Understanding Text on Images with AI at Scale

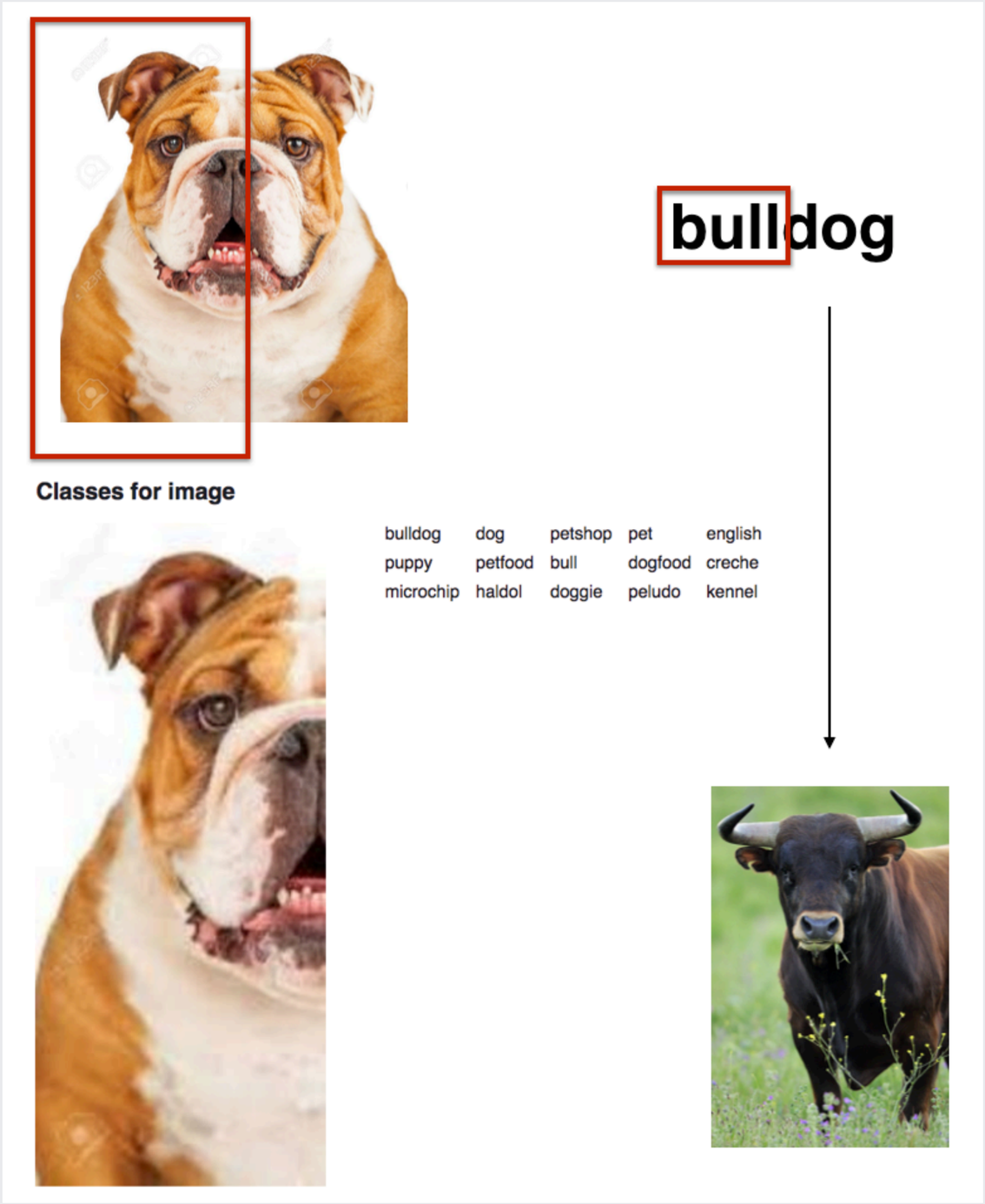
Viswanath Sivakumar

Facebook AI Research (FAIR)

