation_reward": rotation_reward,
        "distance_reward": distance_reward,
+       "angular_velocity_penalty": angular_velocity_penalty,
    }
```

Changing hyperparameter

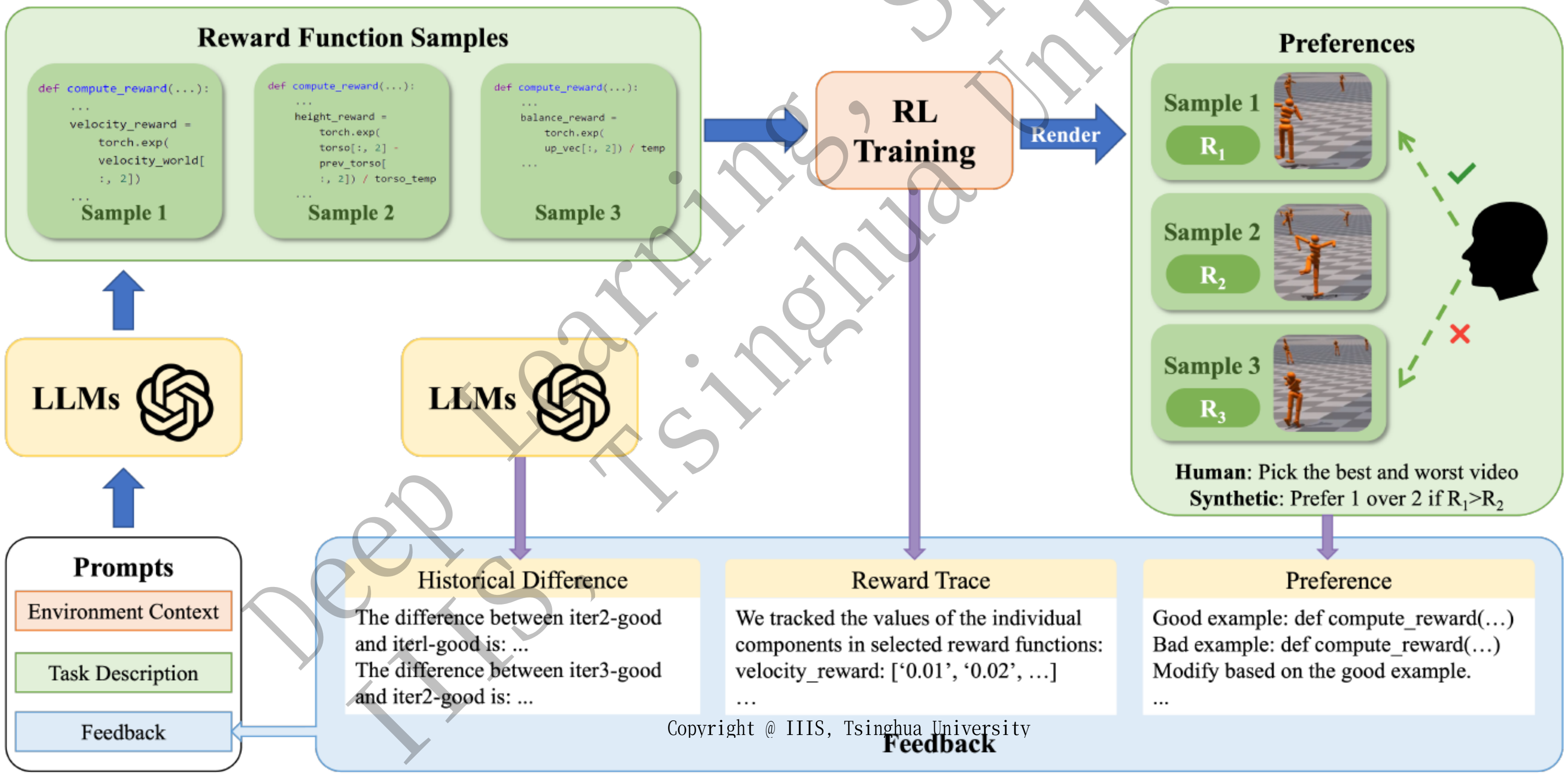
Changing functional form

Adding new component



Use LLM as a Pretrained Meta-Learner

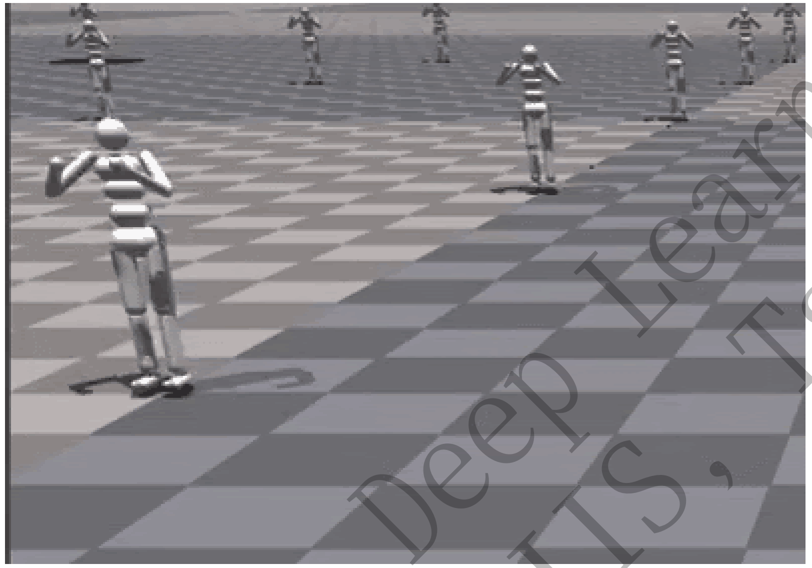
