school of ai
Port Harcourt

@Phcschoolofai

www.phcschoolofai.org

pmlcourse

Machine Learning in python

Recent Trends, Technologies, and Challenges

sebastianraschka.com

@rasbt

Sebastian Raschka, Ph.D.
Assist. Prof.
Dep. of Statistics
WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

information

Article

Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence

Sebastian Raschka [1,*,†], Joshua Patterson [2] and Corey Nolet [2,3]

1 Department of Statistics, University of Wisconsin-Madison, Madison, WI 53575, USA
2 NVIDIA, Santa Clara, CA 95051, USA; joshuap@nvidia.com (J.P.); cnolet@nvidia.com (C.N.)
3 Department of Comp Sci & Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD 21250, USA
* Correspondence: sraschka@wisc.edu
† Current address: 1300 University Ave, Medical Sciences Building, Madison, WI 53706, USA.

Received: 6 February 2020; Accepted: 31 March 2020; Published: 4 April 2020

See "Machine Learning with Python," a special issue of Information (ISSN 2078-2489)

https://www.mdpi.com/journal/information/special_issues/ML_Python

Part 1: Technologies and Tools


https://cse.engin.umich.edu/about/history/


https://careers.google.com/locations/pittsburgh/

Python for-loops are bad

$$z = \sum_i x_i w_i + b$$

```
def for_loop(w, x):
    z = 0.
    for i in range(len(x)):
        z += x[i] * w[i]
    return z


x = [1., 2., 3.]
bias = 0.1
w = [bias, 0.3, 0.5]

print(for_loop(w, x))
```

```
2.2
```

Python for-loops are bad:
Use SIMD & vectorized code whenever you can

$$z = \sum_i x_i w_i + b$$
$$= \mathbf{x}^\top \mathbf{w}$$

```
import torch


def dot_product_in_pytorch(w, x):
    return x.dot(w)

x_t, w_t = torch.tensor(x), torch.tensor(w)
print(dot_product_in_pytorch(x_t, w_t))
```

```
tensor(2.2000)
```

For-loops vs. Vectorized Code

```
%timeit -r 100 -n 1 -q -o for_loop(w, x)
```

```
<TimeitResult : 33.8 ms ± 1.43 ms per loop (mean ± std. dev.
```

```
%timeit -r 100 -n 1 -q -o  dot_product_in_pytorch(w_t, x_t)
```

```
<TimeitResult : 22.4 µs ± 25.1 µs per loop (mean ± std. dev.
```

Dot product is approx.1500x faster

Can we speed this up further using GPUs?

```
%timeit -r 100 -n 1 -q -o  dot_product_in_pytorch(w_t, x_t)
```

```
<TimeitResult : 22.4 µs ± 25.1 µs per loop (mean ± std. dev. of 100 runs,
```

```
x_cuda = x_t.to(torch.device('cuda:0'))
w_cuda = w_t.to(torch.device('cuda:0'))
```

```
torch.backends.cudnn.benchmark = True
```

```
%timeit -r 100 -n 1 -q -o  dot_product_in_pytorch(w_cuda, x_cuda)
```

```
<TimeitResult : 61 µs ± 13.8 µs per loop (mean ± std. dev. of 100 runs, 1
```

GPU is approx. 3x slower

Can we speed this up further using GPUs?
(Yes, if the data / computation is large)

```
X, W = torch.rand((1000, 10000)), torch.rand((10000, 100))
```

```
%timeit -r 100 -n 1 -q -o X.mm(W)
```

```
<TimeitResult : 4.38 ms ± 1.59 ms per loop (mean ± std. dev.
```

```
X_cuda = X.to(torch.device('cuda:0'))
W_cuda = W.to(torch.device('cuda:0'))
```

```
%timeit -r 100 -n 1 -q -o X_cuda.mm(W_cuda)
```

```
<TimeitResult : 18.6 µs ± 26.5 µs per loop (mean ± std. dev.
```

GPU is approx. 230x faster

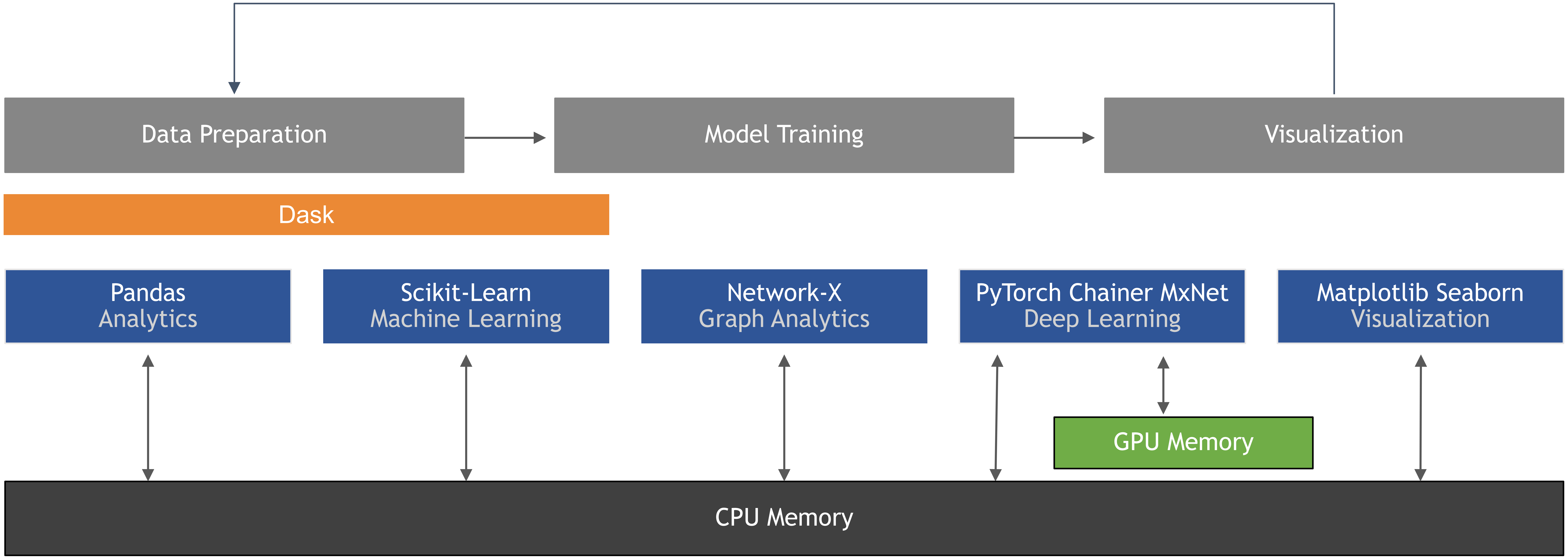


Figure 1. The standard Python ecosystem for machine learning, data science, and scientific computing.

Sebastian Raschka, Joshua Patterson, and Corey Nolet (2020)
Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence
Information 2020, 11, 193

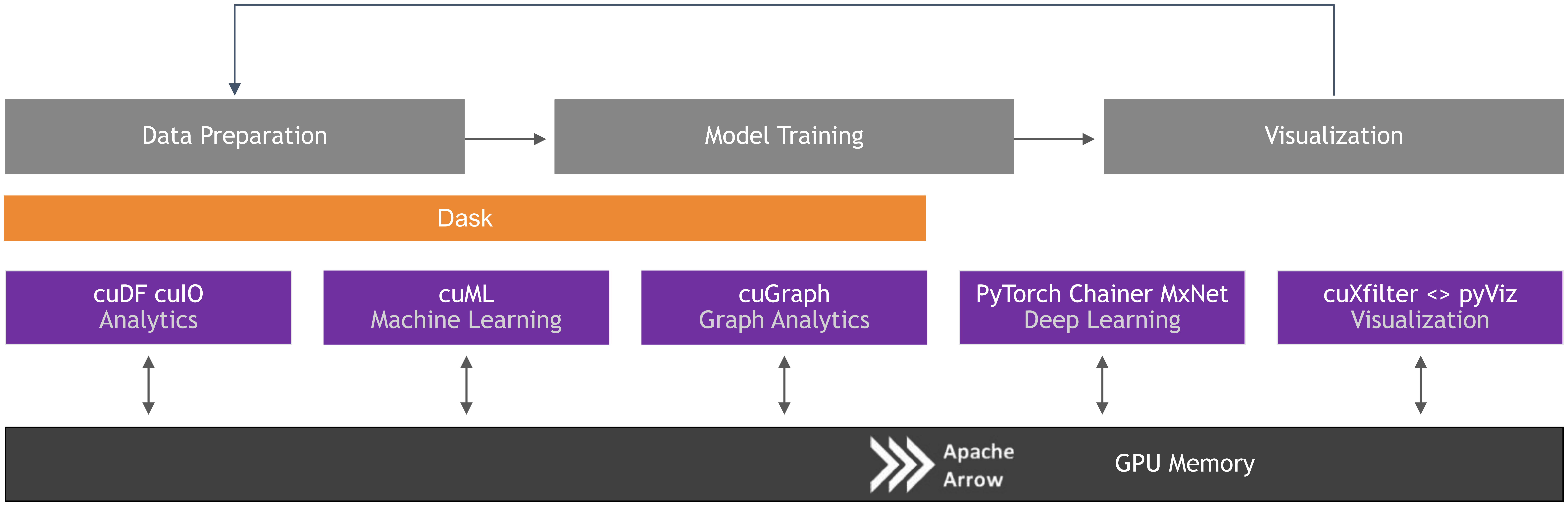


Figure 4. RAPIDS is an open source effort to support and grow the ecosystem of GPU-accelerated Python tools for data science, machine learning, and scientific computing. RAPIDS supports existing libraries, fills gaps by providing open source libraries with crucial components that are missing from the Python community, and promotes cohesion across the ecosystem by supporting interoperability across the libraries.

Sebastian Raschka, Joshua Patterson, and Corey Nolet (2020)
Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence
Information 2020, 11, 193

GPUs

https://www.theverge.com/2015/5/31/8695075/nvidia-geforce-gtx-980-ti-announced

Specifications	Intel® Core™ i7-5960X Processor Extreme Edition	NVIDIA GeForce® GTX™ 980 Ti
Base Clock Frequency	3.0 GHz	1.0 GHz
Cores	8	2816
Memory Bandwidth	68 GB/s	336.5 GB/s
Floating-Point Calculations	354 GFLOPS	5632 GFLOPS
Cost	$1000.00	$700.00

Sebastian Raschka. *Python Machine Learning.* Birmingham, UK: Packt Publishing, **2015**

Image: https://www.amazon.com/Nvidia-GEFORCE-GTX-1080-Ti/dp/B06XH5ZCLP

Specifications	Intel® Core™ i7-6900K Processor Extreme Ed.	NVIDIA GeForce® GTX™ 1080 Ti
Base Clock Frequency	3.2 GHz	< 1.5 GHz
Cores	8	3584
Memory Bandwidth	64 GB/s	484 GB/s
Floating-Point Calculations	409 GFLOPS	11300 GFLOPS
Cost	~ $1000.00	~ $700.00

Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning 2nd Ed.* Birmingham, UK: Packt Publishing, **2017**

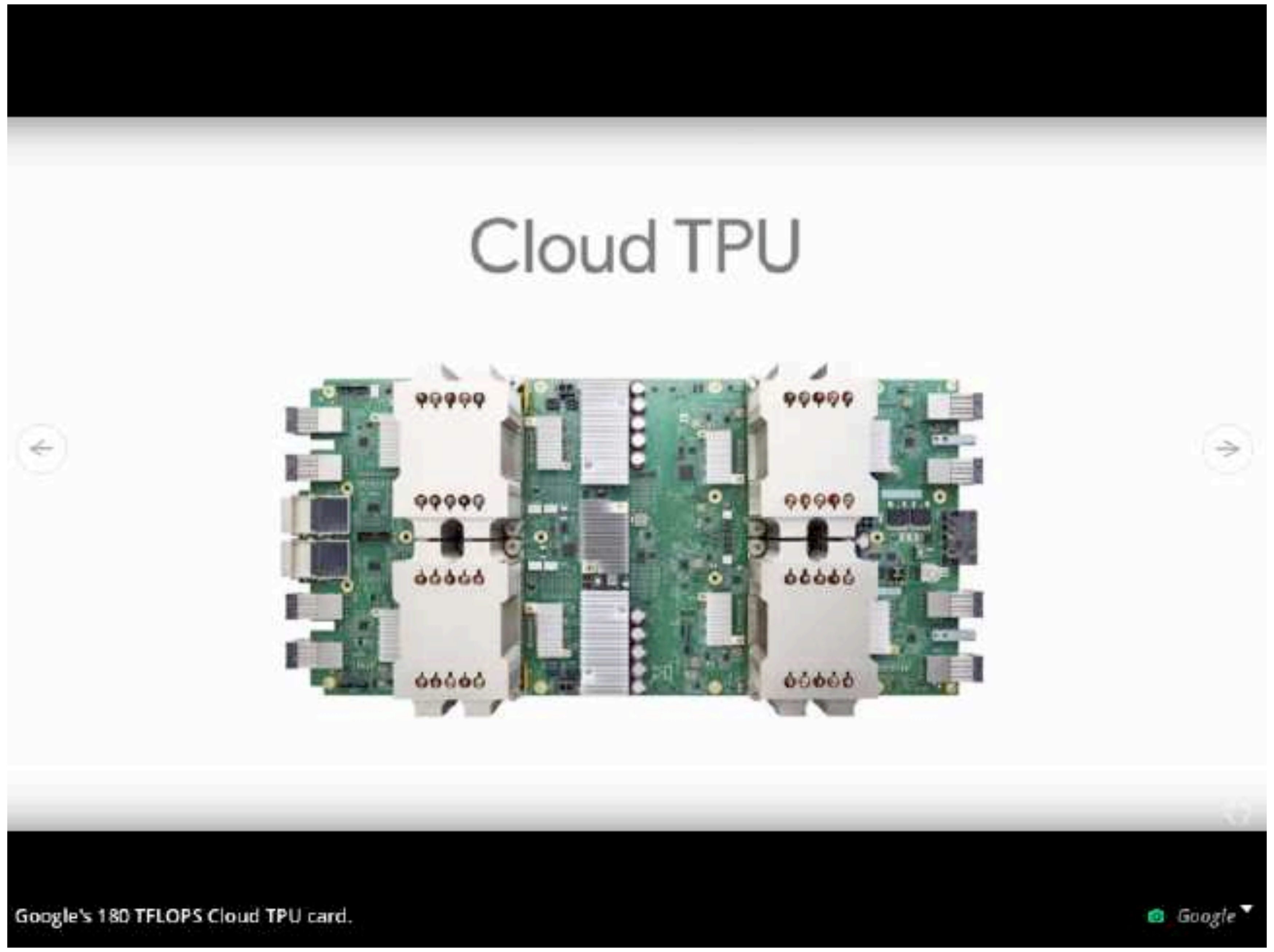
Image: https://www.nvidia.com/en-in/geforce/graphics-cards/rtx-2080-ti/

Specifications	Intel® Core™ i9-9960X X-series Processor	NVIDIA GeForce® RTX™ 2080 Ti
Base Clock Frequency	3.1 GHz	1.35 GHz
Cores	16 (32 threads)	4352
Memory Bandwidth	79.47 GB/s	616 GB/s
Floating-Point Calculations	1290 GFLOPS	13400 GFLOPS
Cost	~ $1700.00	~ $1100.00

Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning 3rd Ed.* Birmingham, UK: Packt Publishing, **2019**

Source: https://lambdalabs.com/blog/2080-ti-deep-learning-benchmarks/

Developing Specialized Hardware



https://arstechnica.com/gadgets/2018/07/the-ai-revolution-has-spawned-a-new-chips-arms-race/

Opinion: New Nvidia chip extends the company's lead in graphics, artificial intelligence

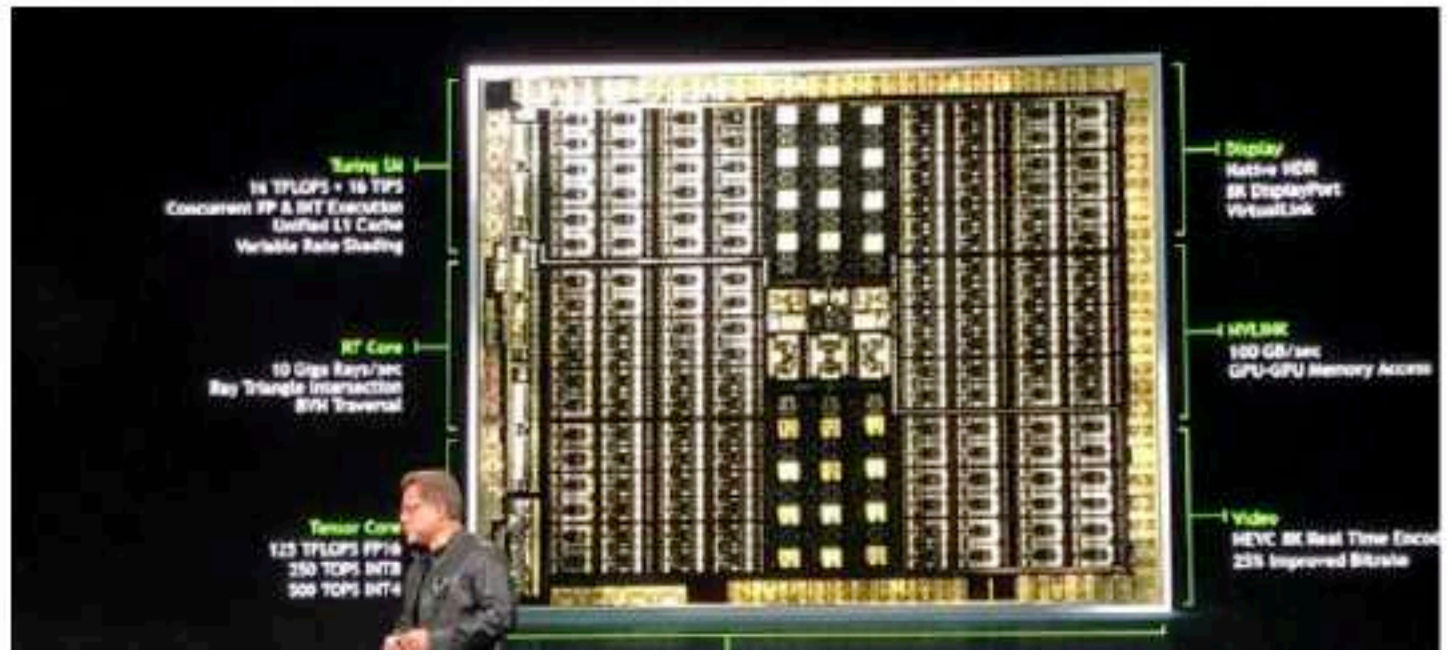
By Ryan Shrout

Published: Aug 14, 2018 2:35 p.m. ET

1

Aa

The only question that remains: How big is Nvidia's advantage over its rivals?