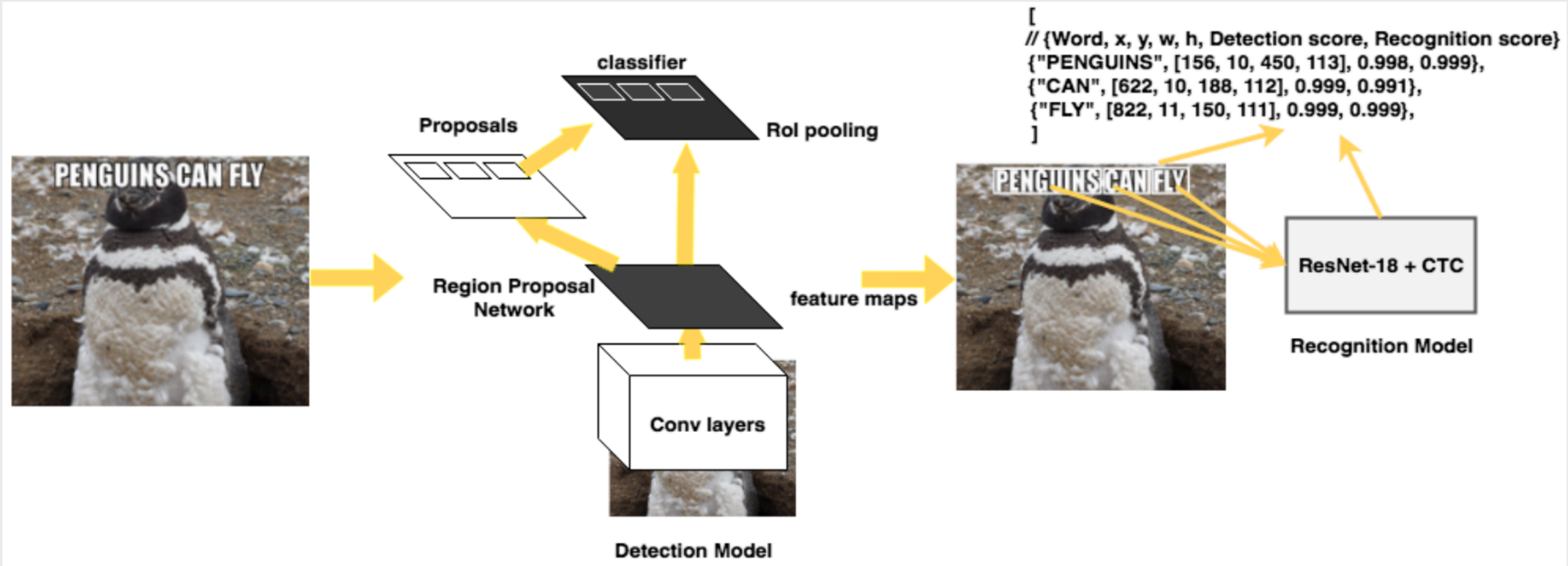

PINGUINO FELIZ TE ESPERA
PINGUINO FELIZ TE ESPERA
EN MAGDALENA EN 2018
EN MAGDALENA EN 2018
VIENES?
VIENES?