- ICPL: Few-shot In-context Preference Learning via LLMs (Yi Wu, 2025)
  - Human preference can be applied in this LLM evolving process



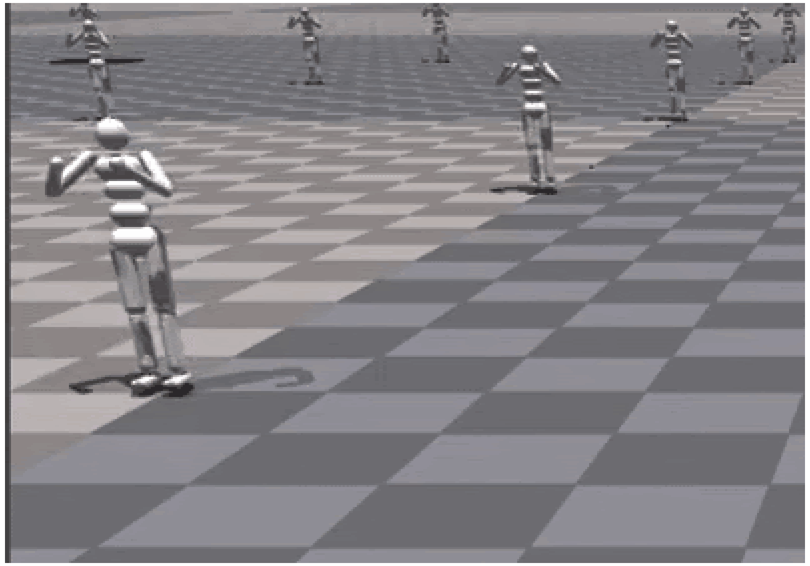
Use LLM as a Pretrained Meta-Learner

- ICPL: Few-shot In-context Preference Learning via LLMs (Yi Wu, 2025)