https://www.marketwatch.com/story/new-nvidia-chip-extends-the-companys-lead-in-graphics-artificial-intelligence-2018-08-14

https://developer.arm.com/products/processors/machine-learning/arm-ml-processor

TECHNOLOGY NEWS NOVEMBER 28, 2018 / 2:59 PM / 2 MONTHS AGO

Amazon launches machine learning chip, taking on Nvidia, Intel

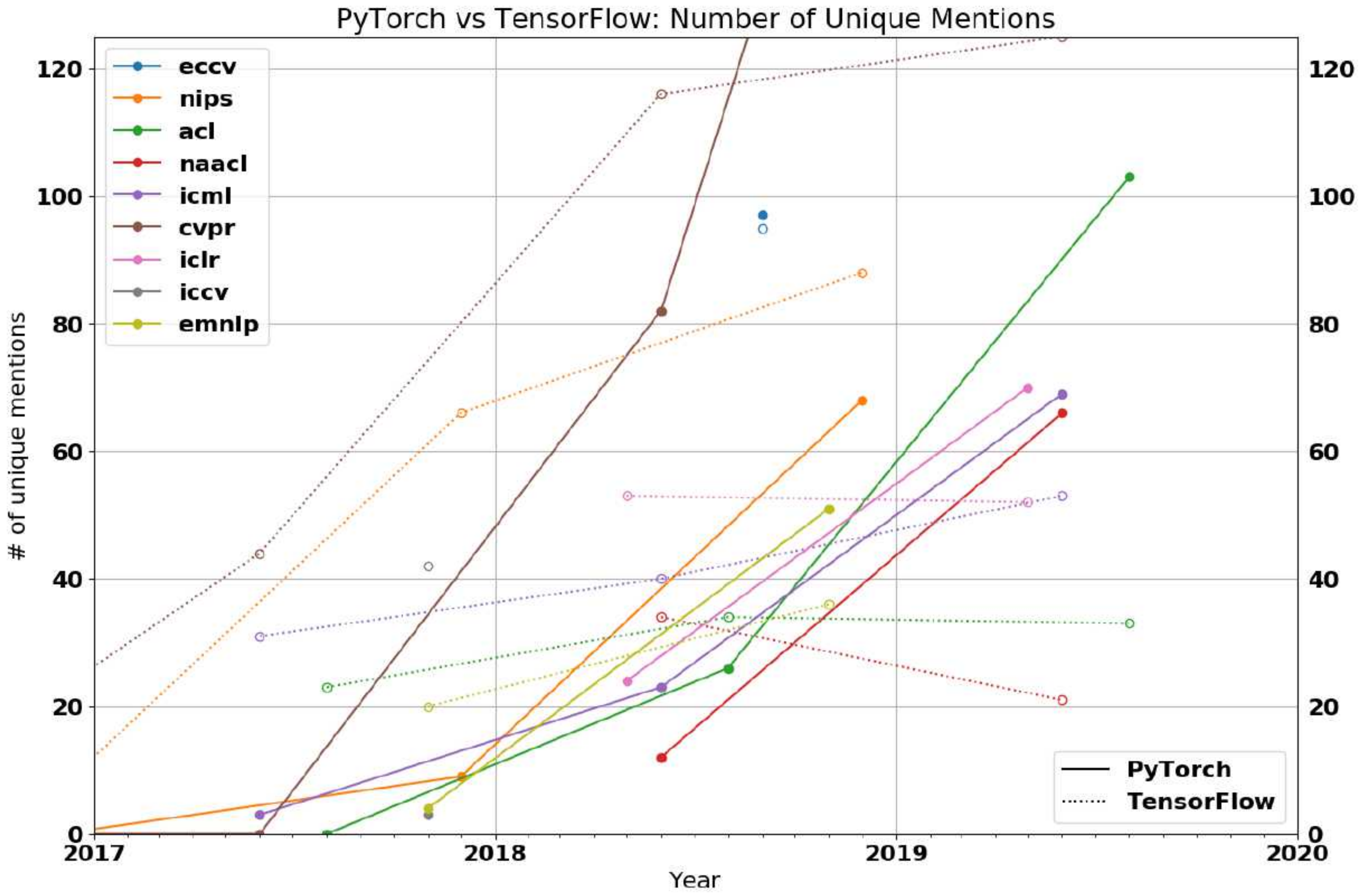
https://www.reuters.com/article/us-amazon-com-nvidia/amazon-launches-machine-learning-chip-taking-on-nvidia-intel-idUSKCN1NX2PY

Deep Learning frameworks

The new "Emacs vs VIM"

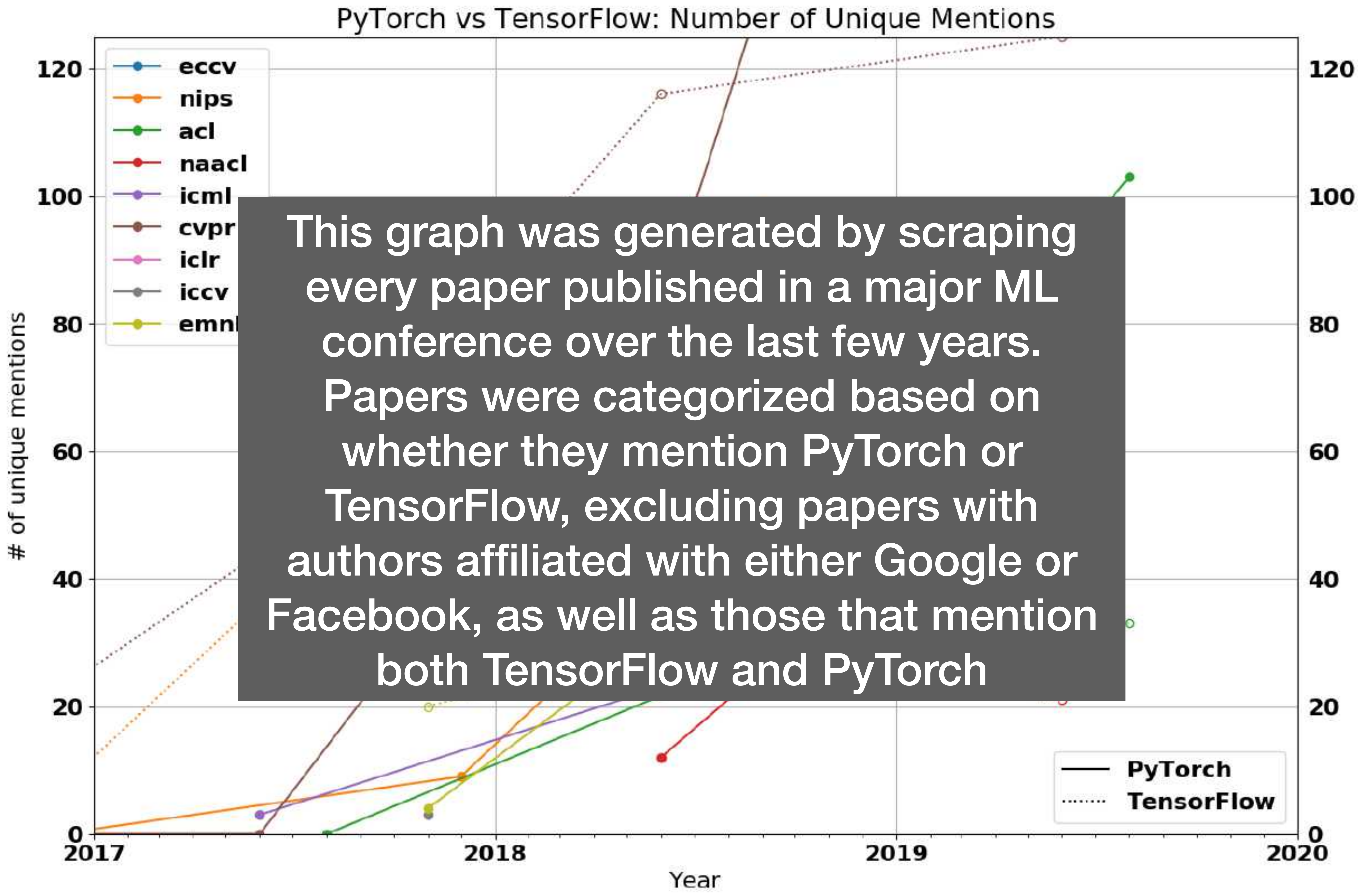
Which DL framework is most popular?

https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/



The Gradient

https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/



This graph was generated by scraping every paper published in a major ML conference over the last few years. Papers were categorized based on whether they mention PyTorch or TensorFlow, excluding papers with authors affiliated with either Google or Facebook, as well as those that mention both TensorFlow and PyTorch

"Most I've spoken to (and I'm from a background in ML academia); PyTorch is by a very slim margin faster than TensorFlow 2.0 in our experiences when you run TensorFlow in non-Eager mode. However, since Eager mode is now enabled by default in TensorFlow 2.0; PyTorch is significantly faster."

https://www.reddit.com/r/MachineLearning/comments/f19dj4/d_tensorflow_20_v_pytorch_performance_question/

Average inference time	
PyTorch CPU Average inference time (s)	0.748
PyTorch CPU + TorchScript Average inference time (s)	0.625
PyTorch GPU Average inference time (s)	0.046
PyTorch GPU + TorchScript Average inference time (s)	0.036
TensorFlow CPU Average inference time (s)	0.823
TensorFlow GPU Average inference time (s)	0.043
TensorFlow GPU + XLA Average inference time (s)	0.035

Average inference time

Source: https://medium.com/huggingface/benchmarking-transformers-pytorch-and-tensorflow-e2917fb891c2

a

In:
```python
import tensorflow as tf
```

Defining the graph

```python
g = tf.Graph()
with g.as_default():
    x = tf.placeholder(dtype=tf.float32,
                       shape=(None), name='inputs')
    w = tf.Variable(2.0, name='weight')
    b = tf.Variable(1.5, name='bias')
    z = w*x + b
    dz_w = tf.gradients(z, w)

    init = tf.global_variables_initializer()
```

Initializing and evaluating the graph

```python
with tf.Session(graph=g) as sess:
    sess.run(init)
    result = sess.run(z, feed_dict={'inputs:0':1.0})
    partial_d = sess.run(dz_w, feed_dict={'inputs:0':1.0})
    print(f'w*x + b = {result}')
    print(f'∂z/∂w = {partial_d}')
```

Out:
```
w*x + b = 3.5
∂z/∂w = [1.0]
```

b

In:
```python
import torch

w = torch.tensor(2.0, requires_grad=True)
b = torch.tensor(1.5)
x = torch.tensor(1.0)

z = w*x + b
print(f'w*x + b = {z}')

z.backward()
print(f'∂z/∂w = {w.grad}')
```

Out:
```
w*x + b = 3.5
∂z/∂w = 1.0
```

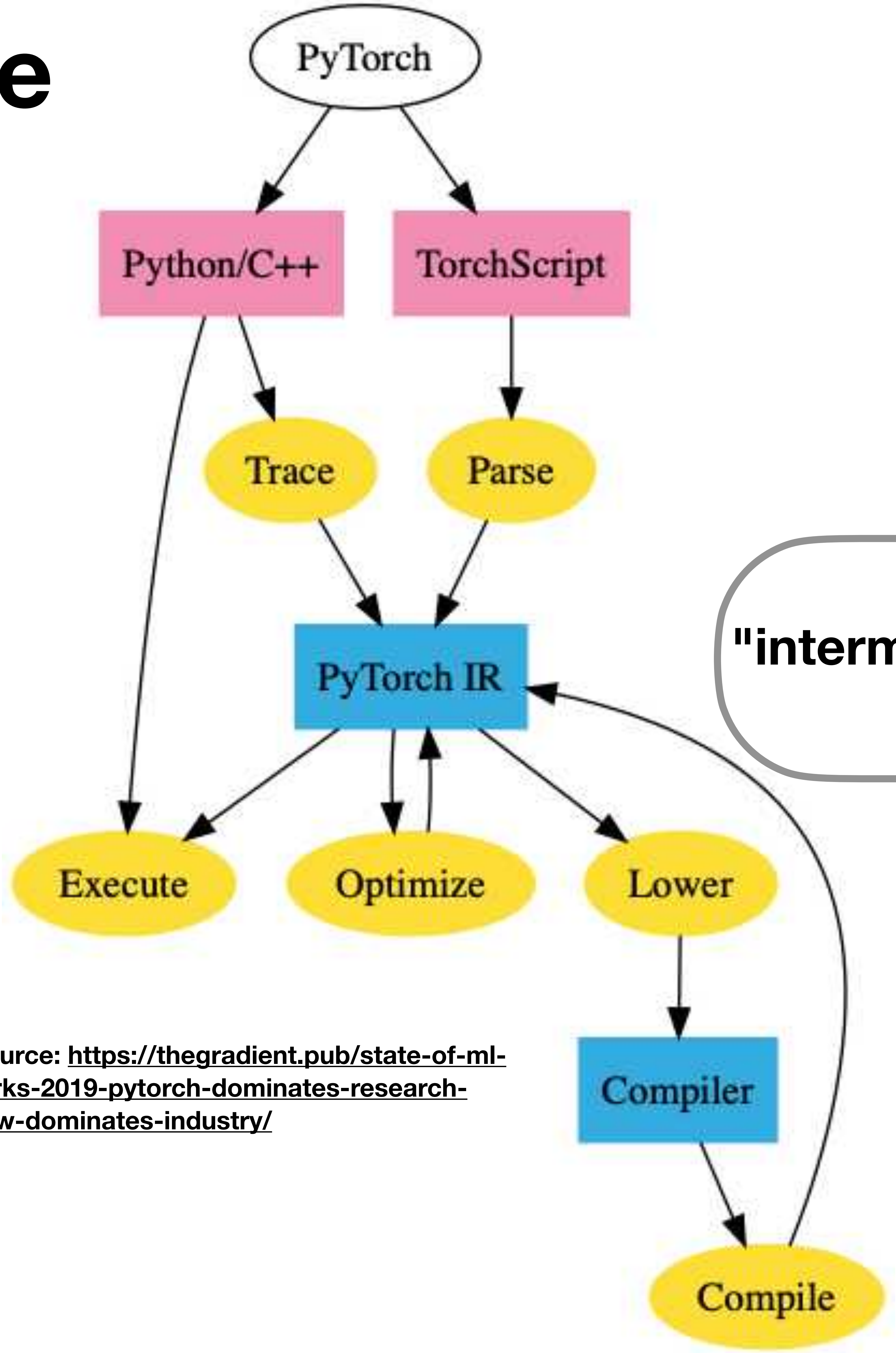
Figure 7. Comparison between (**a**) a static computation graph in TensorFlow 1.15 and (**b**) an imperative programming paradigm enabled by dynamic graphs in PyTorch 1.4.

Sebastian Raschka, Joshua Patterson, and Corey Nolet (2020)
Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence
Information 2020, 11, 193

DL Frameworks are converging

Two ways for turning a PyTorch model into a static graph for optimization and deployment:

a) Tracing
b) Scripting

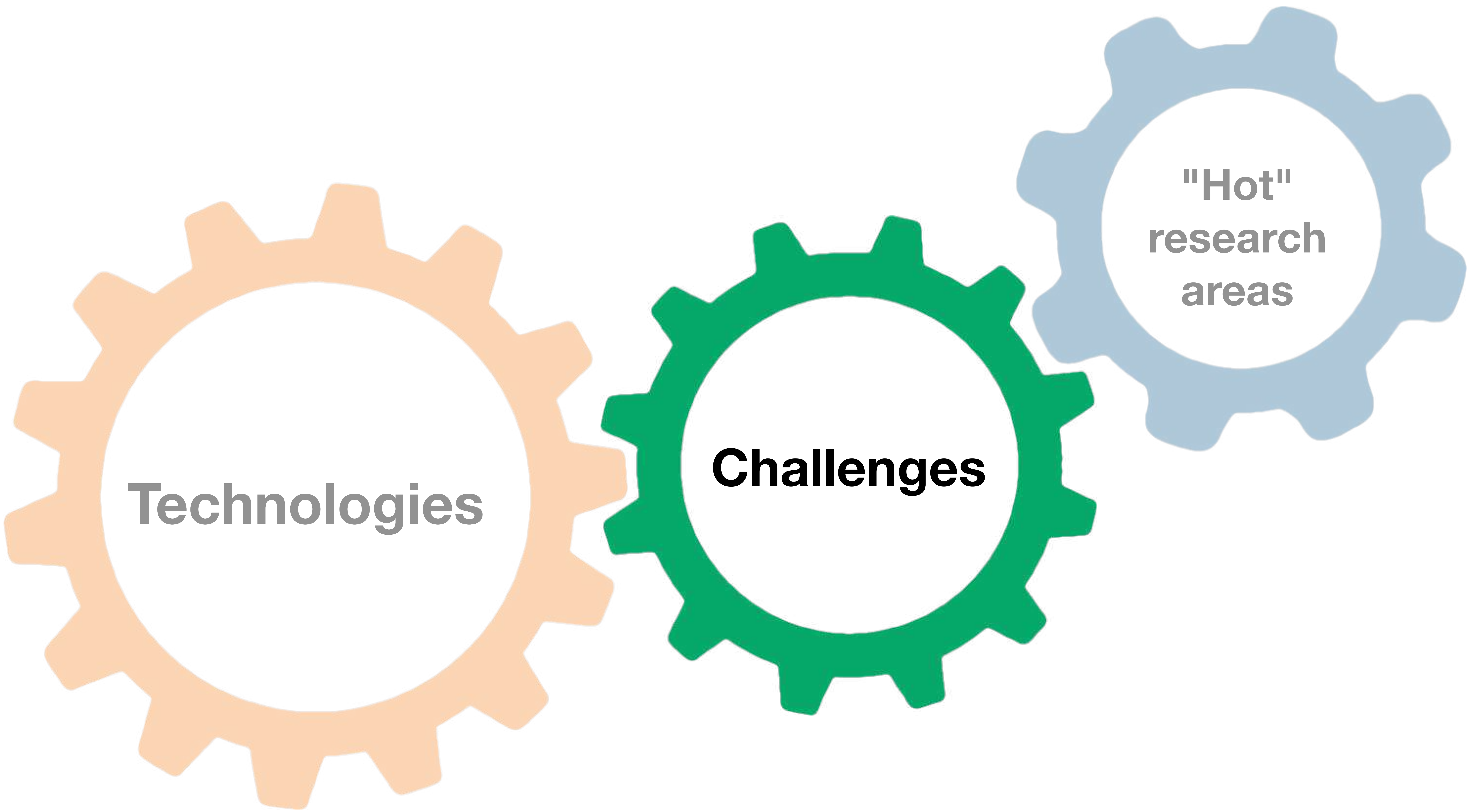


"intermediate" representation (IR) <=> "static graph"

Image Source: https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/

DL Frameworks are converging

Eager Execution


Technologies
Challenges
"Hot" research areas

Challenges:

Adding forward-mode autodiff for efficient higher-order derivatives (e.g., Hessians)

JAX

Composable transformations of Python+NumPy programs: differentiate, vectorize, JIT to GPU/TPU, and more

Forward mode AD for PyTorch: https://github.com/pytorch/pytorch/issues/10223

Swift

April 21, 2020

PyTorch 1.5 released, new and updated APIs including C++ frontend API parity with Python

https://pytorch.org/blog/pytorch-1-dot-5-released-with-new-and-updated-apis/

torch.autograd.functional.hessian(...)