Challenges

- Sizes, fonts, orientations
- Languages
- Scale
- Efficiency

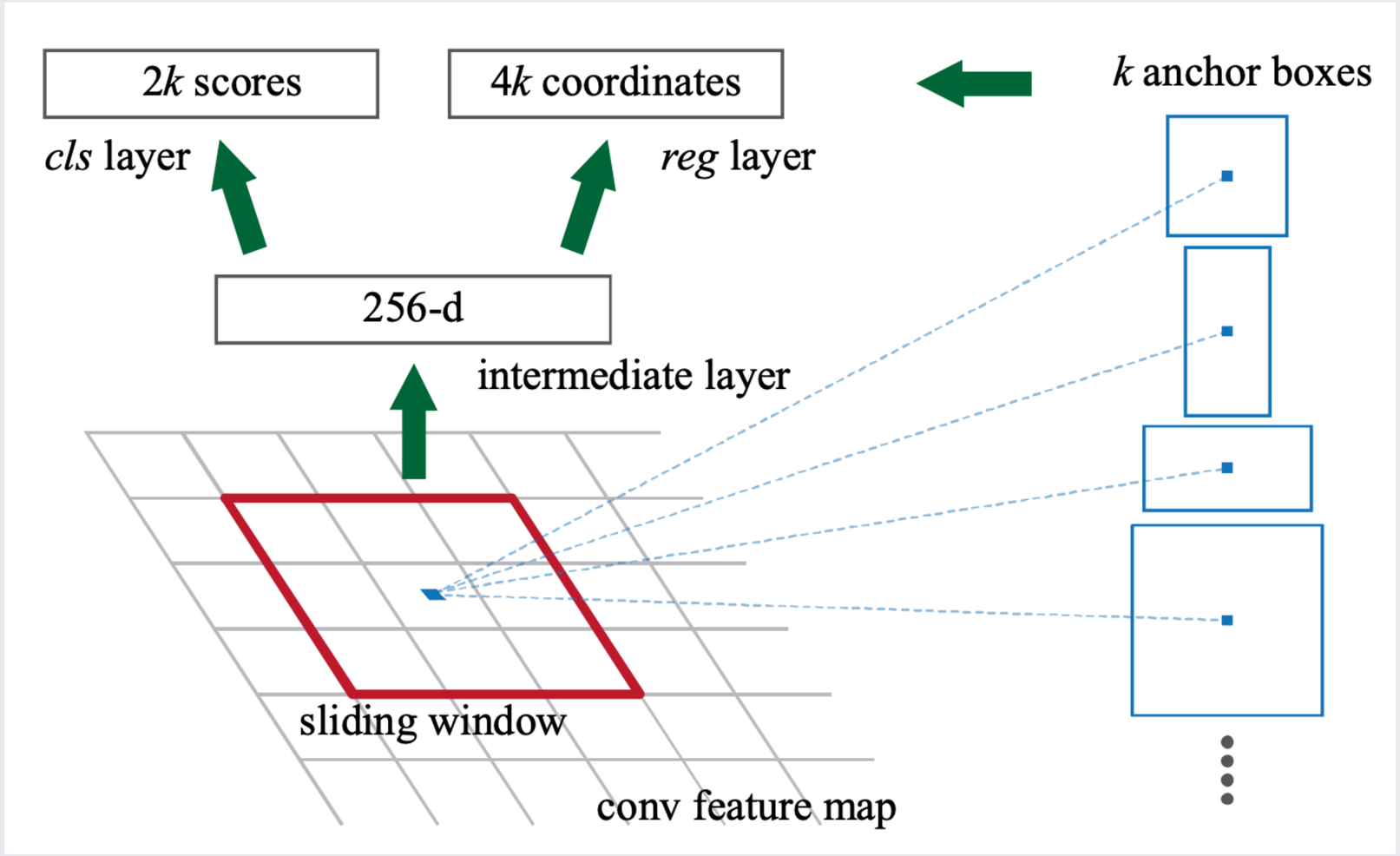


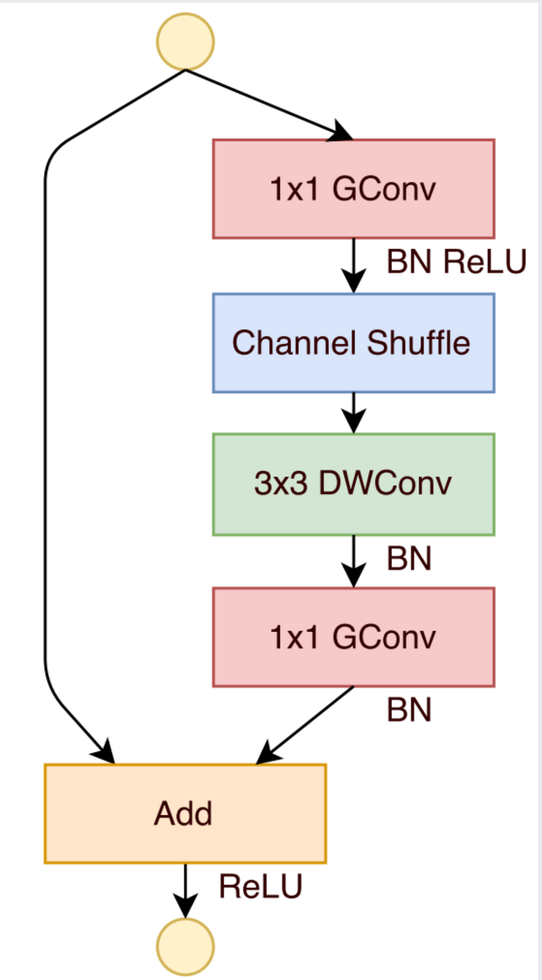
Architecture



Large scale system for text detection and recognition in images, KDD 2018, Viswanath Sivakumar, Albert Gordo, Fedor Borisyuk