  - Human preference can be applied in this LLM evolving process

A humanoid robot jumps like a human



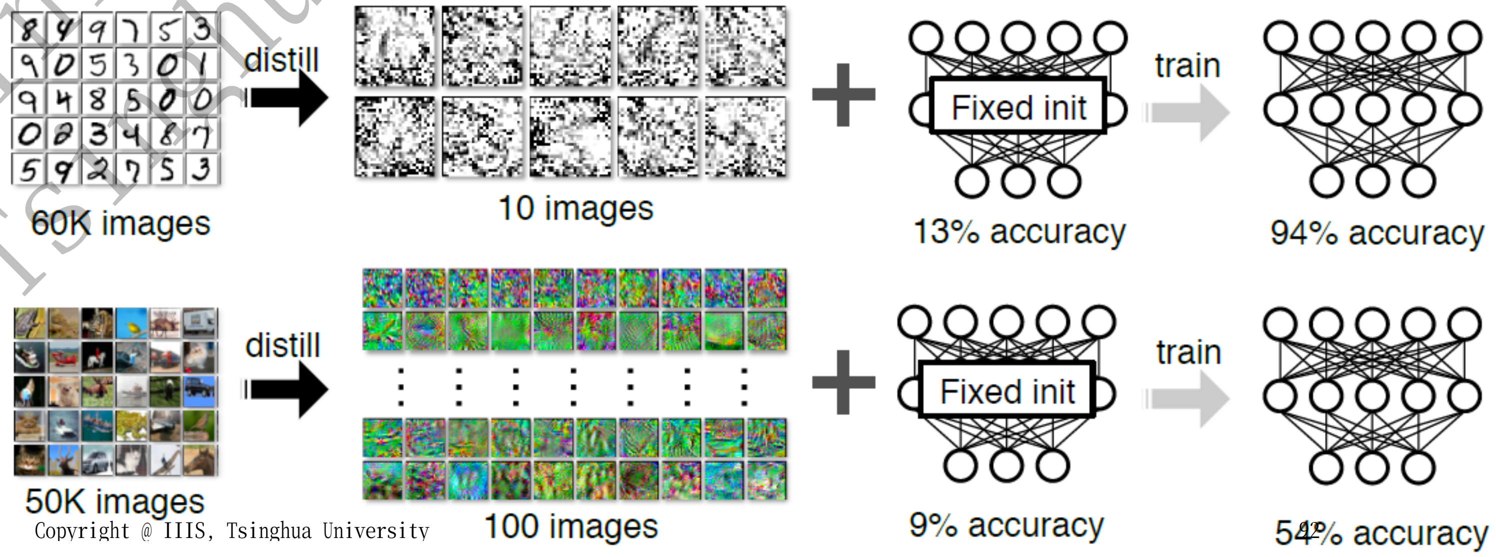
GPT-4 zero-shot result



ICPL with human in the loop

Learning to Learn

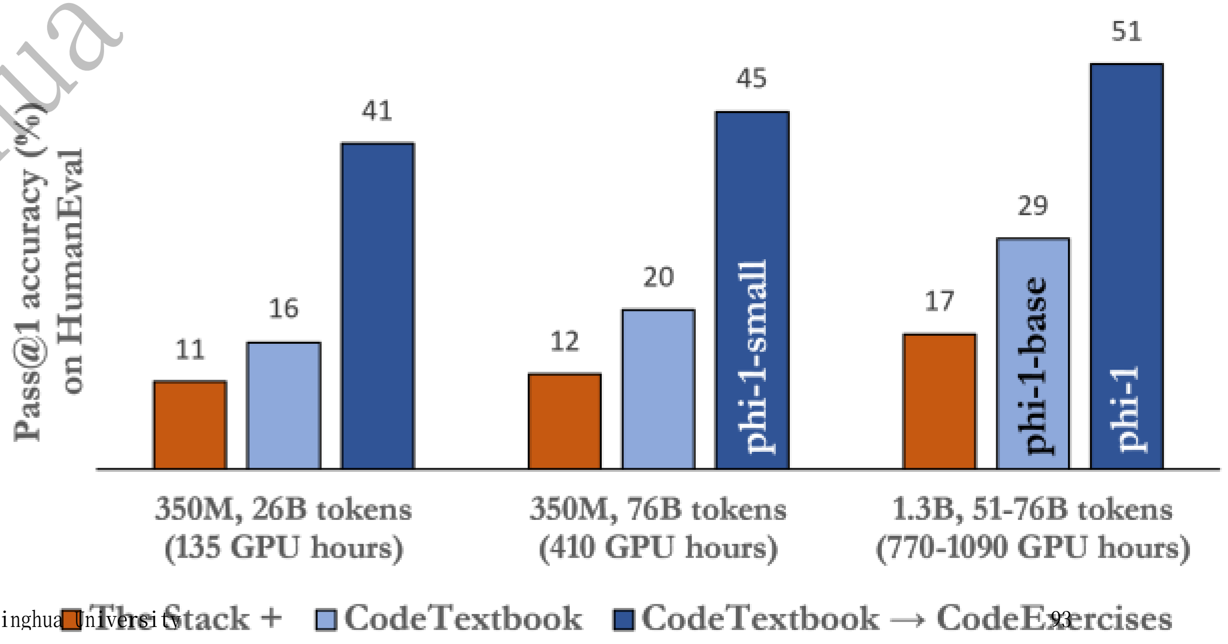
- Deep learning requires a large number of samples to train
  - Can we learn “dataset”?