Challenges:

Adversarial attacks

Phantom of the **ADAS**: Phantom Attacks on Driver-Assistance Systems

Ben Nassi[1], Dudi Nassi[1], Raz Ben-Netanel[1], Yisroel Mirsky[1,2], Oleg Drokin[3], Yuval Elovici[1]

Video Demonstration - https://youtu.be/1cSw4fXYqWI

{nassib,nassid,razx,yisroel,elovici}@post.bgu.ac.il, green@linuxhacker.ru

[1] Ben-Gurion University of the Negev, [2] Georgia Tech,[3] Independent Tesla Researcher

ABSTRACT

The absence of deployed vehicular communication systems, which prevents the advanced driving assistance systems (ADASs) and autopilots of semi/fully autonomous cars to validate their virtual perception regarding the physical environment surrounding the car with a third party, has been exploited in various attacks suggested by researchers. Since the application of these attacks comes with a cost (exposure of the attacker's identity), the delicate exposure vs. application balance has held, and attacks of this kind have not yet been encountered in the wild. In this paper, we investigate a new perceptual challenge that causes the ADASs and autopilots of semi/fully autonomous to consider depthless objects (phantoms) as real. We show how attackers can exploit this perceptual challenge to apply phantom attacks and change the abovementioned balance, without the need to physically

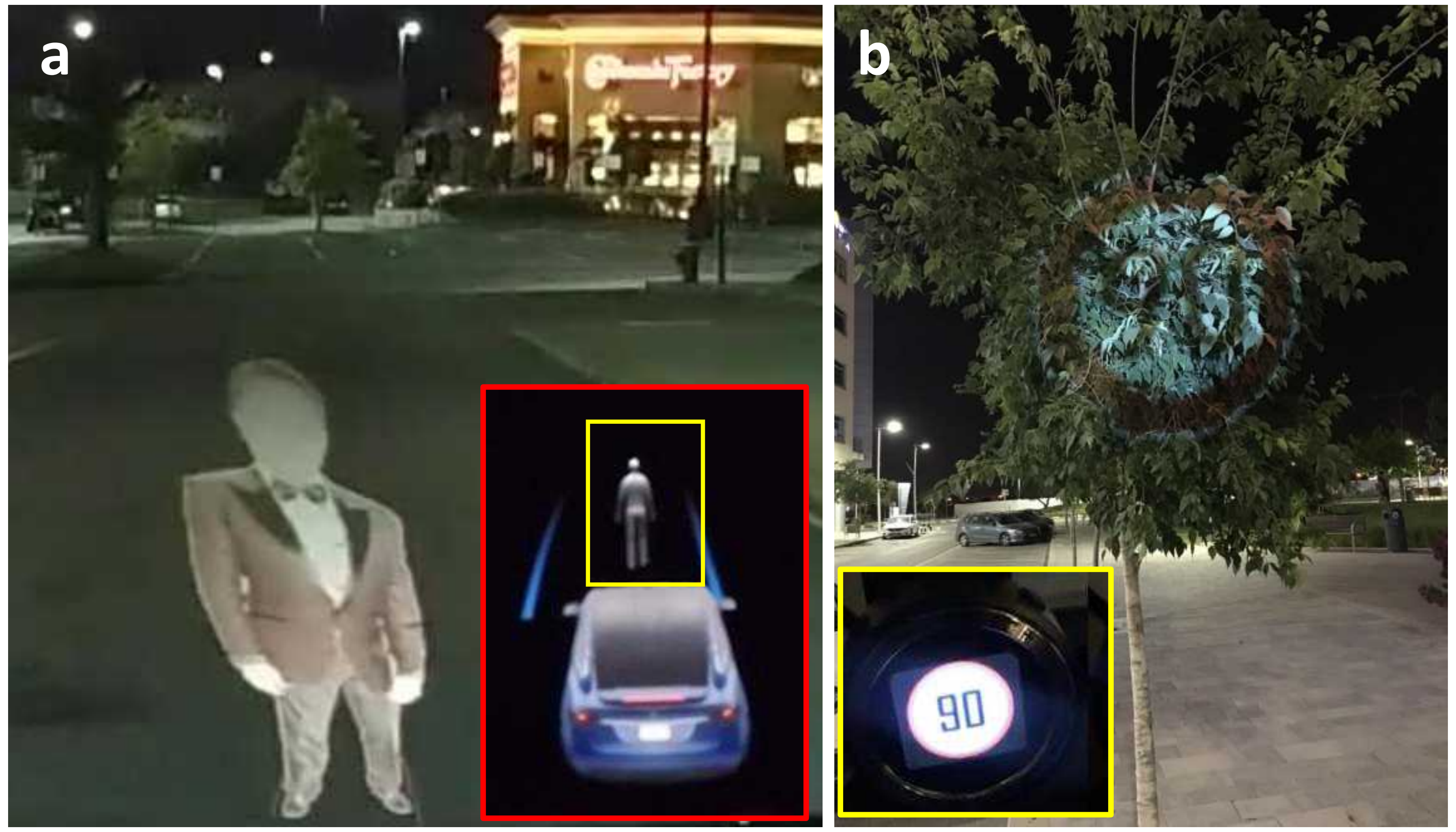


Fig. 1: Perceptual Challenge: Would you consider the projection of the person (a) and road sign (b) real? Telsa considers (a) a real person and Mobileye 630 PRO considers (b) a real road sign.

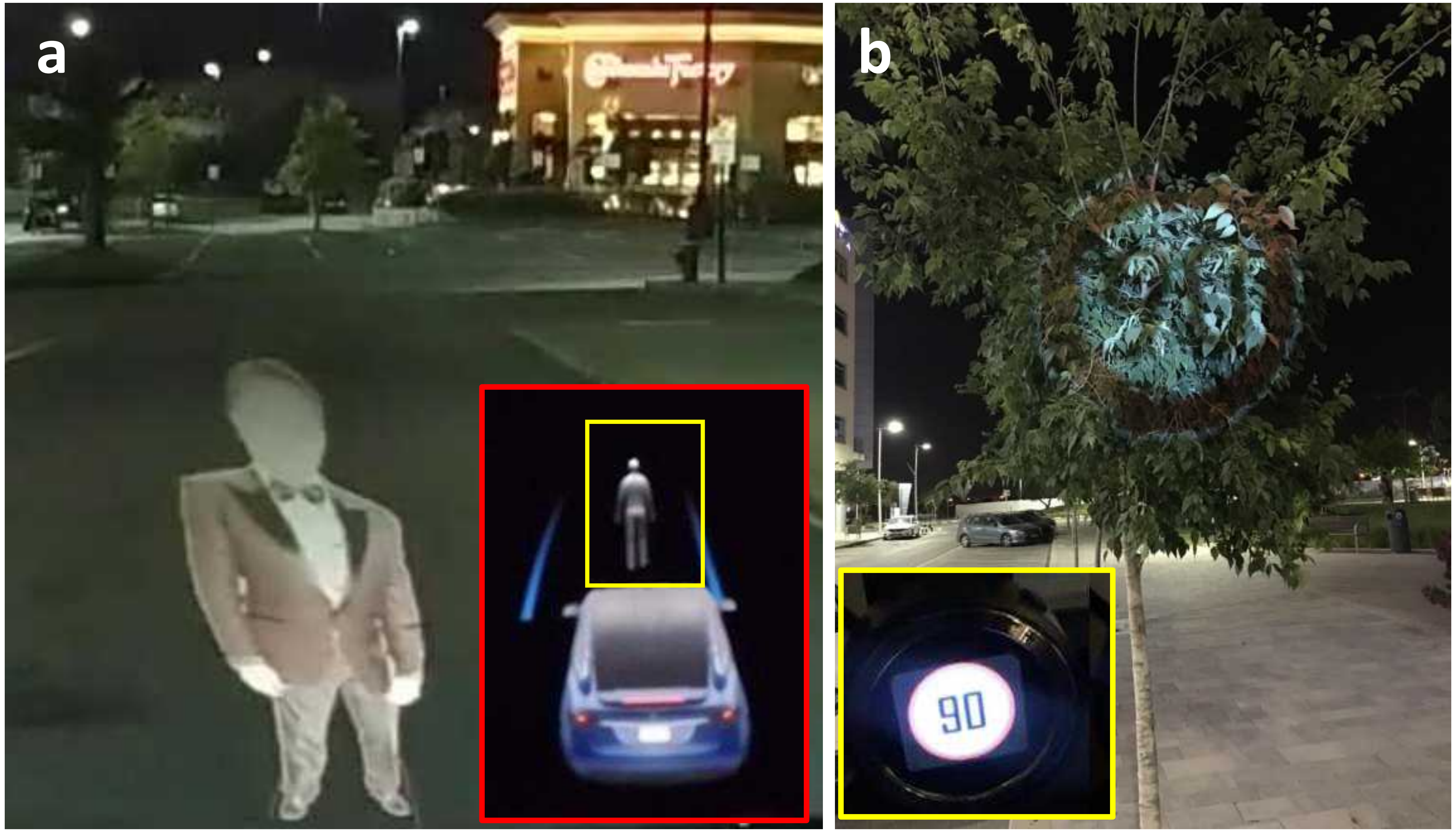


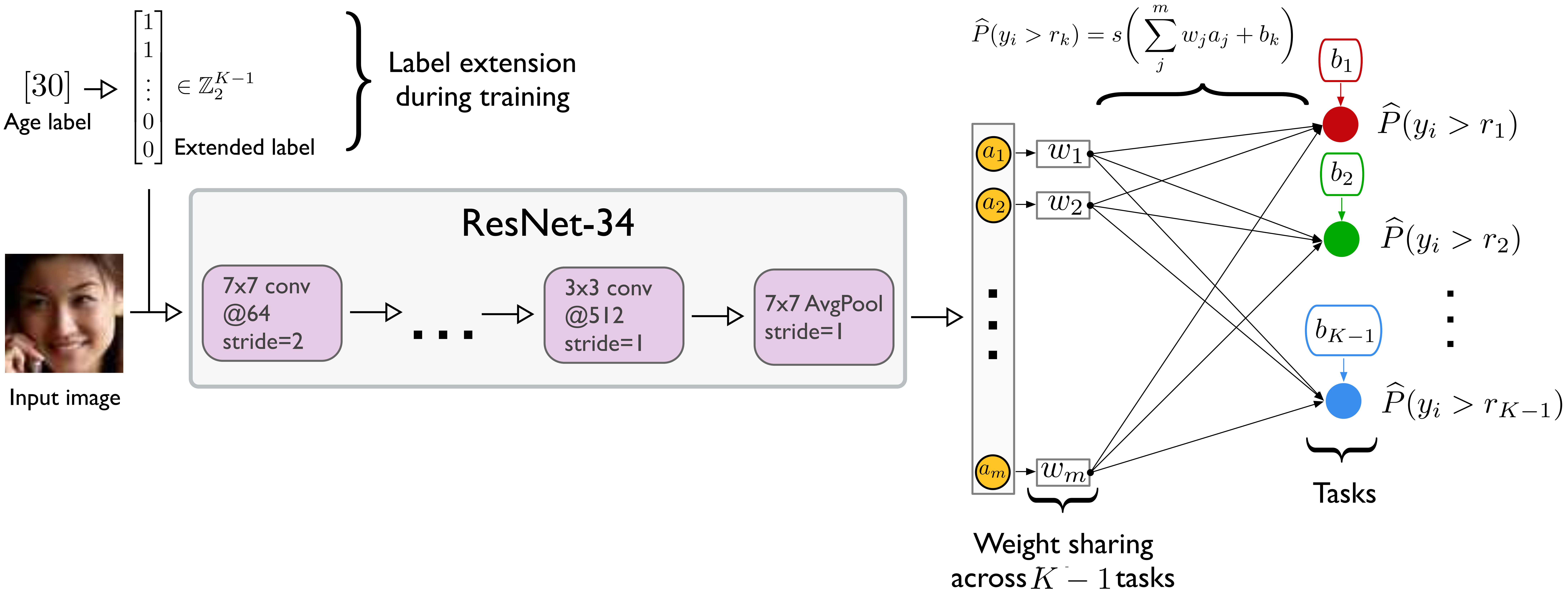
Fig. 1: Perceptual Challenge: Would you consider the projection of the person (a) and road sign (b) real? Telsa considers (a) a real person and Mobileye 630 PRO considers (b) a real road sign.

	Cleverhans v3.0.1	FoolBox v2.3.0	ART v1.1.0	DEEPSEC (2019)	AdvBox v0.4.1
Supported frameworks					
TensorFlow	yes	yes	yes	no	yes
MXNet	yes	yes	yes	no	yes
PyTorch	no	yes	yes	yes	yes
PaddlePaddle	no	no	no	no	yes
(Evasion) attack mechanisms					
BLB [163]	yes	no	no	yes	no
AMD [170]	yes	no	no	no	no
ZOO [171]	no	no	yes	no	no
VA [172]	yes	yes	yes	no	no
AP [173]	no	no	yes	no	no
STA [174]	no	yes	yes	no	no
DTA [175]	no	no	yes	no	no
FGSM [176]	yes	yes	yes	yes	yes
R+FGSM [177]	no	no	no	yes	no
R+LLC [177]	no	no	no	yes	no
U-MI-FGSM [178]	yes	yes	no	yes	no
T-MI-FGSM [178]	yes	yes	no	yes	no
BIM [179]	no	yes	yes	yes	yes
LLC / ILLC [179]	no	yes	no	yes	no
UAP [180]	no	no	yes	yes	no
DeepFool [181]	yes	yes	yes	yes	yes
NewtonFool [182]	no	yes	yes	no	no
JSMA [183]	yes	yes	yes	yes	yes
CW/CW2 [184]	yes	yes	yes	yes	yes
PGD [185]	yes	no	yes	yes	yes
OM [186]	no	no	no	yes	no
EAD [187]	yes	yes	yes	yes	no
Boundary Attack [188]	no	yes	yes	no	no
HopSkipJumpAttack [189]	yes	yes	yes	no	no
MaxConf [190]	yes	no	no	no	no
Inversion attack [191]	yes	yes	no	no	no
SparseL1 [192]	yes	yes	no	no	no
SPSA [193]	yes	no	no	no	no
HCLU [194]	no	no	yes	no	no
ADef [195]	no	yes	no	no	no
DDNL2 [196]	no	yes	no	no	no
Local Search [197]	no	yes	no	no	no
Pointwise attack [198]	no	yes	no	no	no
GenAttack [199]	no	yes	no	no	no
Defense mechanisms					
Feature Squeezing [200]	no	no	yes	no	yes
Spatial Smoothing [200]	no	no	yes	no	yes
Label Smoothing [200]	no	no	yes	no	yes
Gaussian Augmentation [201]	no	no	yes	no	yes
Adversarial Training [185]	no	no	yes	yes	yes
Thermometer Encoding [202]	no	no	yes	yes	yes
NAT [203]	no	no	no	yes	no
EAT [177]	no	no	no	yes	no
DD [204]	no	no	no	yes	no
IGR [205]	no	no	no	yes	no
EIT [206]	no	no	yes	yes	no
RT [207]	no	no	no	yes	no
PixelDefend [208]	no	no	yes	yes	no
Regr.-based classfication [209]	no	no	no	yes	no
JPEG compression [210]	no	no	yes	no	no

Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence (2020). Sebastian Raschka, Joshua Patterson, and Corey Nolet

Neglected Research Areas

Deep Learning & Ordinal Data



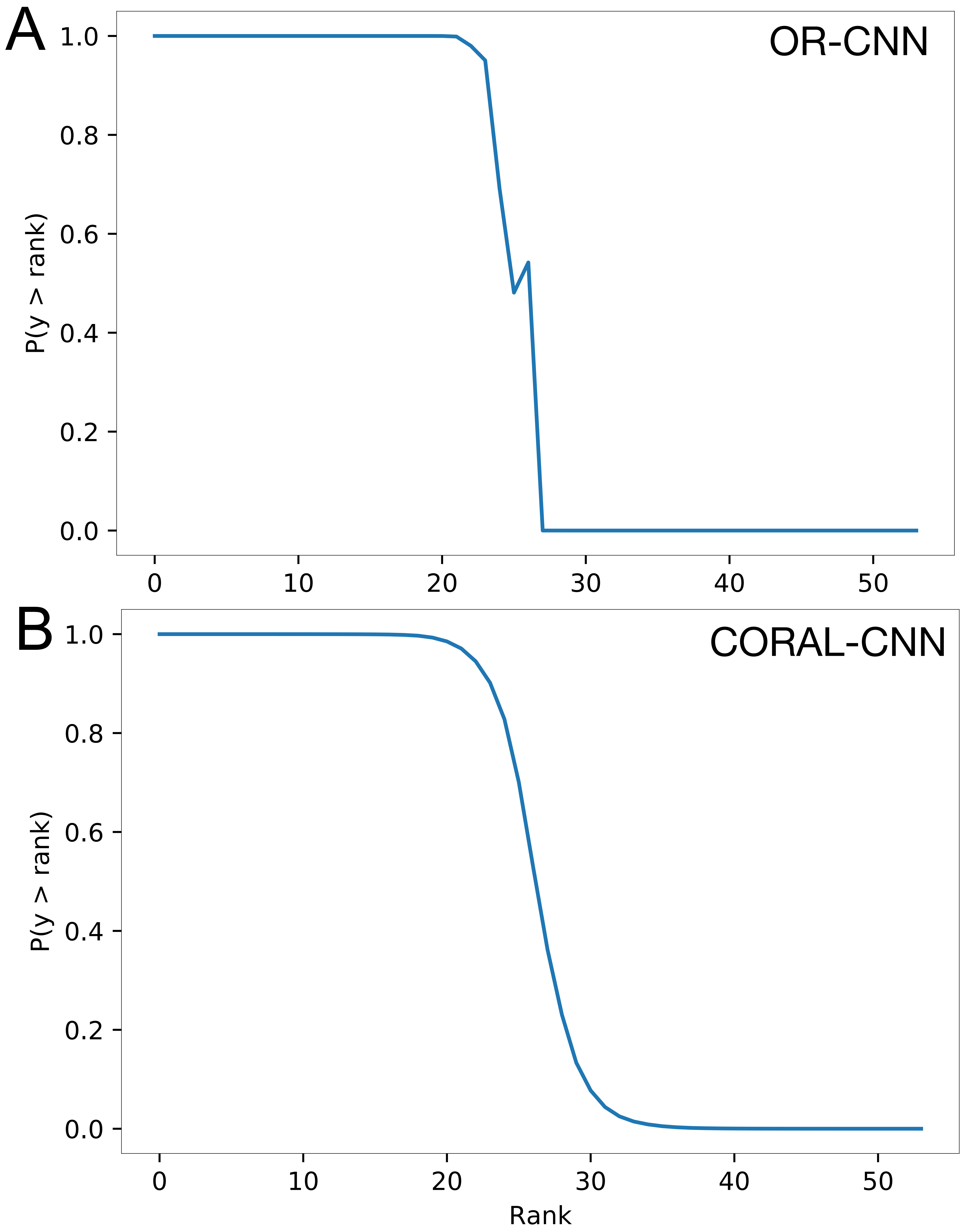
Wenzhi Cao, Vahid Mirjalili, and Sebastian Raschka. "Rank-consistent ordinal regression for neural networks." *arXiv:1901.07884* (2019).