Text Detection

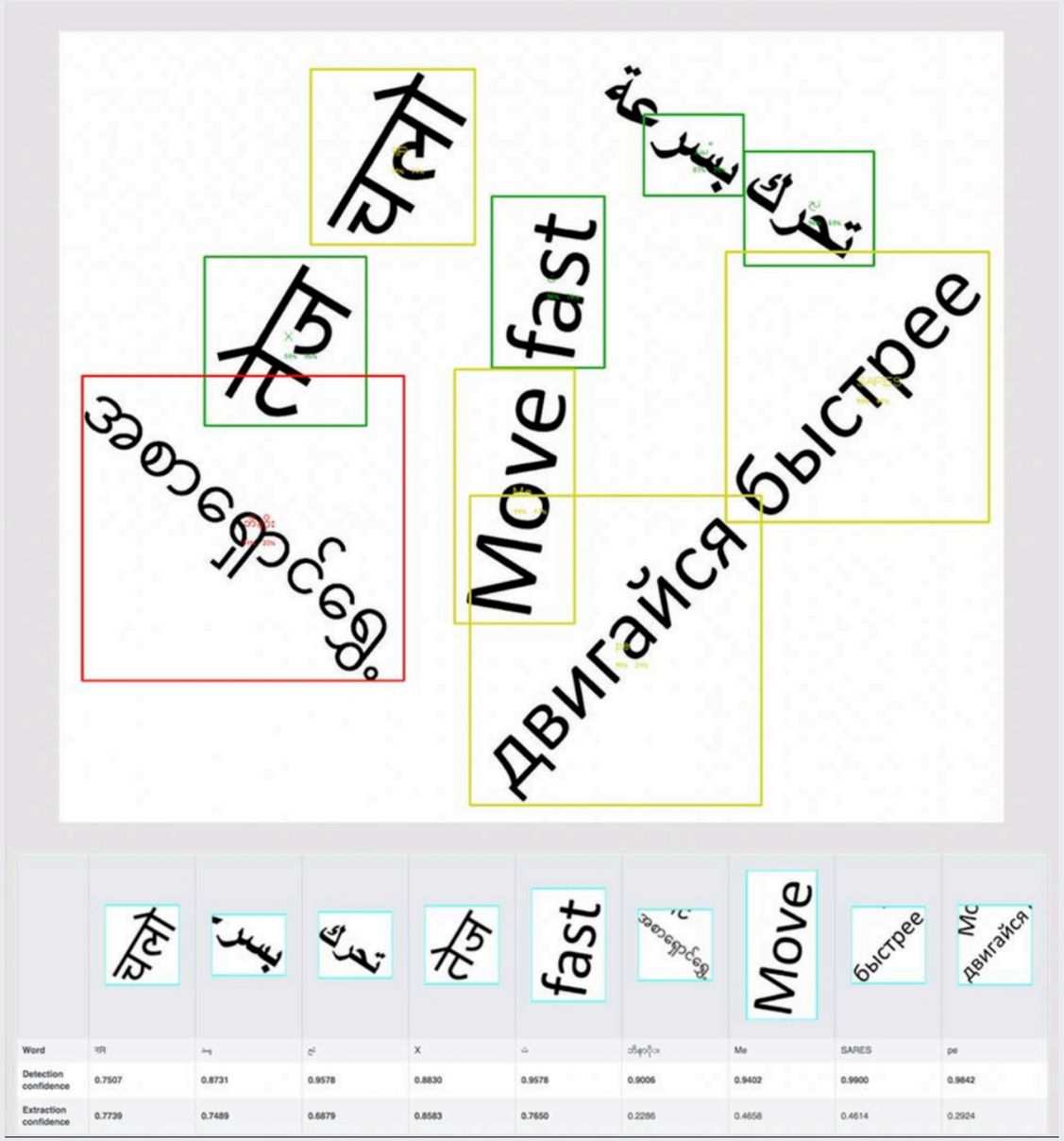




Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Ren et al.