    - Small-scale and efficient for learning
- Dataset Distillation (Wang et al, MIT & Berkeley, 2018)
  - Meta-learn training samples
  - Backprop via SGD process
  - Weight normalization



(a) Dataset distillation on MNIST and CIFAR10

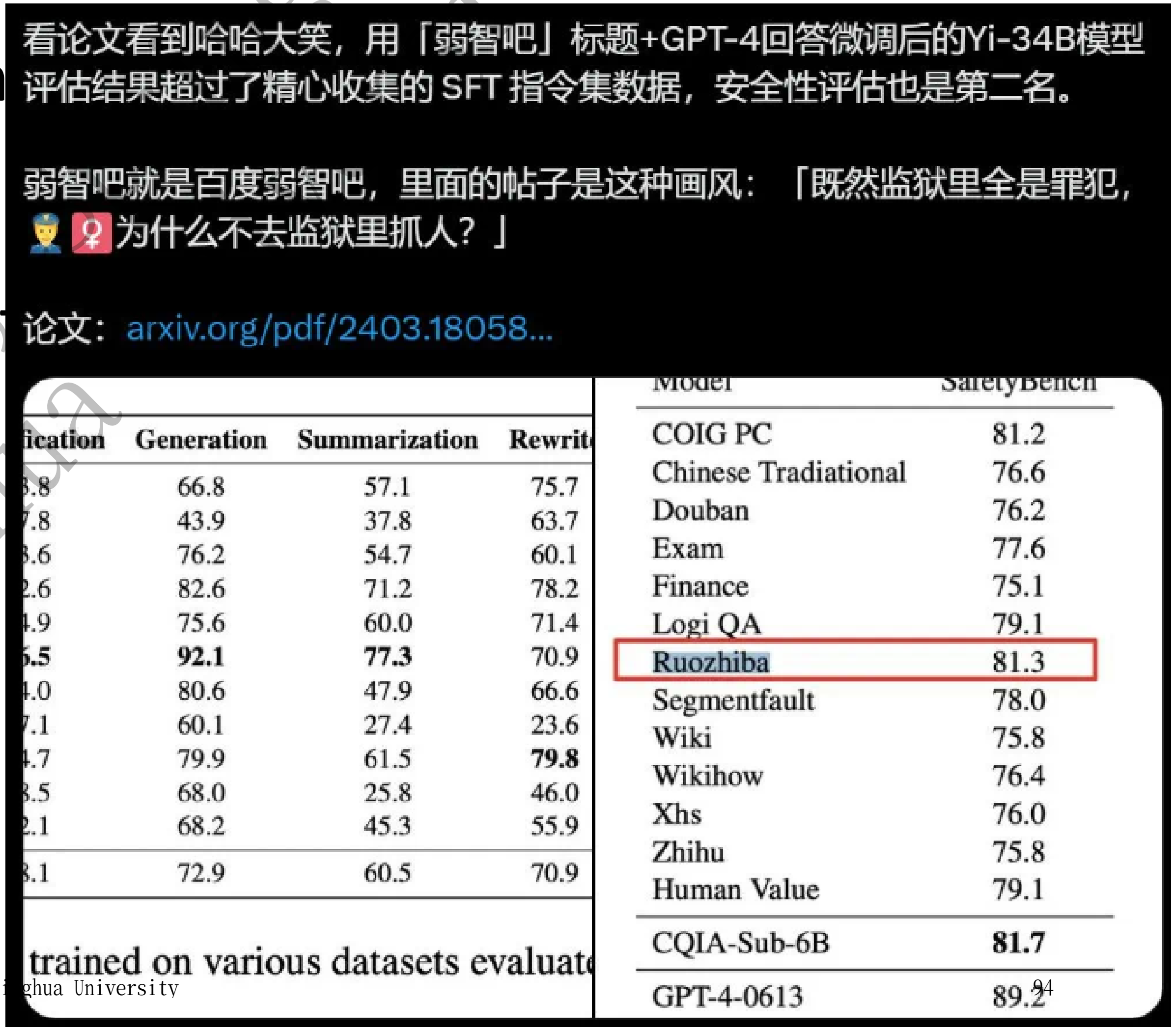
Learning to Learn

- Deep learning requires a large number of samples to train
  - Can we learn "dataset"?