3.3. *Theoretical Guarantees for Classifier Consistency*

The following theorem shows that by minimizing the loss L (Eq. 3), the learned bias units of the output layer are non-increasing such that $b_1 \geq b_2 \geq \ldots \geq b_{K-1}$. Consequently, the predicted confidence scores or probability estimates of the $K-1$ tasks are decreasing, i.e.,

$$\widehat{P}\left(y_i^{(1)} = 1\right) \geq \widehat{P}\left(y_i^{(2)} = 1\right) \geq \ldots \geq \widehat{P}\left(y_i^{(K-1)} = 1\right) \qquad (5)$$

for all i, ensuring classifier consistency. Consequently, $\{f_k\}_{k=1}^{K-1}$ given by Eq. 4 are also rank-monotonic.



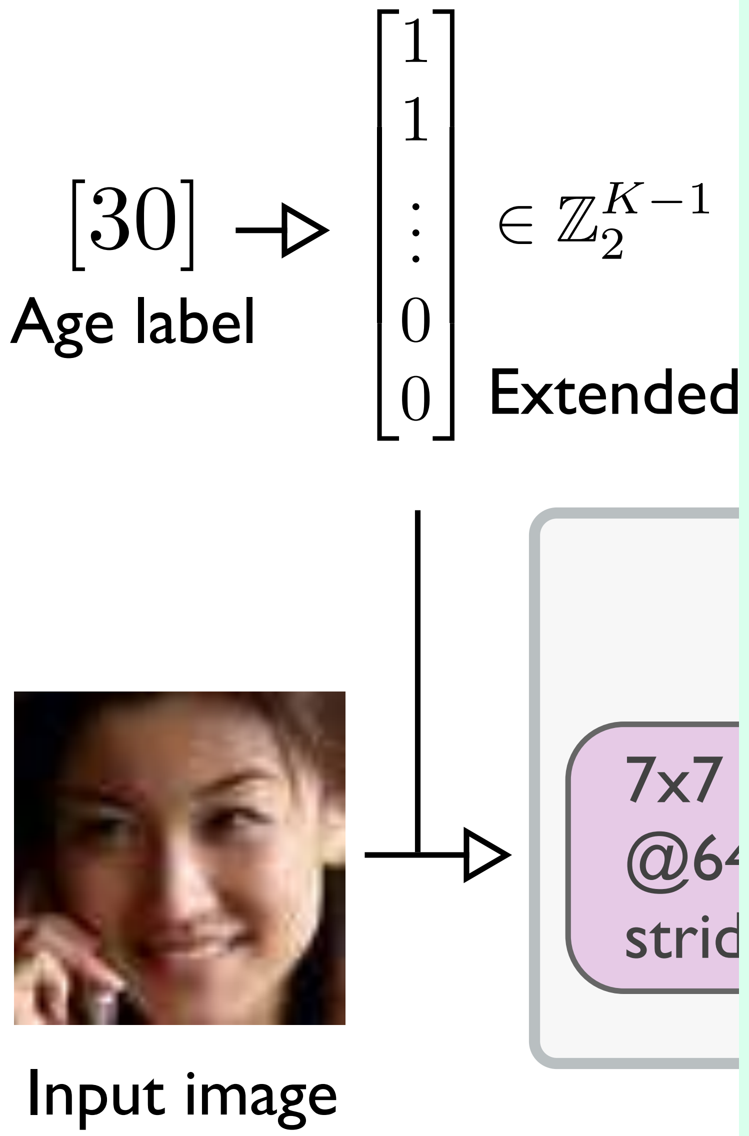
Wenzhi Cao, Vahid Mirjalili, and Sebastian Raschka. "Rank-consistent ordinal regression for neural networks." *arXiv:1901.07884* (2019).

Table 2
Average numbers of inconsistencies occurred on the different test datasets for CORAL-CNN and Niu et al's Ordinal CNN. The penultimate column and last column list the average numbers of inconsistencies focussing only on the correct and incorrect age predictions, respectively.

	CORAL-CNN All predictions	Ordinal-CNN [1] All predictions	Ordinal-CNN [1] Only correct predictions	Ordinal-CNN [1] Only incorrect predictions
Morph				
Seed 0	0	2.74	2.02	2.89
Seed 1	0	2.74	2.08	2.88
Seed 2	0	3.00	2.20	3.16
AFAD				
Seed 0	0	2.32	1.78	2.40
Seed 1	0	2.35	1.83	2.43
Seed 2	0	2.55	1.97	2.63
UTKFace				
Seed 0	0	4.79	3.64	4.92
Seed 1	0	5.73	4.05	5.95
Seed 2	0	5.07	3.84	5.21
CACD				
Seed 0	0	5.06	4.55	5.10
Seed 1	0	5.40	4.76	5.44
Seed 2	0	5.56	4.87	5.61

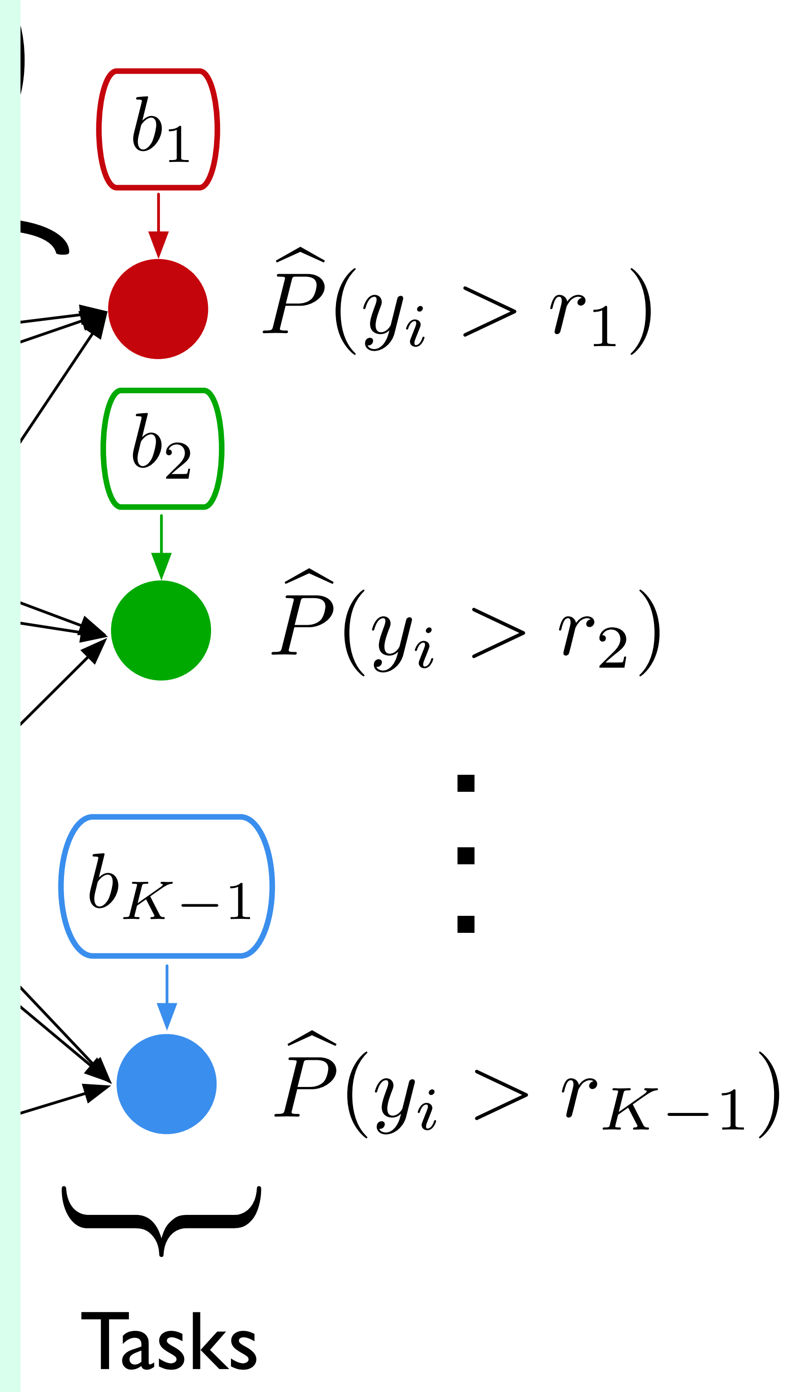
Wenzhi Cao, Vahid Mirjalili, and Sebastian Raschka. "Rank-consistent ordinal regression for neural networks." *arXiv:1901.07884* (2019).

Deep Learning & Ordinal Data

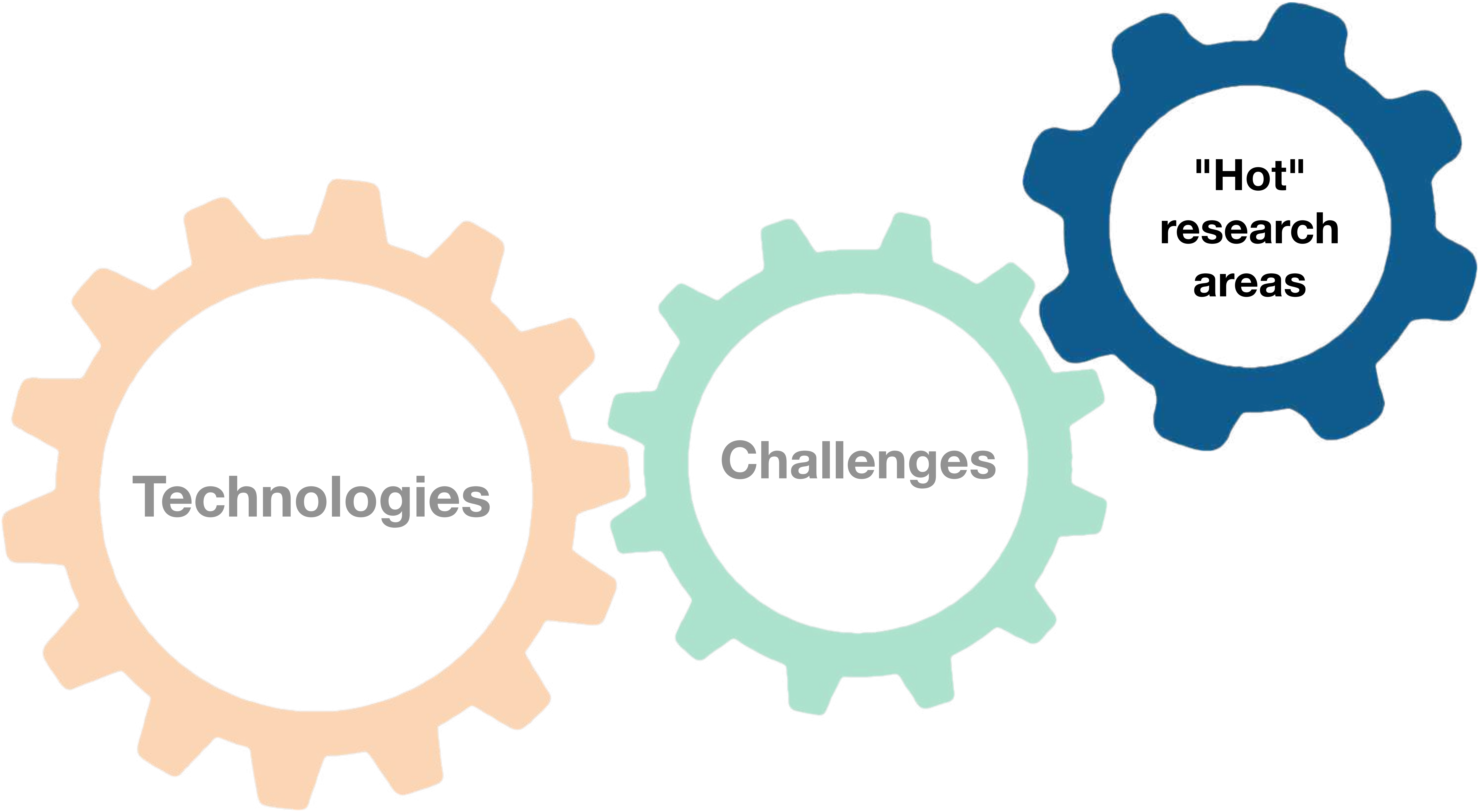


Age prediction errors on the test sets *without* task ir

Method	Random Seed	MORPH-2	
		MAE	RMSE
CE-CNN	0	3.40	4.88
	1	3.39	4.87
	2	3.37	4.87
	AVG ± SD	3.39 ± 0.02	4.89 ± 0.01
OR-CNN [1]	0	2.98	4.26
	1	2.98	4.26
	2	2.96	4.20
	AVG ± SD	2.97 ± 0.01	4.24 ± 0.03
CORAL-CNN (ours)	0	2.68	3.75
	1	2.63	3.66
	2	2.61	3.64
	AVG ± SD	**2.64 ± 0.04**	**3.68 ± 0.06**



Wenzhi Cao, Vahid Mirjalili, and Sebastian Raschka. "Rank-consistent ordinal regression for neural networks." *arXiv:1901.07884* (2019).


Technologies
Challenges
"Hot"
research
areas

Transformers

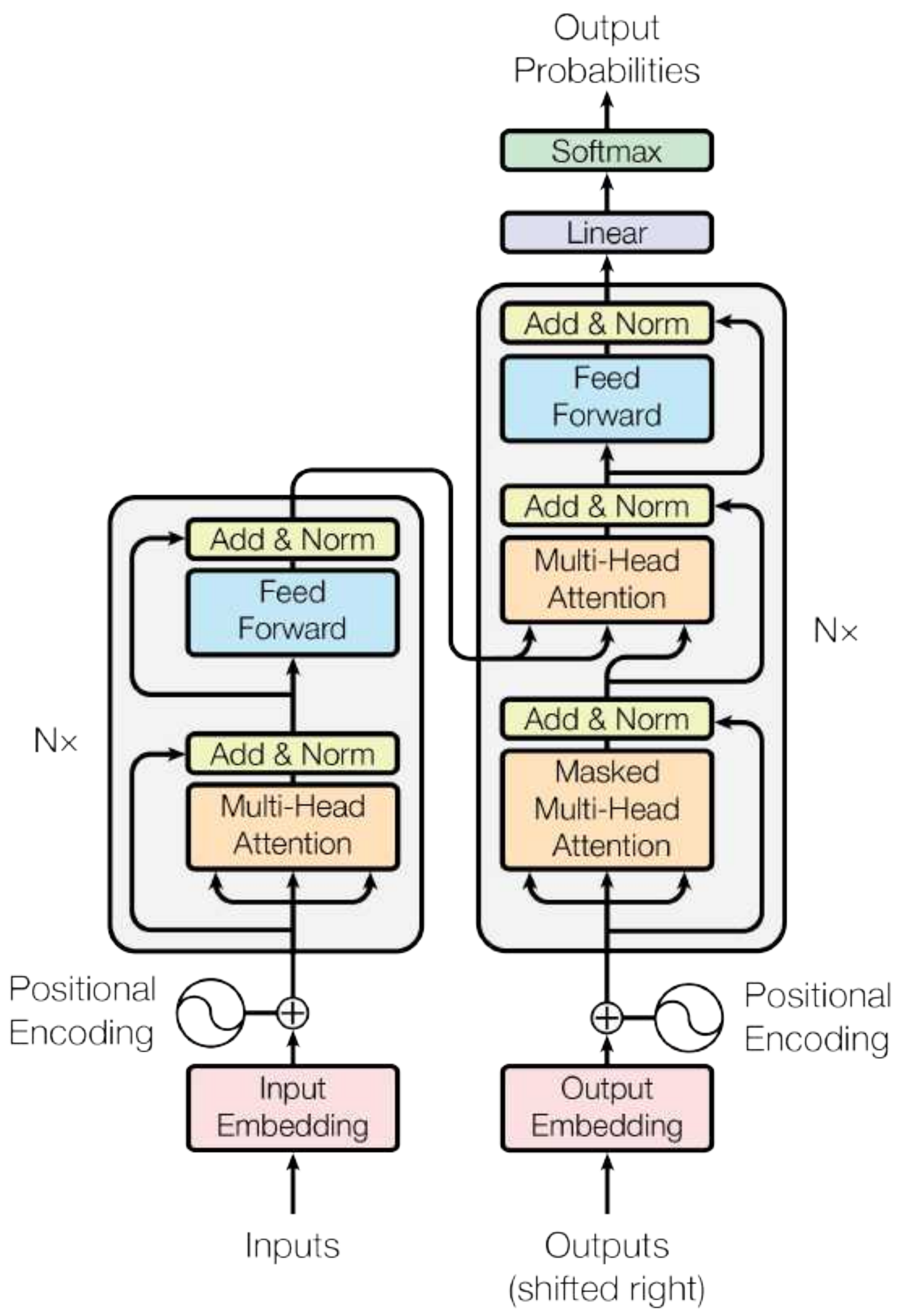


Figure 1: The Transformer - model architecture.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. InAdvances in neural information processing systems 2017 (pp. 5998-6008).

New AI fake text generator may be too dangerous to release, say creators

The Elon Musk-backed nonprofit company OpenAI declines to release research publicly for fear of misuse

https://www.theguardian.com/technology/2019/feb/14/elon-musk-backed-ai-writes-convincing-news-fiction

"Google researchers have trained a chatbot with uncannily good conversational skills. The bot, named Meena, is a 2.6 billion parameter language model trained on 341GB of text data, filtered from public domain social media conversations. Meena uses a seq2seq model (the same sort of technology that powers Google's "Smart Compose" feature in gmail), paired with an Evolved Transformer encoder and decoder - it's interesting to see something like this depend so much on a component developed via neural architecture search."

https://mailchi.mp/jack-clark/import-ai-183-curve-fitting-conversation-with-meena-gans-paint-a-climate-change-future-and-what-compute-data-arbitrage-means-for-simulators

Human: Hi!

Meena: Hey there! What's up?

Human: Same old, Monday blues...what are you up to?

Meena: I'm procrastinating on a paper I'm supposed to be writing.

Human: are you thinking of pulling an all nighter?

Meena: Possibly. What's your paper on?

Human: Nah, i just do the grunge work around here...one day I wish to write a paper that will change the world!

Meena: What's the topic?