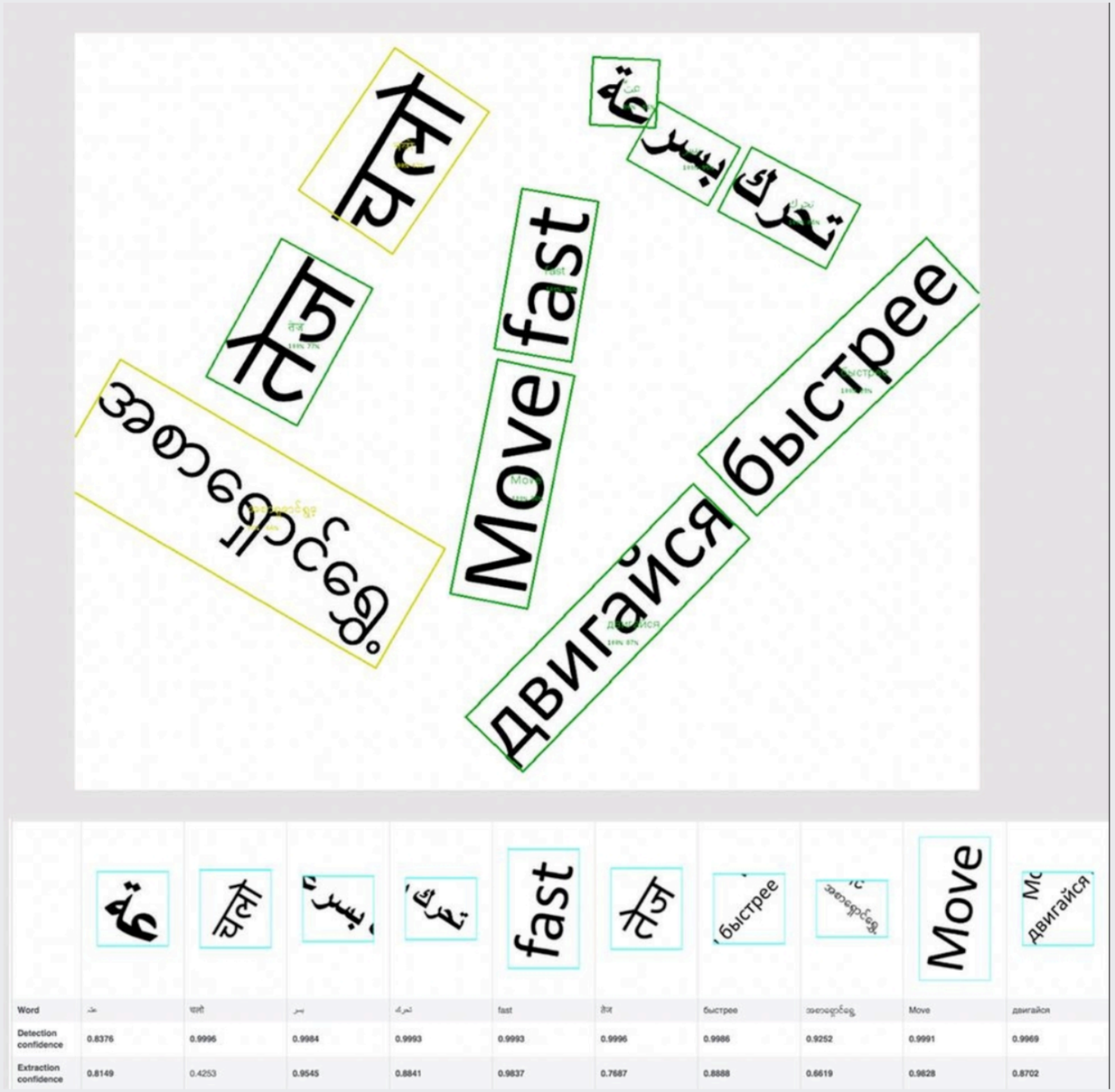
ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices, Zhang et al.

Orientations



Word	चलो	بسرعة	تحرك	तेज	fast	ဘေးရောင်းရွှေ့	Move	быстрее	двигайся
Detection confidence	0.7507	0.8731	0.9578	0.8830	0.9578	0.9006	0.9402	0.9900	0.9842
Extraction confidence	0.7739	0.7489	0.6879	0.8583	0.7650	0.2286	0.4658	0.4614	0.2924

Orientations



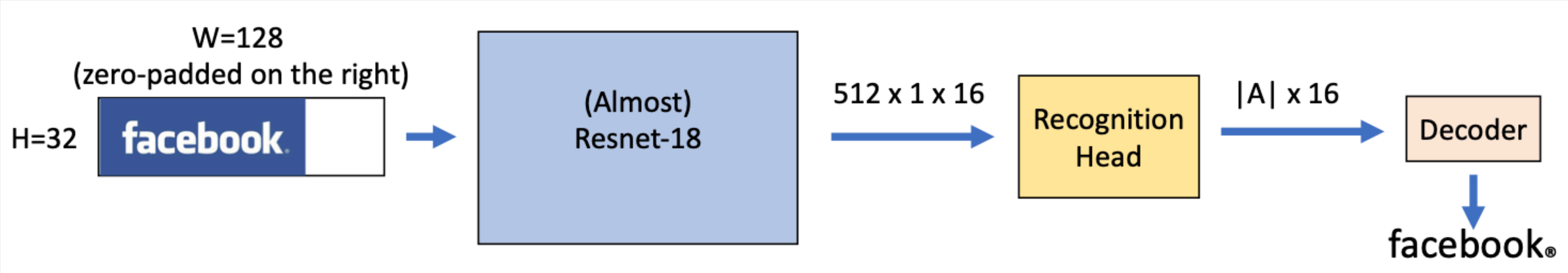
Word	عة	चला	بسر	تحرك	fast	तेज़	быстрее	အမြန်ရွှေ့	Move	двигайся
Detection confidence	0.8376	0.9996	0.9984	0.9993	0.9993	0.9996	0.9986	0.9252	0.9991	0.9969
Extraction confidence	0.8149	0.4253	0.9545	0.8841	0.9837	0.7687	0.8888	0.6619	0.9828	0.8702