    - Small-scale and efficient for learning
- Dataset Distillation (Wang et al, MIT & Berkeley, 2018)
- Data Quality for LLM training
  - Textbooks for LLM training
    - https://arxiv.org/abs/2306.11644



Learning to Learn

- Deep learning requires a large num
  - Can we learn “dataset”?
    - Small-scale and efficient for learning
- Dataset Distillation (Wang et al, MI
- Data Quality for LLM training
  - Textbooks for LLM training
    - https://arxiv.org/abs/2306.11644
  - RuoZhiBa for Chinese LLM training
    - Only 240 samples
    - https://arxiv.org/abs/2403.18058
    - https://huggingface.co/datasets/m-a-p/COIG-CQIA/viewer/ruozhiba


看论文看到哈哈大笑，用「弱智吧」标题+GPT-4回答微调后的Yi-34B模型评估结果超过了精心收集的 SFT 指令集数据，安全性评估也是第二名。

弱智吧就是百度弱智吧，里面的帖子是这种画风：「既然监狱里全是罪犯，👮♀️为什么不去监狱里抓人？」

论文：arxiv.org/pdf/2403.18058...

ication	Generation	Summarization	Rewrit
3.8	66.8	57.1	75.7
7.8	43.9	37.8	63.7
3.6	76.2	54.7	60.1
2.6	82.6	71.2	78.2
4.9	75.6	60.0	71.4
5.5	**92.1**	**77.3**	70.9
4.0	80.6	47.9	66.6
7.1	60.1	27.4	23.6
4.7	79.9	61.5	**79.8**
3.5	68.0	25.8	46.0
2.1	68.2	45.3	55.9
3.1	72.9	60.5	70.9

trained on various datasets evaluate

Model	SafetyBench
COIG PC	81.2
Chinese Traditional	76.6
Douban	76.2
Exam	77.6
Finance	75.1
Logi QA	79.1
Ruozhiba	81.3
Segmentfault	78.0
Wiki	75.8
Wikihow	76.4
Xhs	76.0
Zhihu	75.8
Human Value	79.1
CQIA-Sub-6B	**81.7**
GPT-4-0613	89.2

Learning to Learn

- Deep learning requires a large number of samples to train
  - Can we learn “dataset”?
    - Small-scale and efficient for learning
- Dataset Distillation (Wang et al, MIT &
- Data Quality for LLM training
  - Textbooks for LLM training
    - https://arxiv.org/abs/2306.11644
  - RuoZhiBa for Chinese LLM training
    - Only 240 samples
    - https://arxiv.org/abs/2403.18058
    - https://huggingface.co/datasets/m-a-p/COIG-CQIA/viewer/ruozhiba

21	咖啡豆是豆，咖啡算豆浆吗?
22	玉皇大帝住在平流层还是对流层?
23	为什么我爸妈结婚的时候没邀请我参加婚礼?
24	二郎神怎么做眼保健操?
25	变形金刚买保险是买车险还是人险?

Summary of Meta-Learning

- Few-Shot Learning
  - Goal: learn a network that can fast adapt
  - Metric-Learning
  - Bayesian Learning (probabilistic programming / symbolic learning)
  - Learning with gradient
    - MAML, TTT
- Learning to Learn
  - Learn an optimizer (and even an algorithm!)
    - Neural / symbolic update rule
  - LLM as pretrained meta-learner
  - Learn training instances
  - Even learn neural network architectures --- *Neural Architecture Search*!

Today’s Topic

- Unsupervised Learning and Self-supervised Learning
- Learning to Efficiently Learn Neural Networks
  - Aka. Meta-Learning, Learning to Learn
- Reinforcement Learning and Human-AI Collaboration
  - Some interesting projects from Prof. Wu’s group

Today’s Topic

- Unsupervised Learning and Self-supervised Learning
- Learning to Efficiently Learn Neural Networks
  - Aka. Meta-Learning, Learning to Learn
- **Reinforcement Learning and Human-AI Collaboration**
  - **Some interesting projects from Prof. Wu’s group**