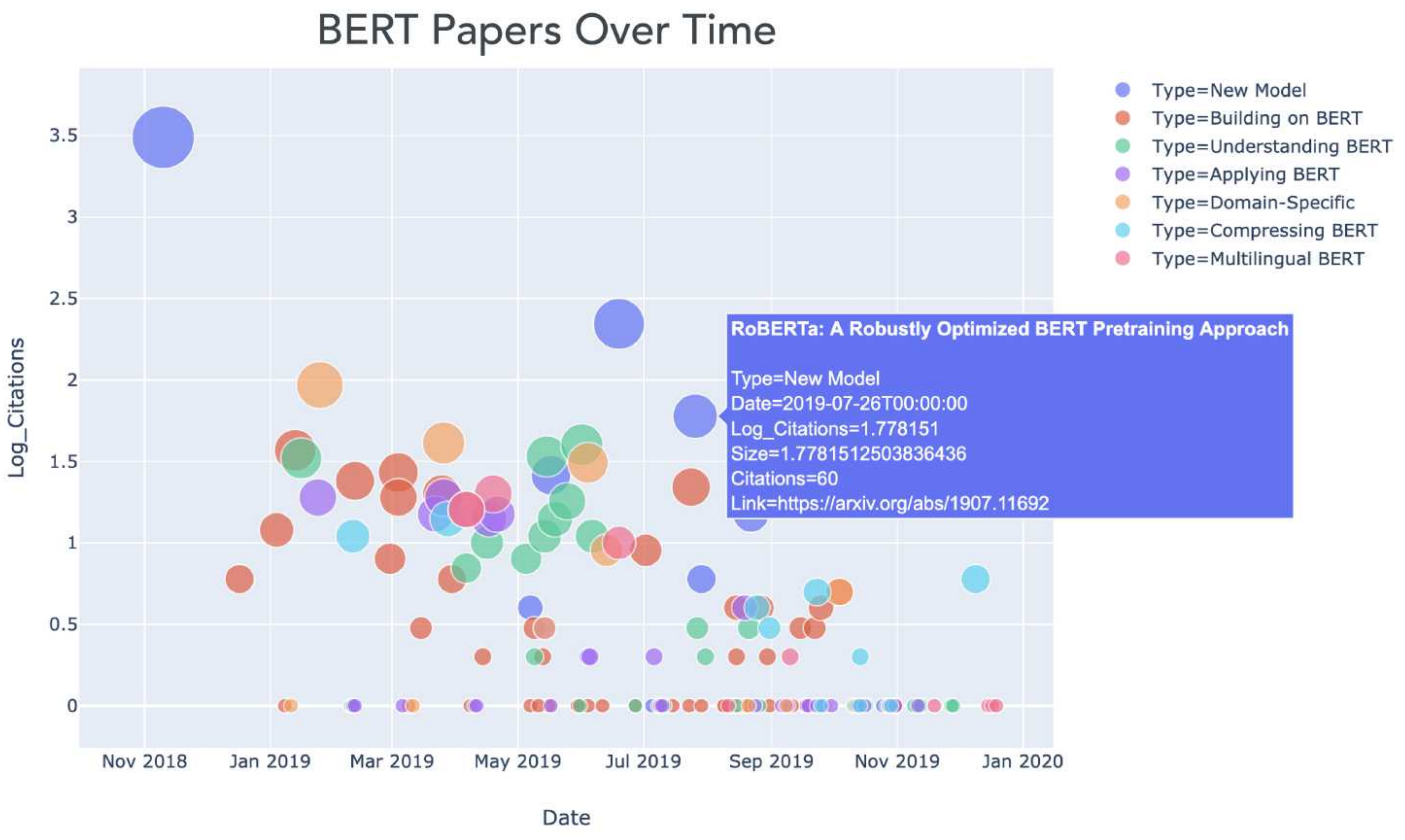
Human: not sure, any suggestions?

Meena: Well what's the assignment on?

Human: breathing under water

https://github.com/google-research/google-research/blob/master/meena/meena.txt

Image Source: https://medium.com/huggingface/distilbert-8cf3380435b5



https://github.com/nslatysheva/BERT_papers/blob/master/BERT_Papers.csv

THE COST OF TRAINING NLP MODELS
A CONCISE OVERVIEW

Or Sharir
AI21 Labs
ors@ai21.com

Barak Peleg
AI21 Labs
barakp@ai21.com

Yoav Shoham
AI21 Labs
yoavs@ai21.com

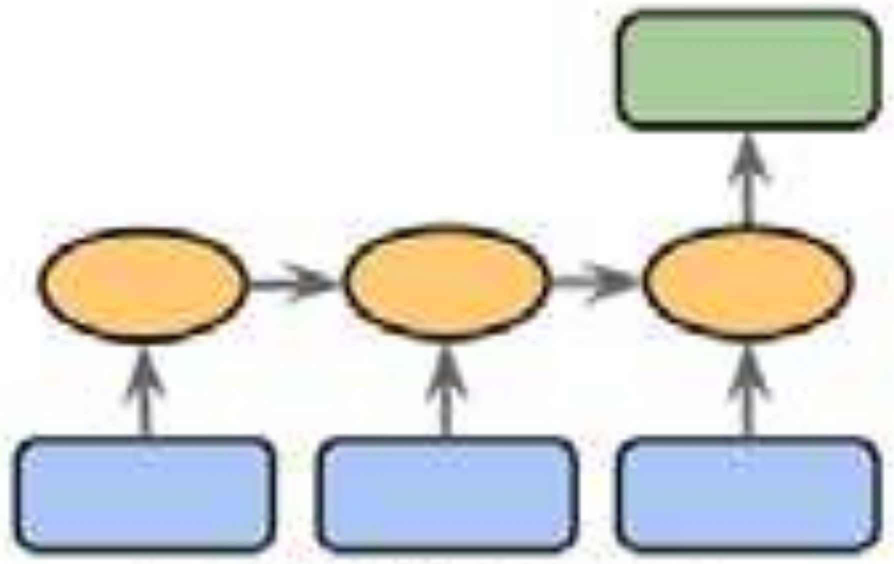
April 2020

http://arxiv.org/abs/2004.08900

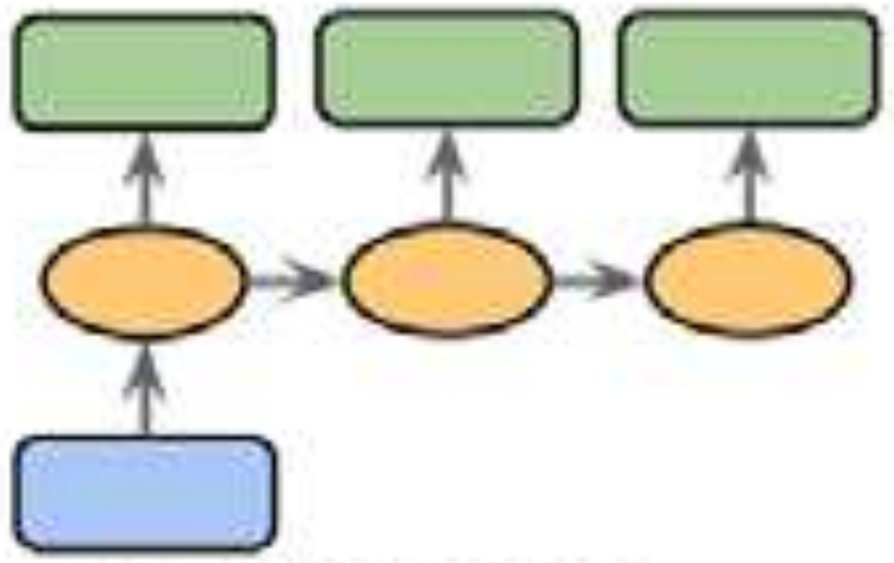
Costs: Not for the faint hearted

- $2.5k - $50k (110 million parameter model)
- $10k - $200k (340 million parameter model)
- $80k - $1.6m (1.5 billion parameter model)

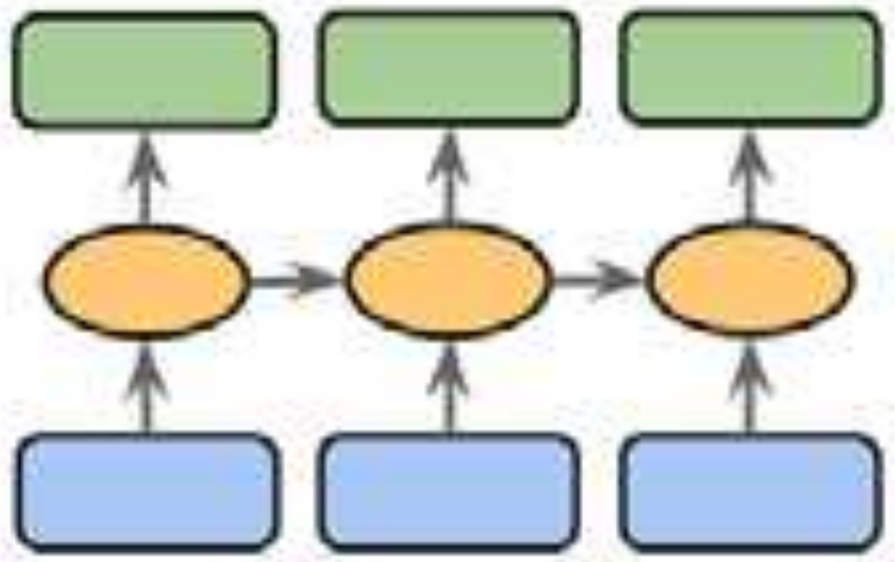
Recurrent Neural Networks



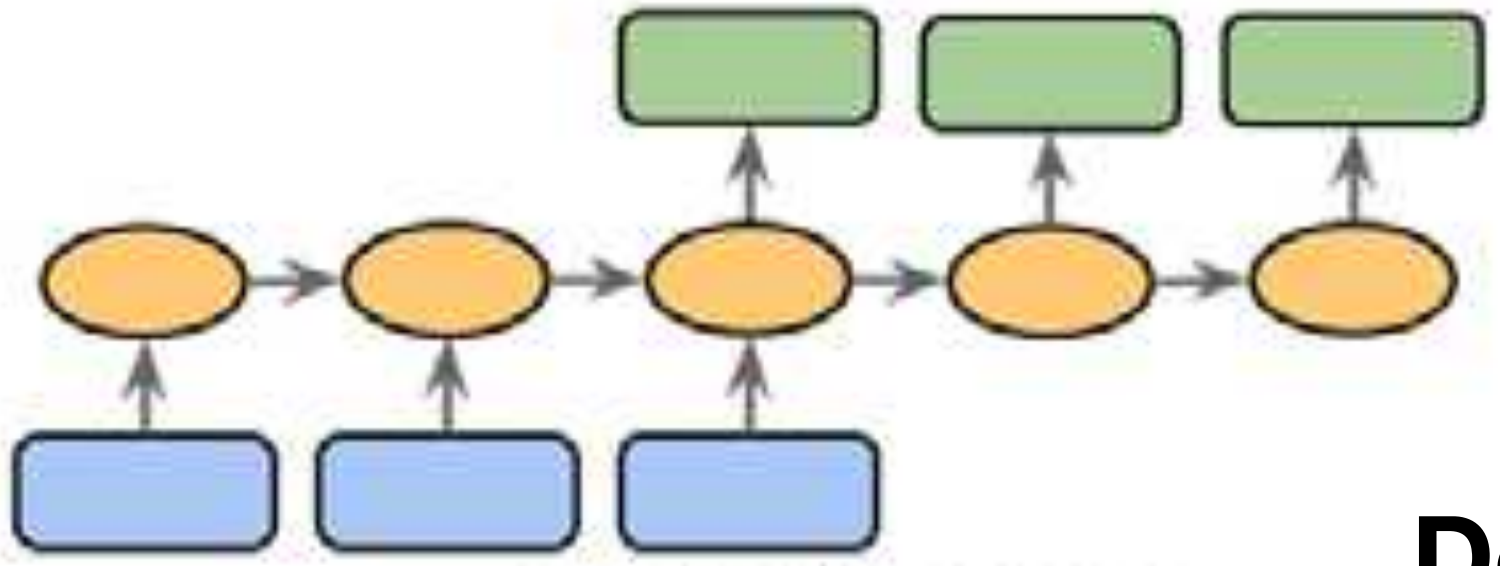
many-to-one



one-to-many



many-to-many



many-to-many

Despite LSTM, it may be hard to memorize long sequences/ sentences (e.g., for language translation)

Attention Mechanism

- originally developed for language translation:
  Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

"... allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word ..."

"traditional" encoder+decoder RNN

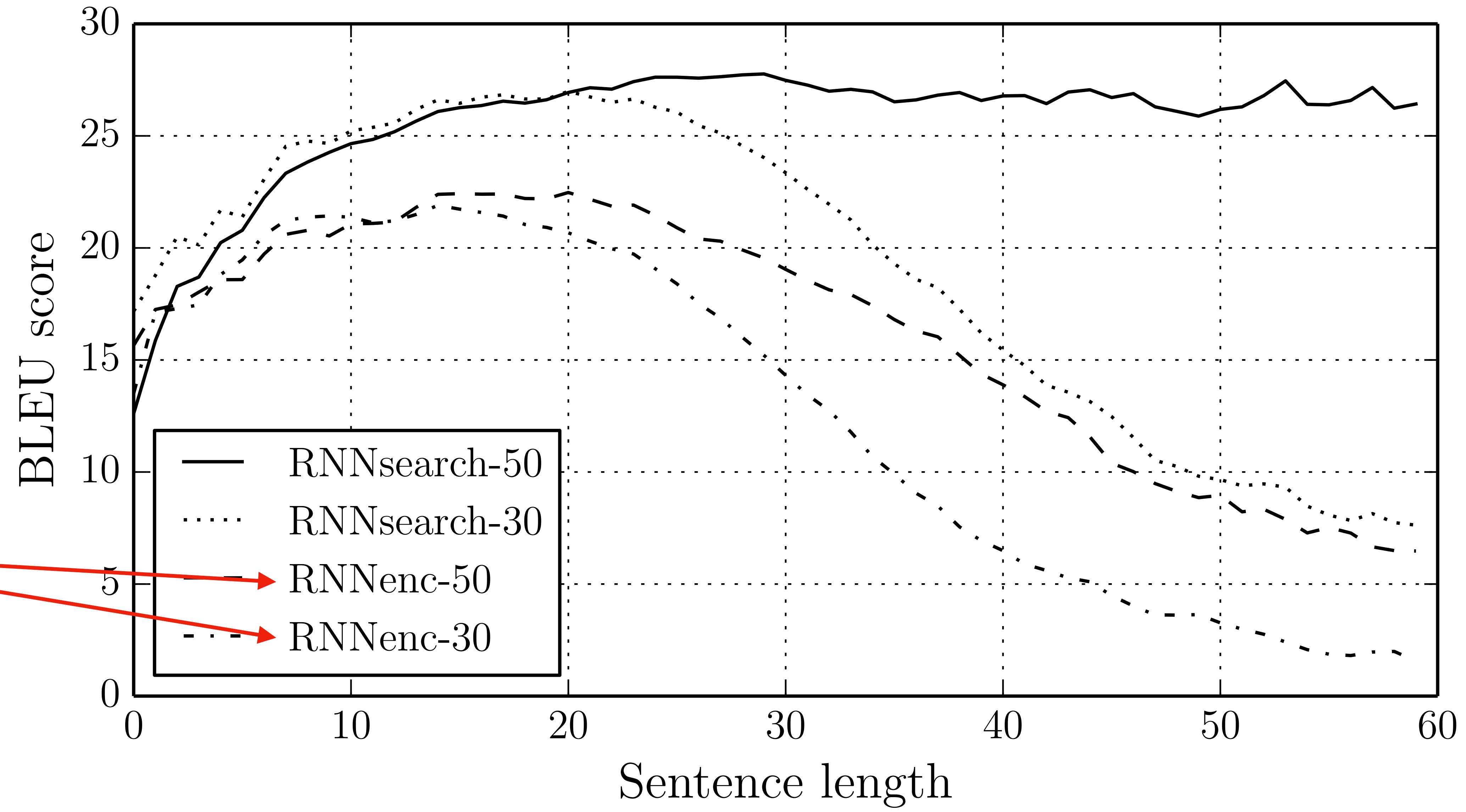


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

Attention Mechanism

- originally developed for language translation:
  Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

"... allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word ..."

Assign attention weight to each word to know how much "attention" the model should pay to each word (i.e., for each word, the network learns a "context")

Attention Mechanism

- originally developed for language translation:
  Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