Improving Rotated Text Detection with Rotation Region Proposal Networks, Huang et al.

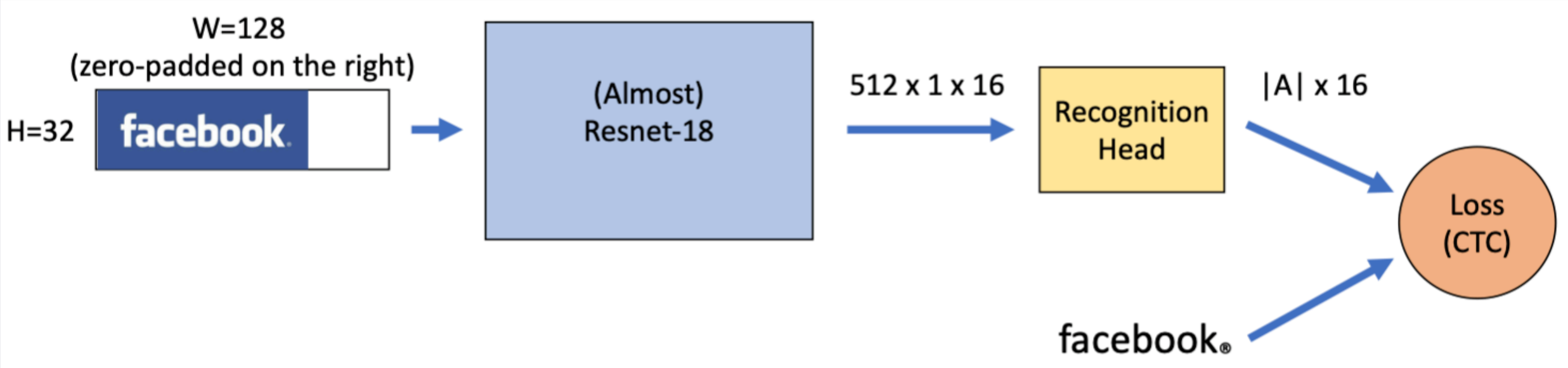
Text Recognition

Text Recognition

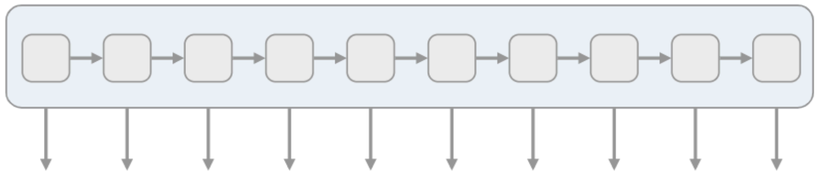
Text Recognition



Text Recognition



Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks, Graves et al.



The input is fed into an RNN, for example.

h	h	h	h	h	h	h	h	h	h
e	e	e	e	e	e	e	e	e	e
l	l	l	l	l	l	l	l	l	l
o	o	o	o	o	o	o	o	o	o
ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ

The network gives $p_t(a \mid X)$, a distribution over the outputs {h, e, l, o, ϵ} for each input step.

h	e	ϵ	l	l	ϵ	l	l	o	o
h	h	e	l	l	ϵ	ϵ	l	ϵ	o
ϵ	e	ϵ	l	l	ϵ	ϵ	l	o	o

With the per time-step output distribution, we compute the probability of different sequences