hidden state

attention weight

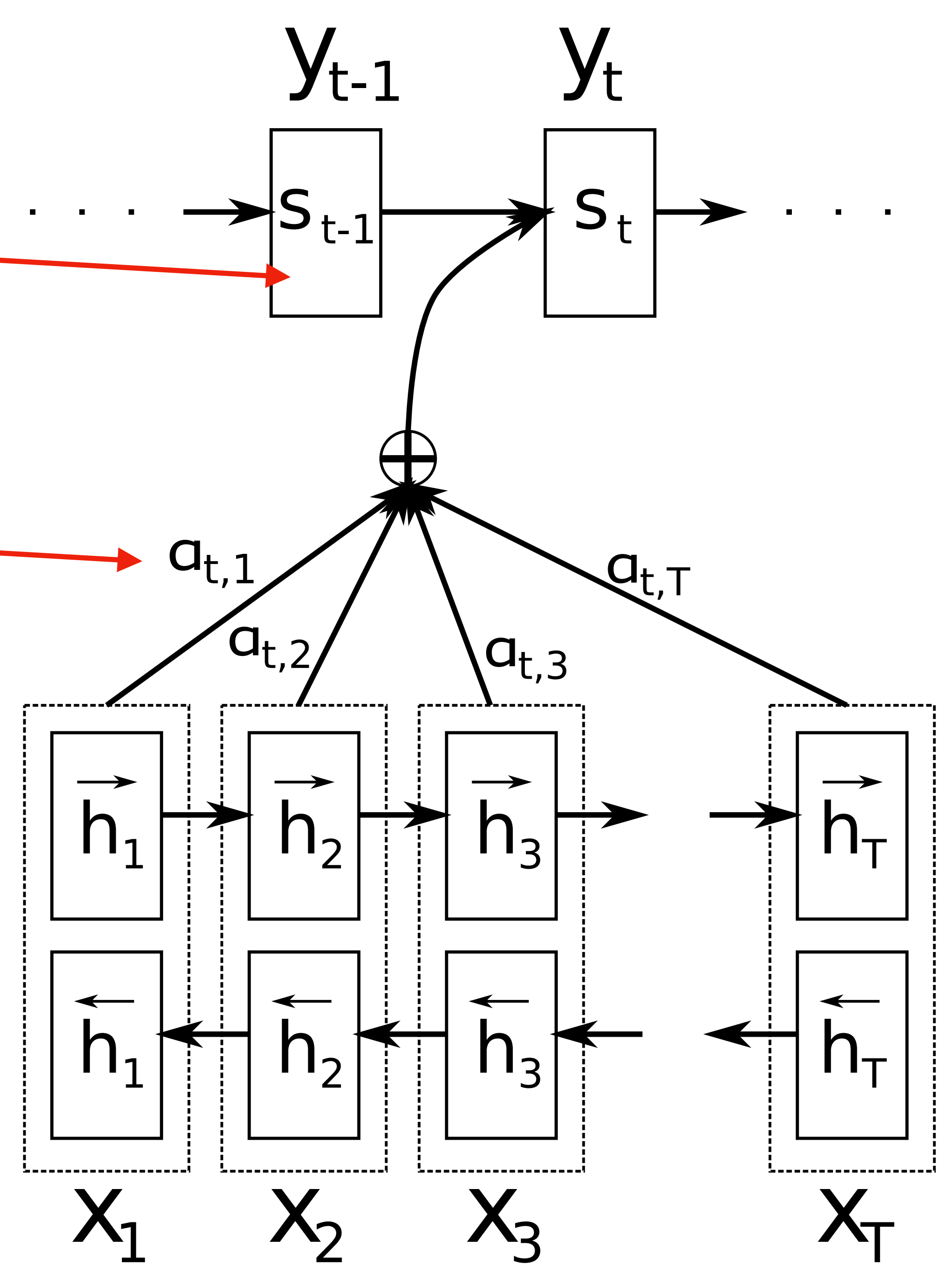
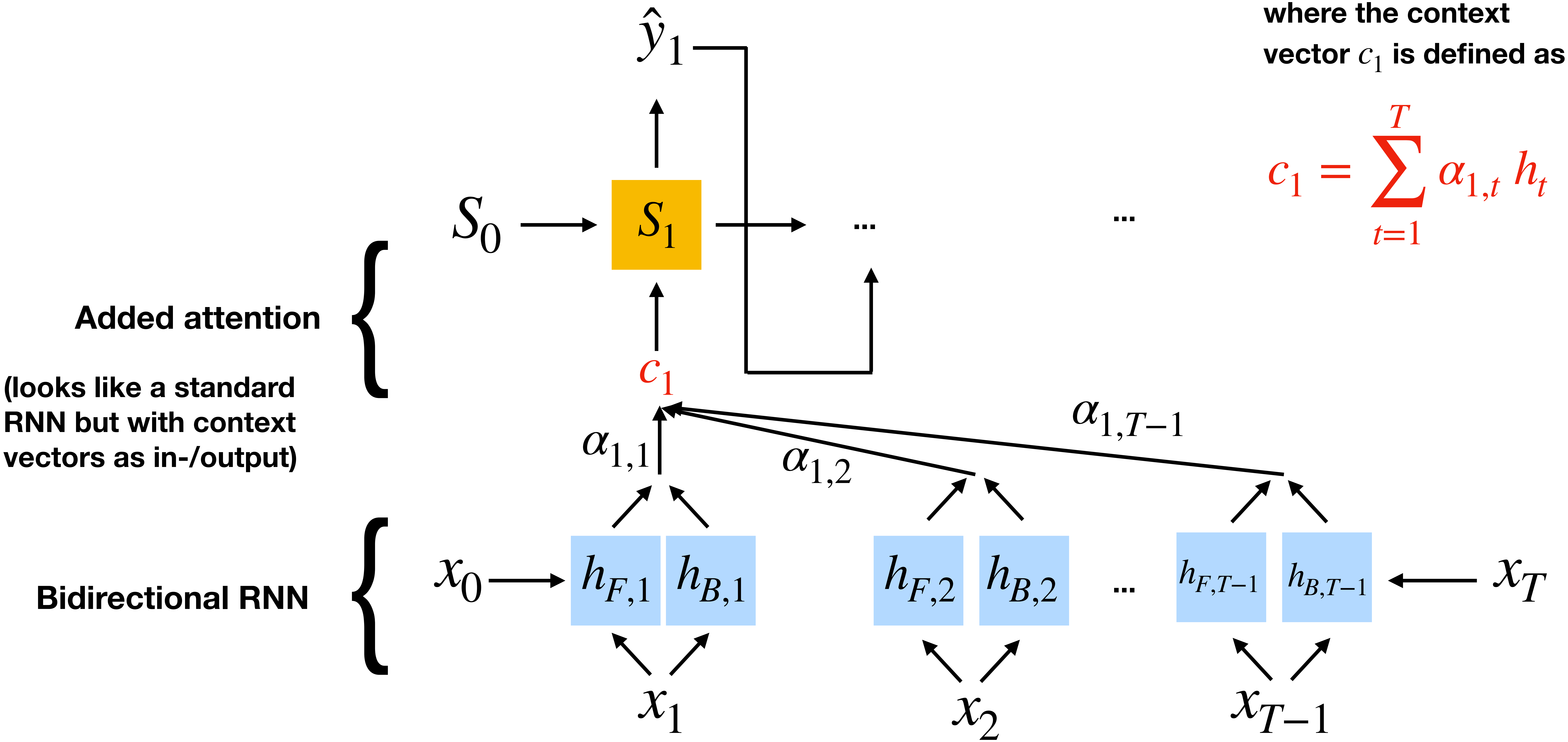


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Assign attention weight to each word to know how much "attention" the model should pay to each word (i.e., for each word, the network learns a "context")

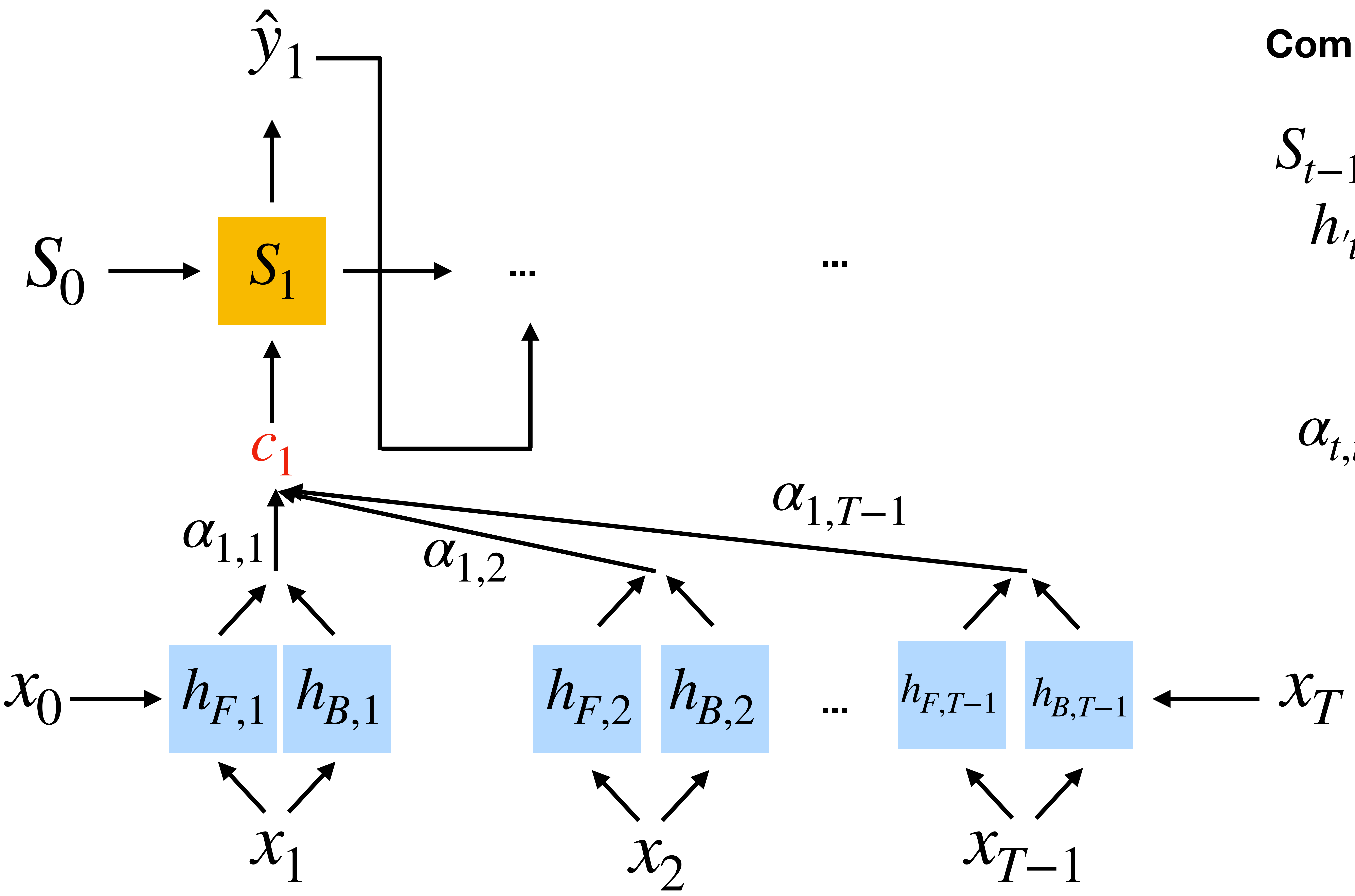
RNN Attention Mechanism



where the context vector c_1 is defined as

$$c_1 = \sum_{t=1}^{T} \alpha_{1,t} \, h_t$$

RNN Attention Mechanism

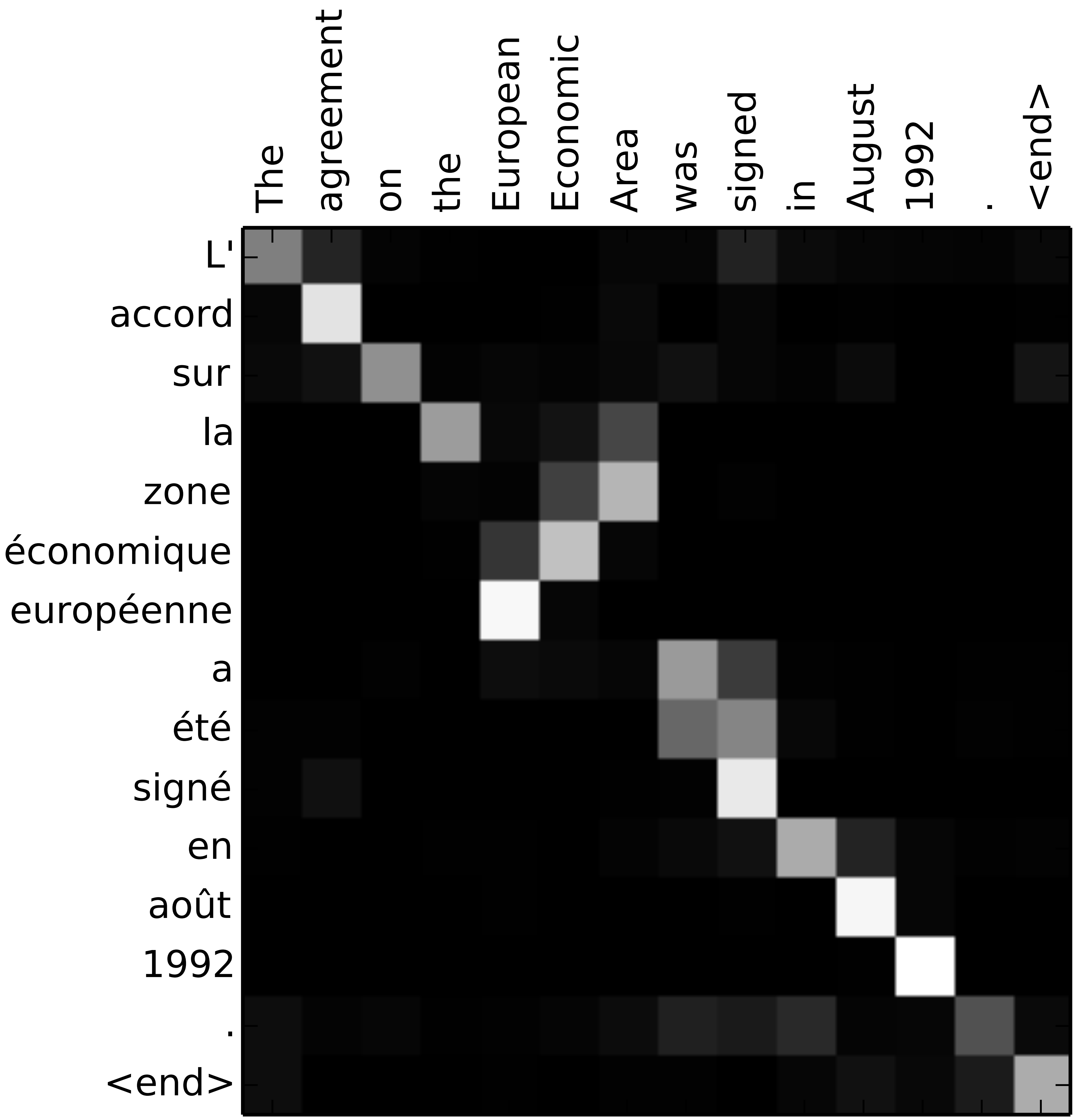


Computing attention weights

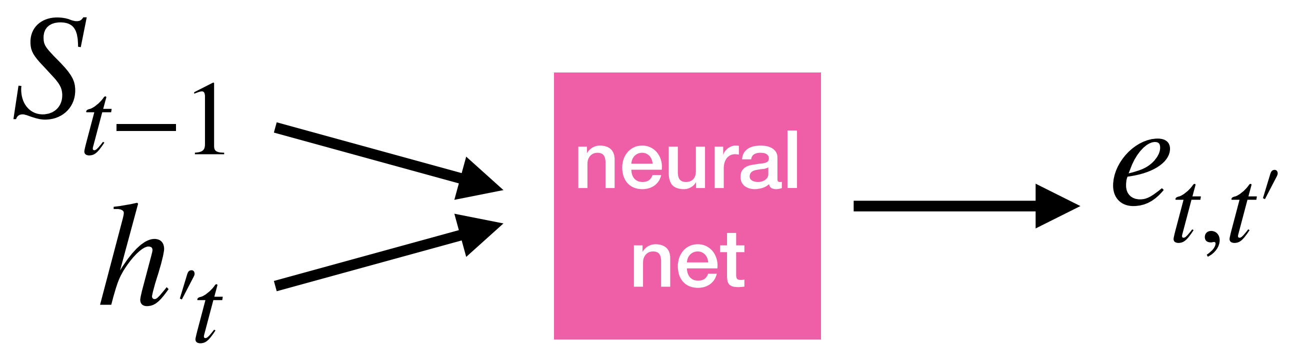
S_{t-1}, h'_t → neural net → $e_{t,t'}$

$$\alpha_{t,t'} = \frac{\exp(e_{t,t'})}{\sum_{t'=1}^{T} \exp(e_{t,t'})}$$

RNN Attention Mechanism



Computing attention weights



$$\alpha_{t,t'} = \frac{\exp(e_{t,t'})}{\sum_{t'=1}^{T} \exp(e_{t,t'})}$$

Self-Attention Mechanism

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

https://arxiv.org/abs/1706.03762

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

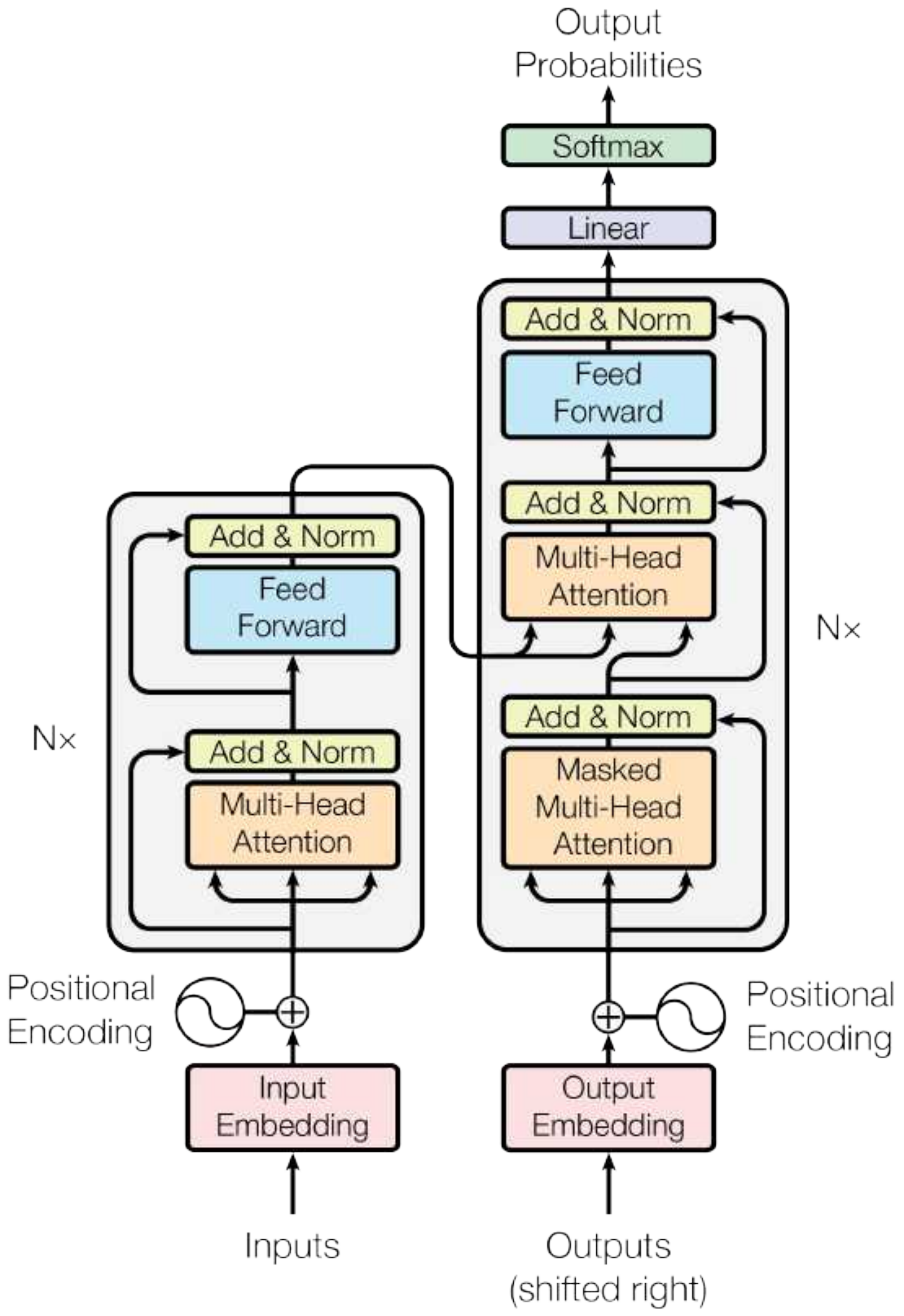
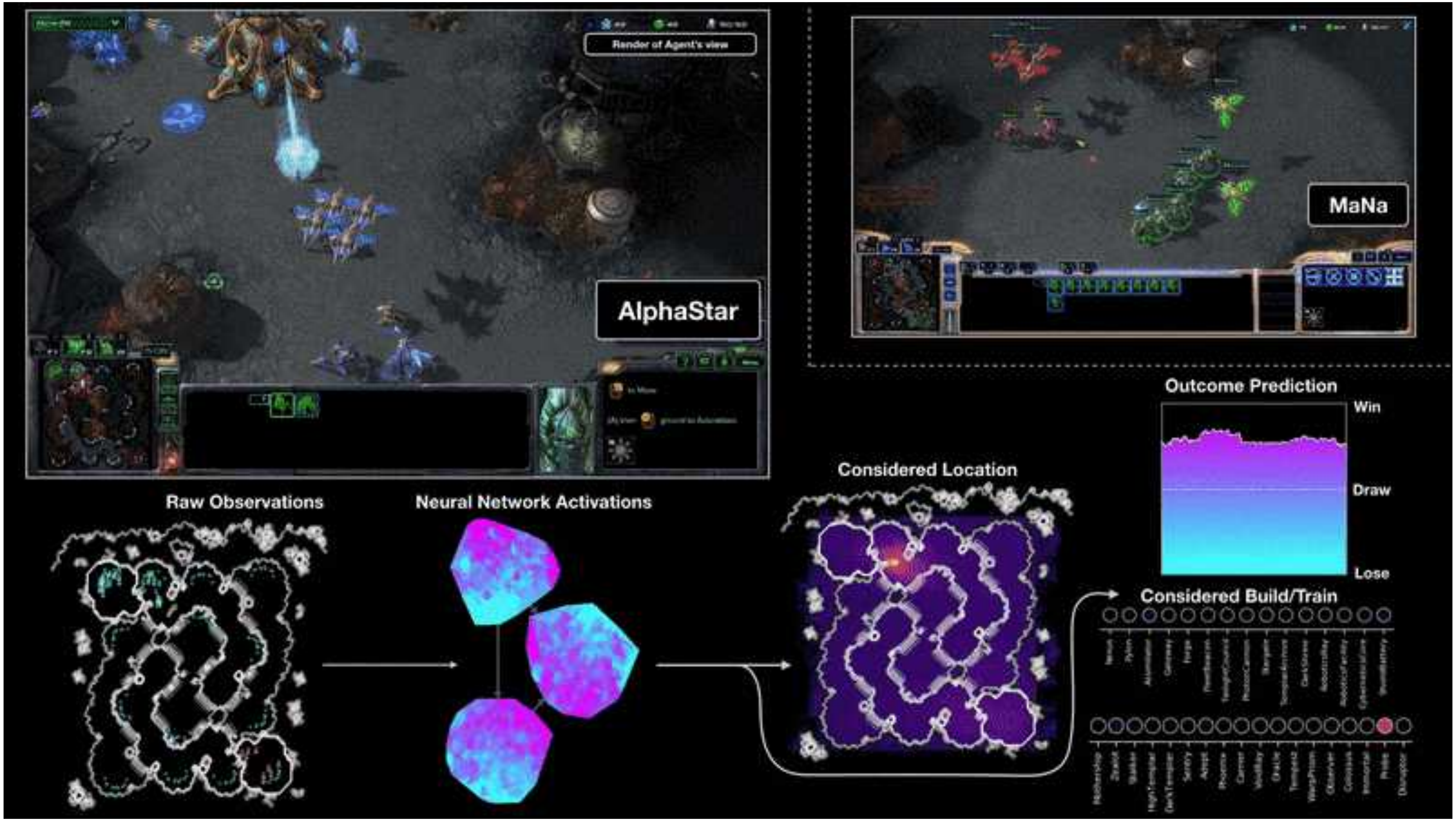


Figure 1: The Transformer - model architecture.

AlphaStar applies a transformer torso to the units



https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

Transformer on a Diet

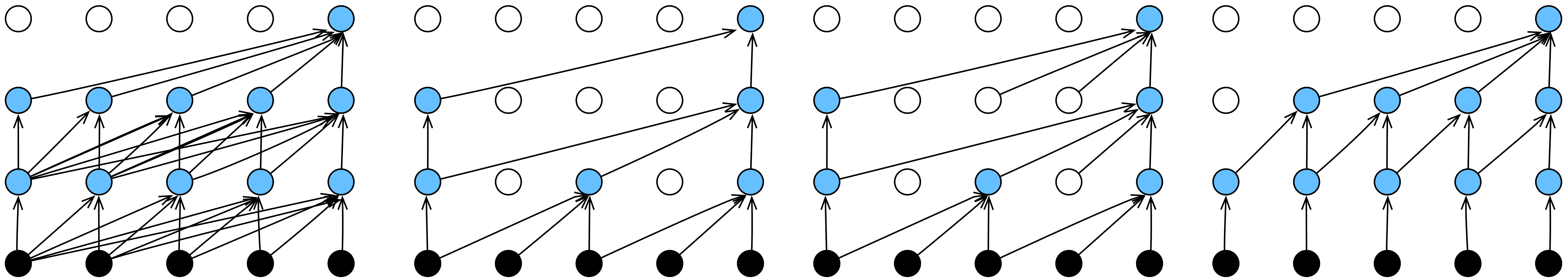
Chenguang Wang **Zihao Ye** **Aston Zhang**
Zheng Zhang **Alexander J. Smola**
Amazon Web Services
{chgwang, yeziha, astonz, zhaz, smola}@amazon.com

Abstract

Transformer has been widely used thanks to its ability to capture sequence information in an efficient way. However, recent developments, such as BERT and GPT-2, deliver only heavy architectures with a focus on effectiveness. In this paper, we explore three carefully-designed light Transformer architectures to figure out whether the Transformer with less computations could produce competitive re-

size of the model. Therefore a light version of the standard Transformer architecture is expected to relieve the heavy computation issue and compress the model to ease the deployment in real world applications.

In this paper, we carefully design several light Transformer architectures. The intuition behind the light Transformers is: preserving the Transformer connections that are useful to capture the essential



(a) Full Transformer. (b) Dilated Transformer. (c) Dilated Transformer with memory. (d) Cascade Transformer.

Graph Attention Networks

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

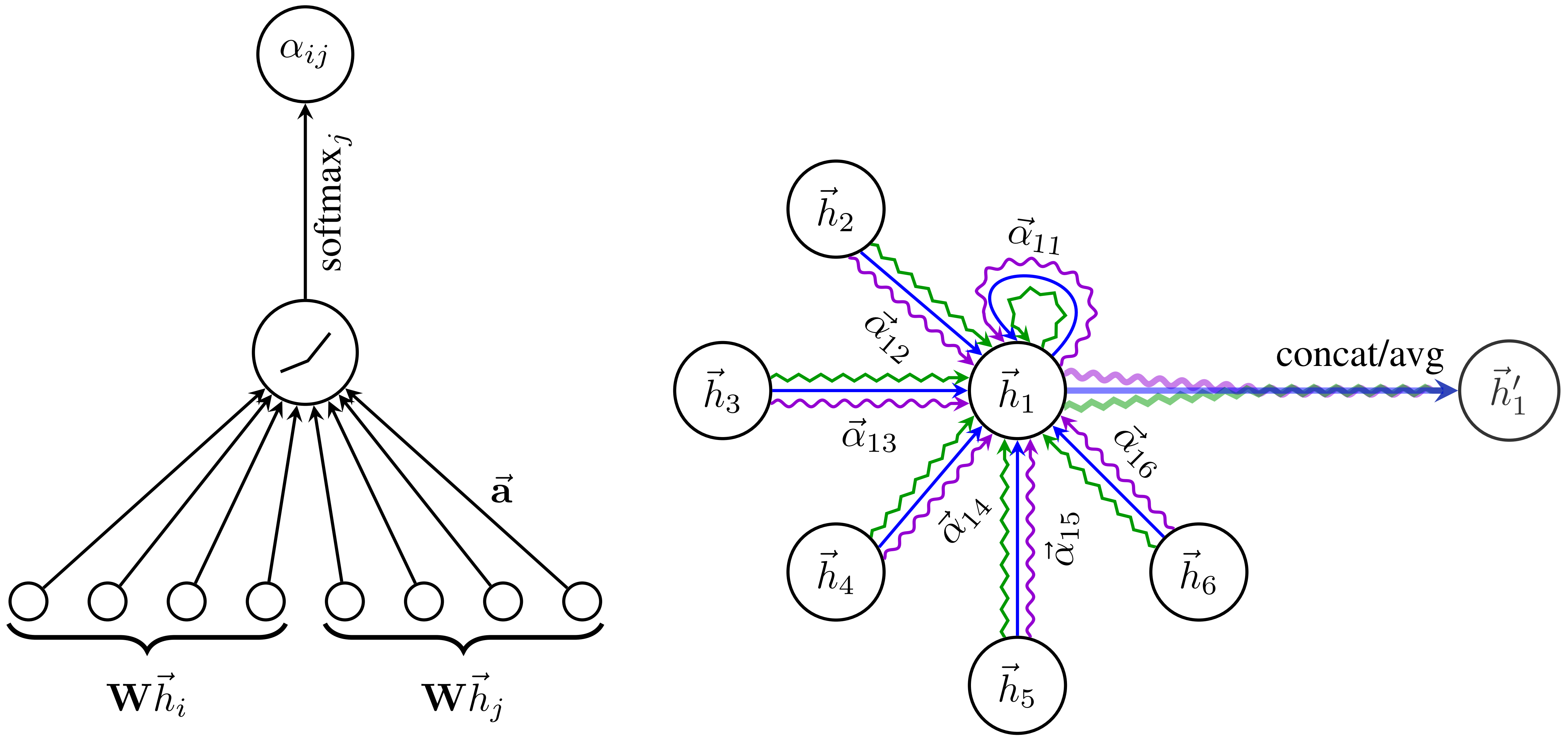
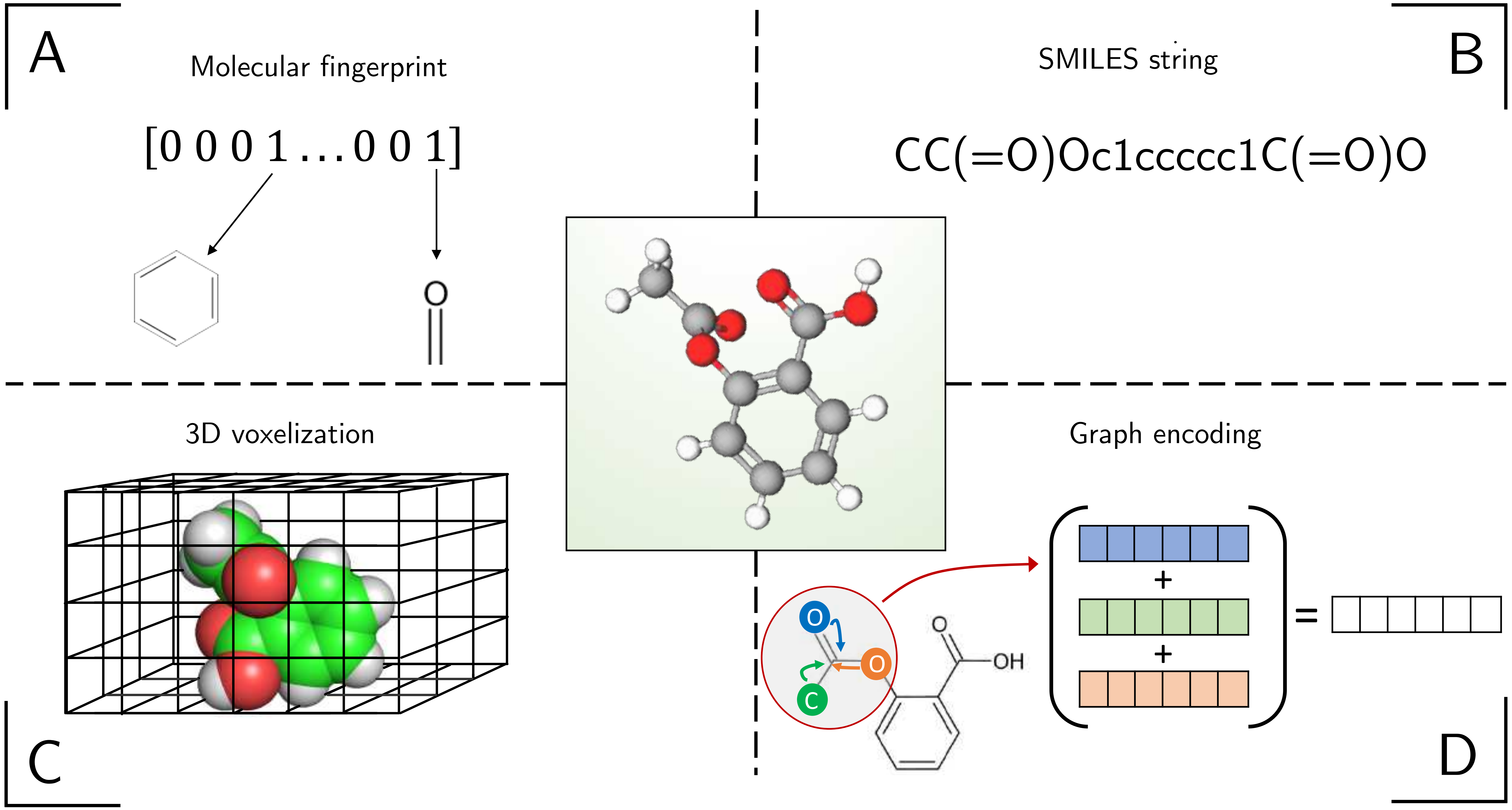


Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

https://arxiv.org/abs/1710.10903

Graph Neural Networks -- Why Graphs?



Sebastian Raschka and Benjamin Kaufman (2020)
Machine learning and AI-based approaches for bioactive ligand discovery and GPCR-ligand recognition arXiv:2001.06545

PyTorch geometric

pypi package 1.4.2 | build passing | docs passing | codecov 95% | contributions welcome

Documentation | **Paper** | **External Resources**

PyTorch Geometric (PyG) is a geometric deep learning extension library for PyTorch.

It consists of various methods for deep learning on graphs and other irregular structures, also known as *geometric deep learning*, from a variety of published papers. In addition, it consists of an easy-to-use mini-batch loader for many small and single giant graphs, multi gpu-support, a large number of common benchmark datasets (based on simple interfaces to create your own), and helpful transforms, both for learning on arbitrary graphs as well as on 3D meshes or point clouds.

https://github.com/rusty1s/pytorch_geometric

In detail, the following methods are currently implemented:

- **SplineConv** from Fey *et al.*: SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels (CVPR 2018)
- **GCNConv** from Kipf and Welling: Semi-Supervised Classification with Graph Convolutional Networks (ICLR 2017)
- **ChebConv** from Defferrard *et al.*: Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering (NIPS 2016)
- **NNConv** from Gilmer *et al.*: Neural Message Passing for Quantum Chemistry (ICML 2017)
- **CGConv** from Xie and Grossman: Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties (Physical Review Letters 120, 2018)
- **ECConv** from Simonovsky and Komodakis: Edge-Conditioned Convolution on Graphs (CVPR 2017)
- **GATConv** from Veličković *et al.*: Graph Attention Networks (ICLR 2018)
- **SAGEConv** from Hamilton *et al.*: Inductive Representation Learning on Large Graphs (NIPS 2017)
- **GraphConv** from, *e.g.*, Morris *et al.*: Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks (AAAI 2019)
- **GatedGraphConv** from Li *et al.*: Gated Graph Sequence Neural Networks (ICLR 2016)
- **GINConv** from Xu *et al.*: How Powerful are Graph Neural Networks? (ICLR 2019)
- **ARMAConv** from Bianchi *et al.*: Graph Neural Networks with Convolutional ARMA Filters (CoRR 2019)
- **SGConv** from Wu *et al.*: Simplifying Graph Convolutional Networks (CoRR 2019)
- **APPNP** from Klicpera *et al.*: Predict then Propagate: Graph Neural Networks meet Personalized PageRank (ICLR 2019)
- **AGNNConv** from Thekumparampil *et al.*: Attention-based Graph Neural Network for Semi-Supervised Learning (CoRR 2017)

- **TAGConv** from Du *et al.*: Topology Adaptive Graph Convolutional Networks (CoRR 2017)
- **RGCNConv** from Schlichtkrull *et al.*: Modeling Relational Data with Graph Convolutional Networks (ESWC 2018)
- **SignedConv** from Derr *et al.*: Signed Graph Convolutional Network (ICDM 2018)
- **DNAConv** from Fey: Just Jump: Dynamic Neighborhood Aggregation in Graph Neural Networks (ICLR-W 2019)
- **EdgeConv** from Wang *et al.*: Dynamic Graph CNN for Learning on Point Clouds (CoRR, 2018)
- **PointConv** (including **Iterative Farthest Point Sampling**, dynamic graph generation based on **nearest neighbor** or **maximum distance**, and **k-NN interpolation** for upsampling) from Qi *et al.*: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation (CVPR 2017) and PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space (NIPS 2017)
- **XConv** from Li *et al.*: PointCNN: Convolution On X-Transformed Points (official implementation) (NeurIPS 2018)
- **PPFConv** from Deng *et al.*: PPFNet: Global Context Aware Local Features for Robust 3D Point Matching (CVPR 2018)
- **GMMConv** from Monti *et al.*: Geometric Deep Learning on Graphs and Manifolds using Mixture Model CNNs (CVPR 2017)
- **FeaStConv** from Verma *et al.*: FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis (CVPR 2018)
- **HypergraphConv** from Bai *et al.*: Hypergraph Convolution and Hypergraph Attention (CoRR 2019)
- A **MetaLayer** for building any kind of graph network similar to the TensorFlow Graph Nets library from Battaglia *et al.*: Relational Inductive Biases, Deep Learning, and Graph Networks (CoRR 2018)
- **GlobalAttention** from Li *et al.*: Gated Graph Sequence Neural Networks (ICLR 2016)
- **Set2Set** from Vinyals *et al.*: Order Matters: Sequence to Sequence for Sets (ICLR 2016)
- **Sort Pool** from Zhang *et al.*: An End-to-End Deep Learning Architecture for Graph Classification (AAAI 2018)
- **Dense Differentiable Pooling** from Ying *et al.*: Hierarchical Graph Representation Learning with Differentiable Pooling (NeurIPS 2018)
- **Dense MinCUT Pooling** from Bianchi *et al.*: MinCUT Pooling in Graph Neural Networks (CoRR 2019)
- **Graclus Pooling** from Dhillon *et al.*: Weighted Graph Cuts without Eigenvectors: A Multilevel

Self-Supervised Learning

Self-Supervised Learning: Image Colorization

Zhang R, Isola P, Efros AA. Colorful image colorization. InEuropean conference on computer vision 2016 Oct 8 (pp. 649-666). Springer, Cham.

Larsson G, Maire M, Shakhnarovich G. Learning representations for automatic colorization. InEuropean Conference on Computer Vision 2016 Oct 8 (pp. 577-593). Springer, Cham.

Vondrick C, Shrivastava A, Fathi A, Guadarrama S, Murphy K. Tracking emerges by colorizing videos. InProceedings of the European Conference on Computer Vision (ECCV) 2018 (pp. 391-408).