$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

The CTC conditional **probability**

marginalizes over the set of valid alignments

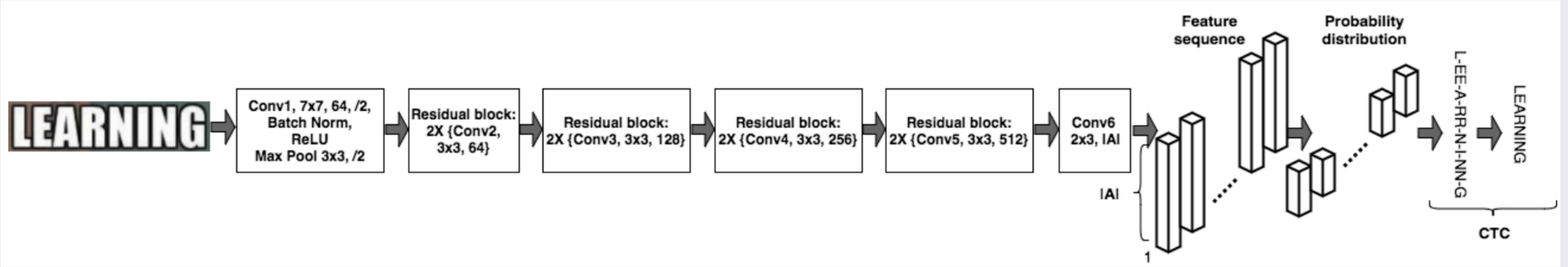
computing the **probability** for a single alignment step-by-step.

h	e	l	l	o
e	l	l	o	
h	e	l	o	

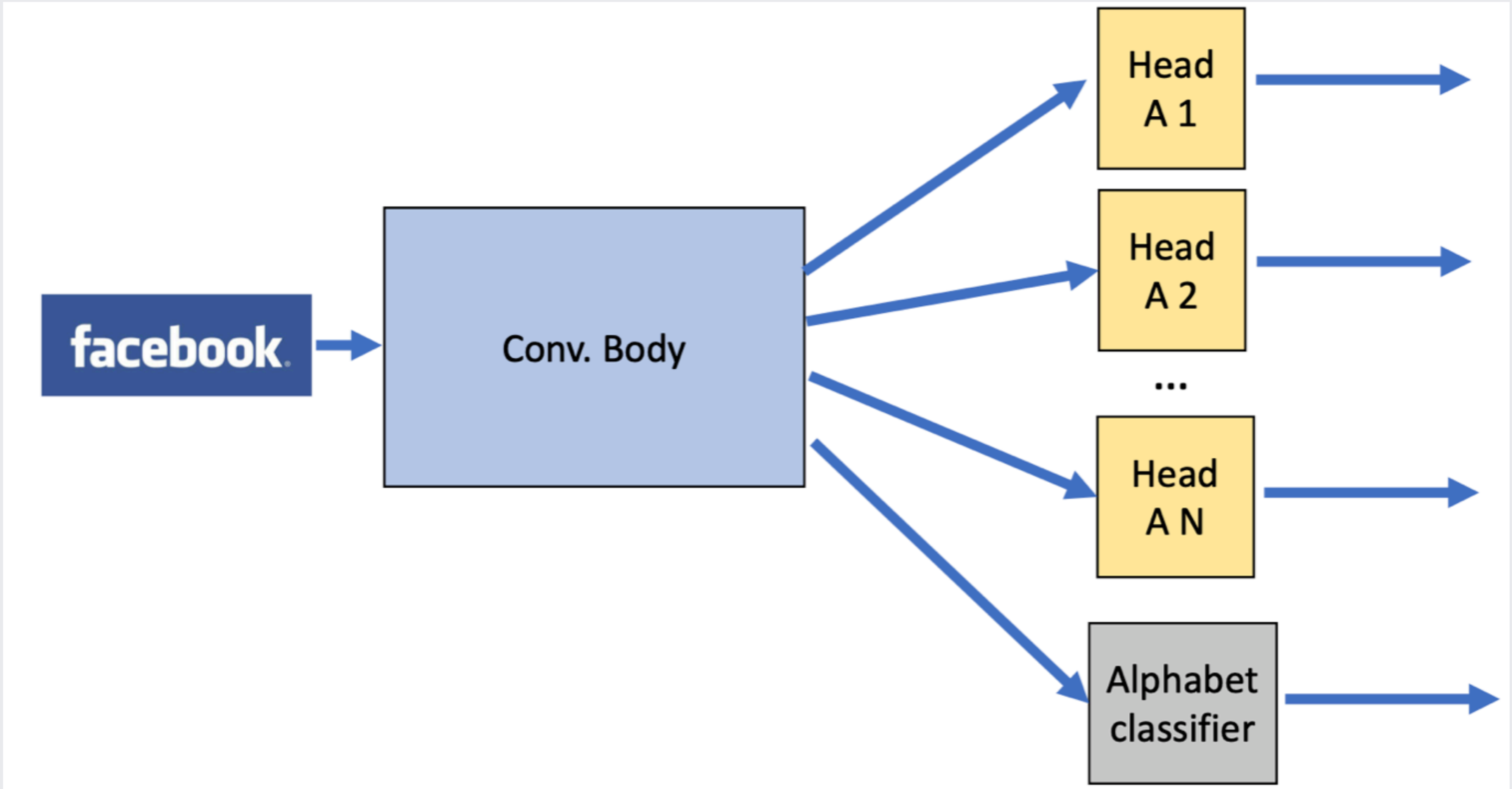
By marginalizing over alignments, we get a distribution over outputs.