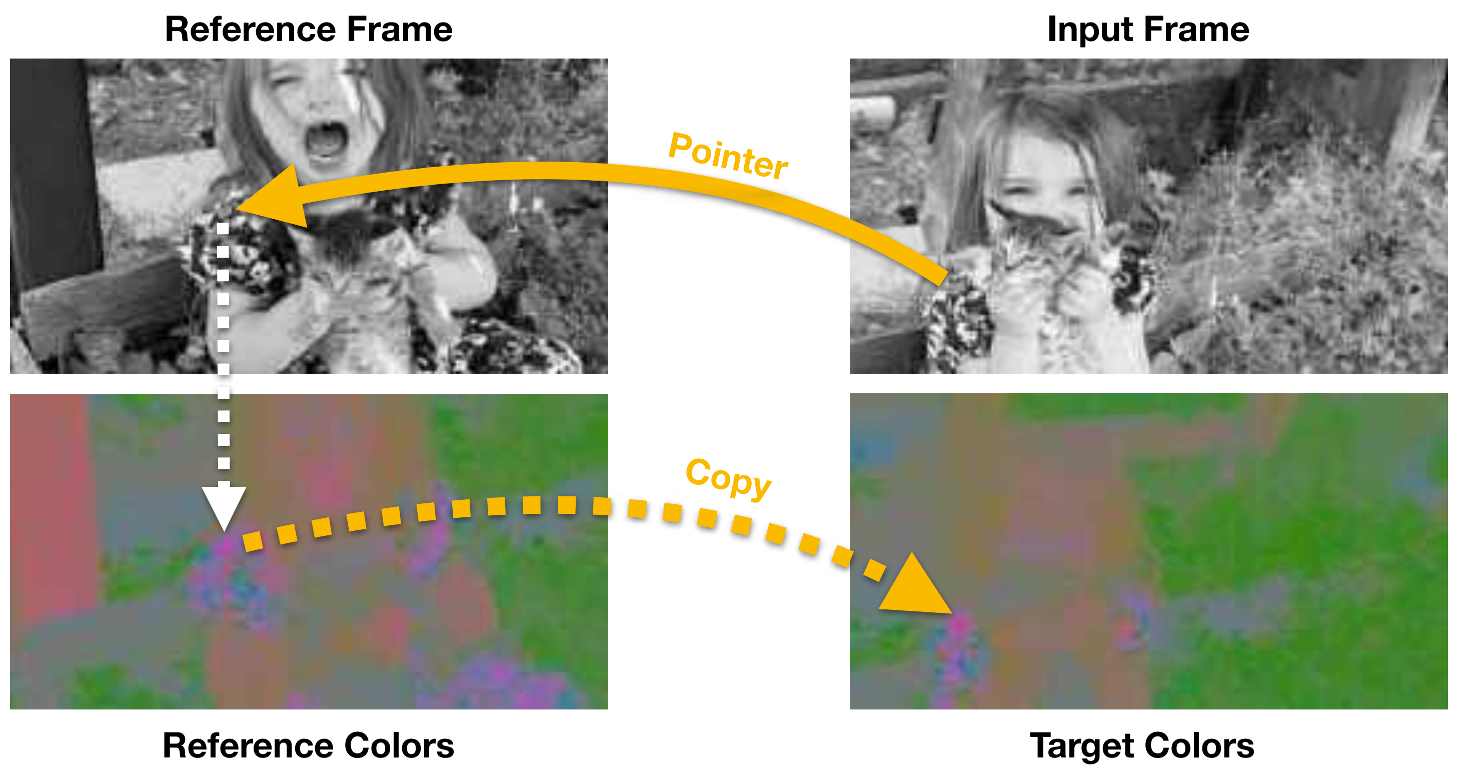
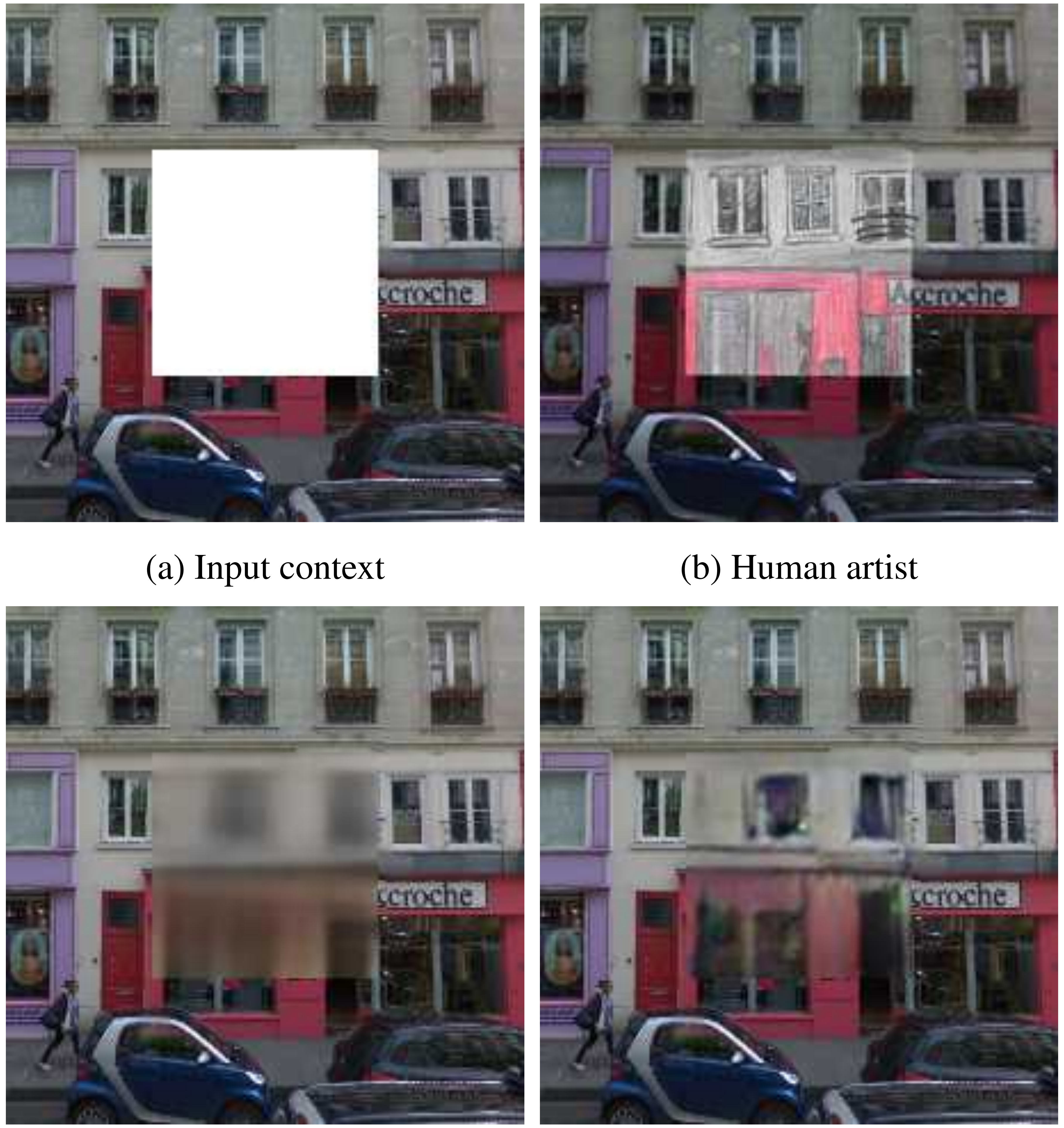


Fig. 1. Self-supervised Tracking: We capitalize on large amounts of unlabeled video to learn a self-supervised model for tracking. The model learns to predict the target colors for a gray-scale input frame by pointing to a colorful reference frame, and copying the color channels. Although we train without ground-truth labels, experiments and visualizations suggest that tracking emerges automatically in this model.

Self-Supervised Learning: Inpainting

Pathak D, Krahenbuhl P, Donahue J, Darrell T, Efros AA. Context encoders: Feature learning by inpainting. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 2536-2544).



(a) Input context

(b) Human artist

(c) Context Encoder ($L2$ loss)

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

Self-Supervised Learning: Jigsaw Puzzles & Context Predictions

Noroozi M, Favaro P. Unsupervised learning of visual representations by solving jigsaw puzzles. In European Conference on Computer Vision 2016 Oct 8 (pp. 69-84). Springer, Cham.

Doersch C, Gupta A, Efros AA. Unsupervised visual representation learning by context prediction. InProceedings of the IEEE International Conference on Computer Vision 2015 (pp. 1422-1430).

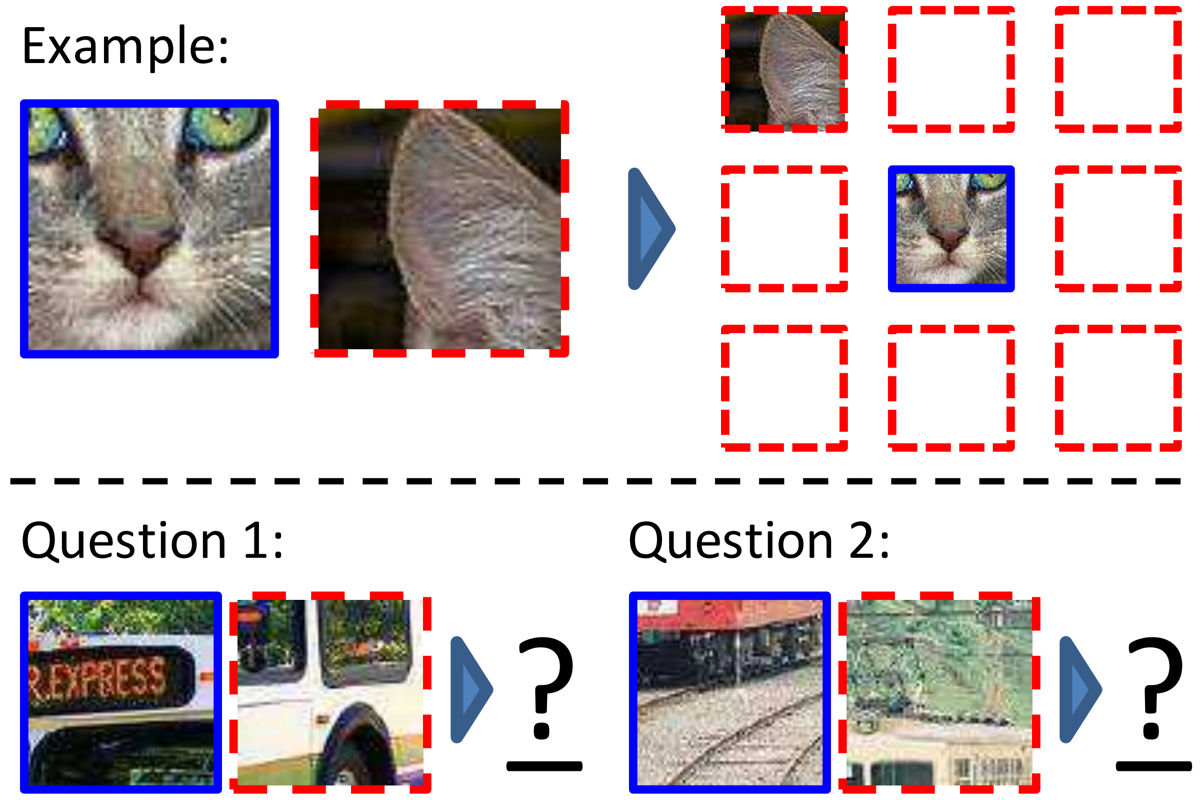


Figure 1. Our task for learning patch representations involves randomly sampling a patch (blue) and then one of eight possible neighbors (red). Can you guess the spatial configuration for the two pairs of patches? Note that the task is much easier once you have recognized the object!

Answer key: Q1: Bottom right Q2: Top center

Self-Supervised Learning: Recognizing Artifacts

Jenni S, Favaro P. Self-supervised feature learning by learning to spot artifacts. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018 (pp. 2733-2742).

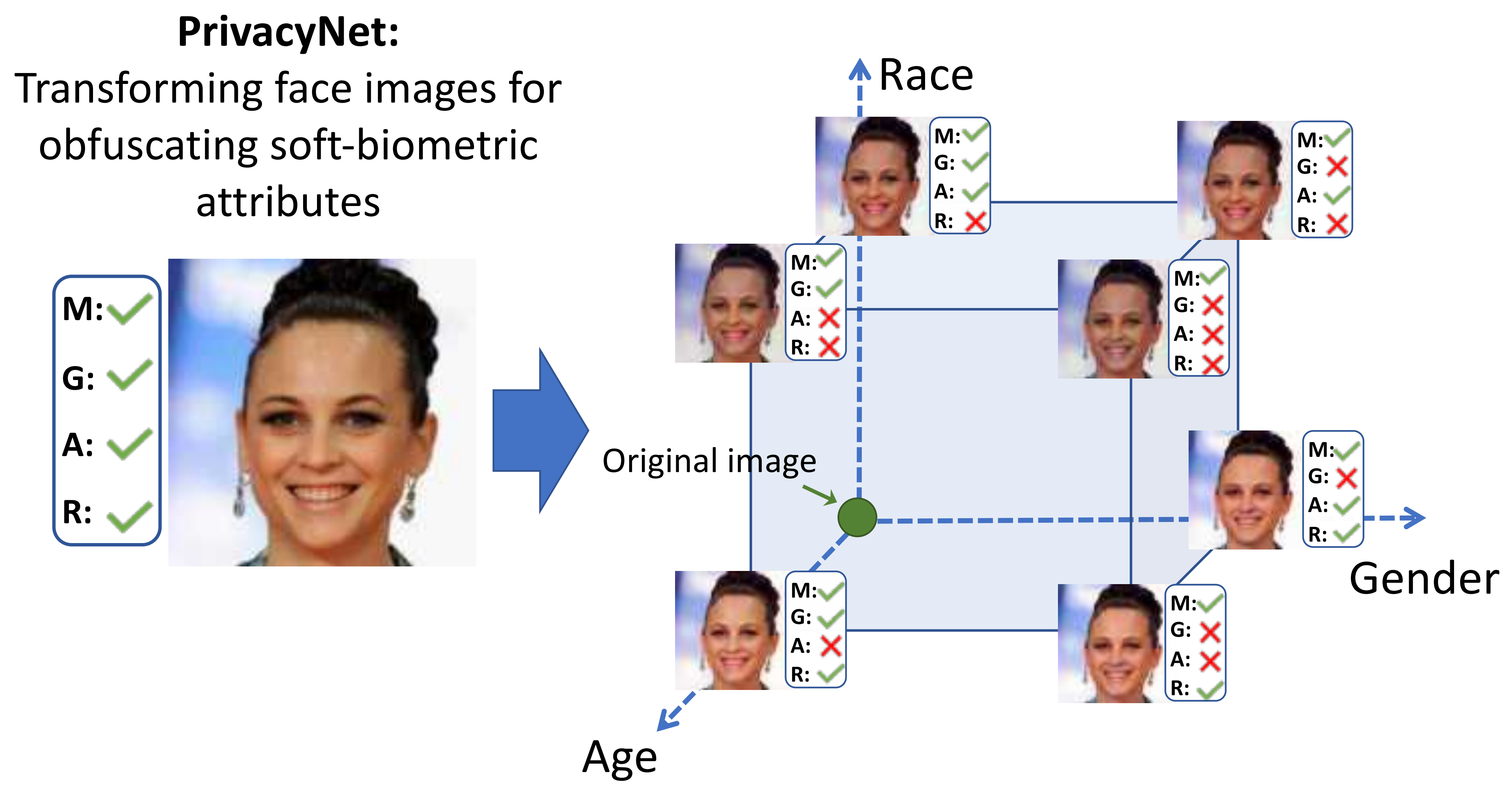
Figure 1. A mixture of real images (green border) and images with synthetic artifacts (red border). Is a good object representation necessary to tell them apart?

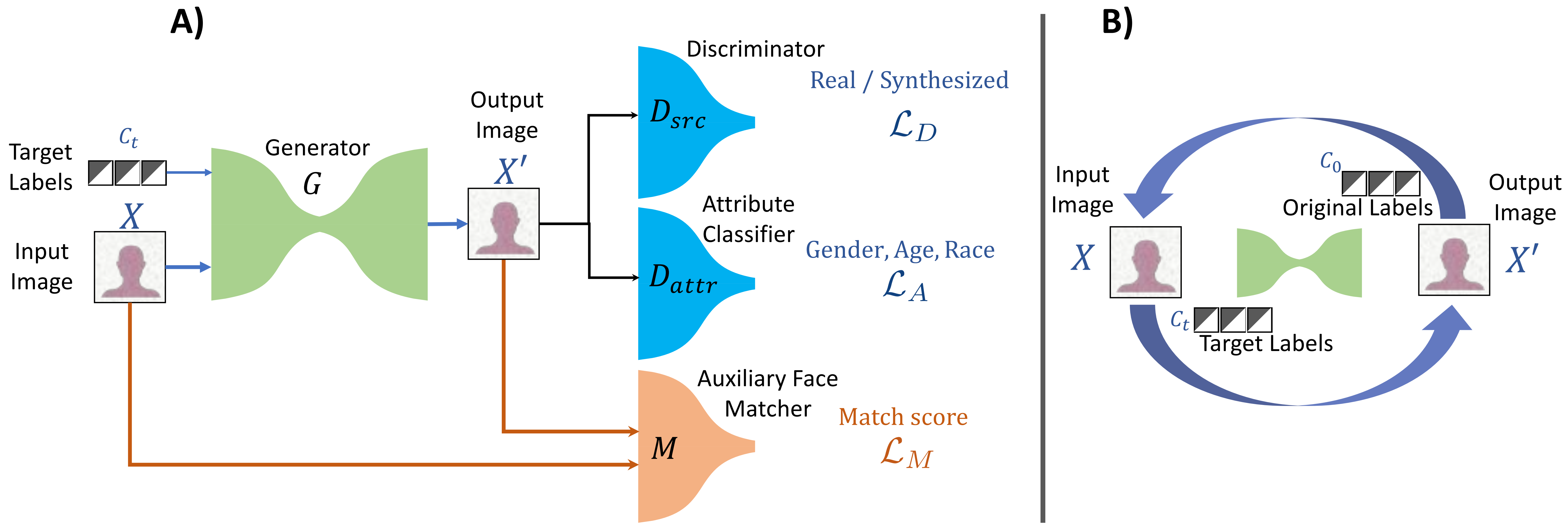
APRIL 14, 2020

Half of Americans have decided not to use a product or service because of privacy concerns

https://www.pewresearch.org/fact-tank/2020/04/14/half-of-americans-have-decided-not-to-use-a-product-or-service-because-of-privacy-concerns/

~2018-present: Increasing focus on user privacy





Vahid Mirjalili, Sebastian Raschka, Arun Ross (2020)
PrivacyNet: Semi-Adversarial Networks for Multi-attribute Face Privacy
arXiv:2001.00561

AI / DL and the "Small Data" Bottleneck

Transfer learning (pre-train on related dataset)

Self-supervised learning (like transfer learning but pre-train on related task)

Synthetic data (GAN, autoencoder)

One- and few-shot learning (train on many tasks with few examples per class)

Human-in-the-loop