Sequence Modeling With CTC, https://distill.pub/2017/ctc/

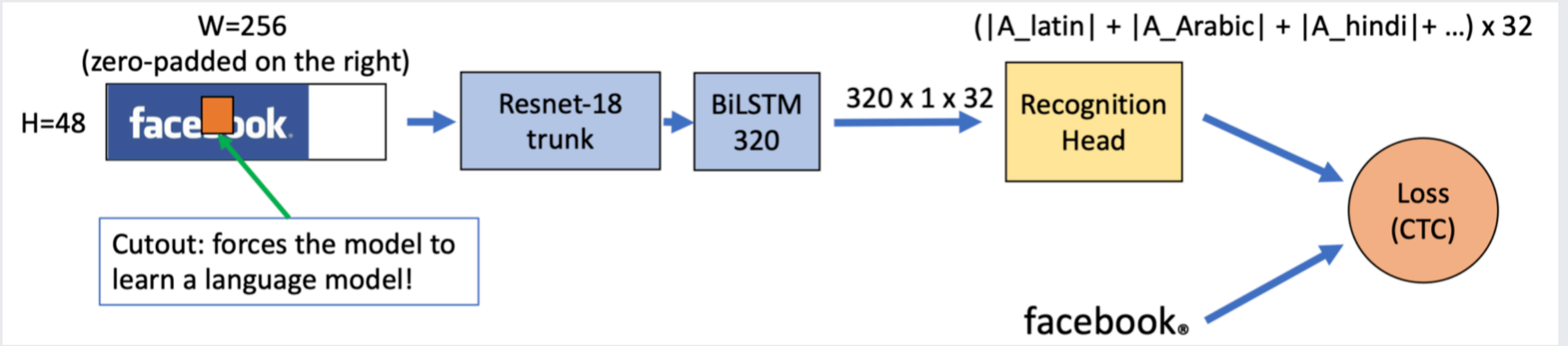
Text Recognition



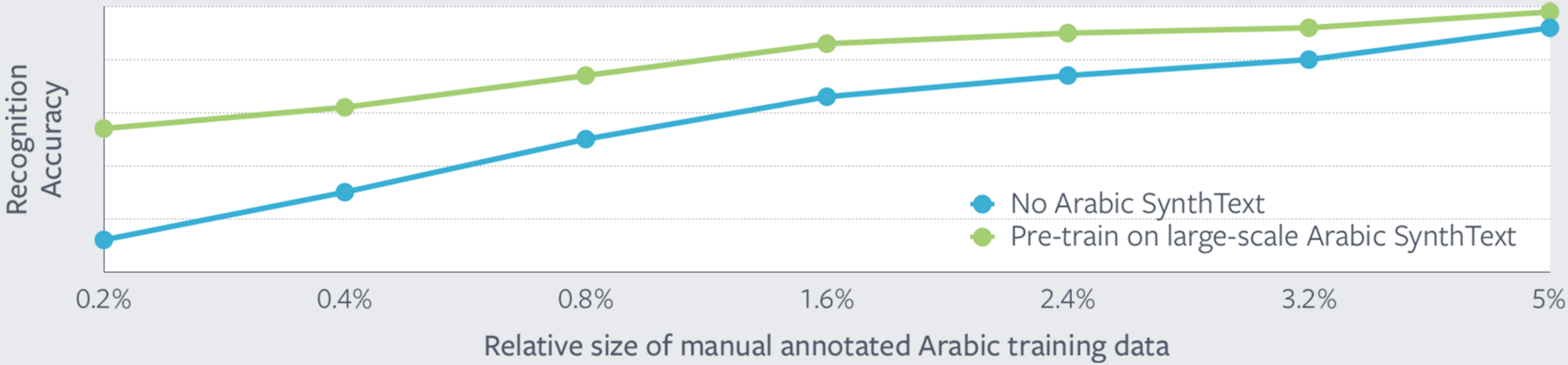
Multilingual



Multilingual



Synthetic Data



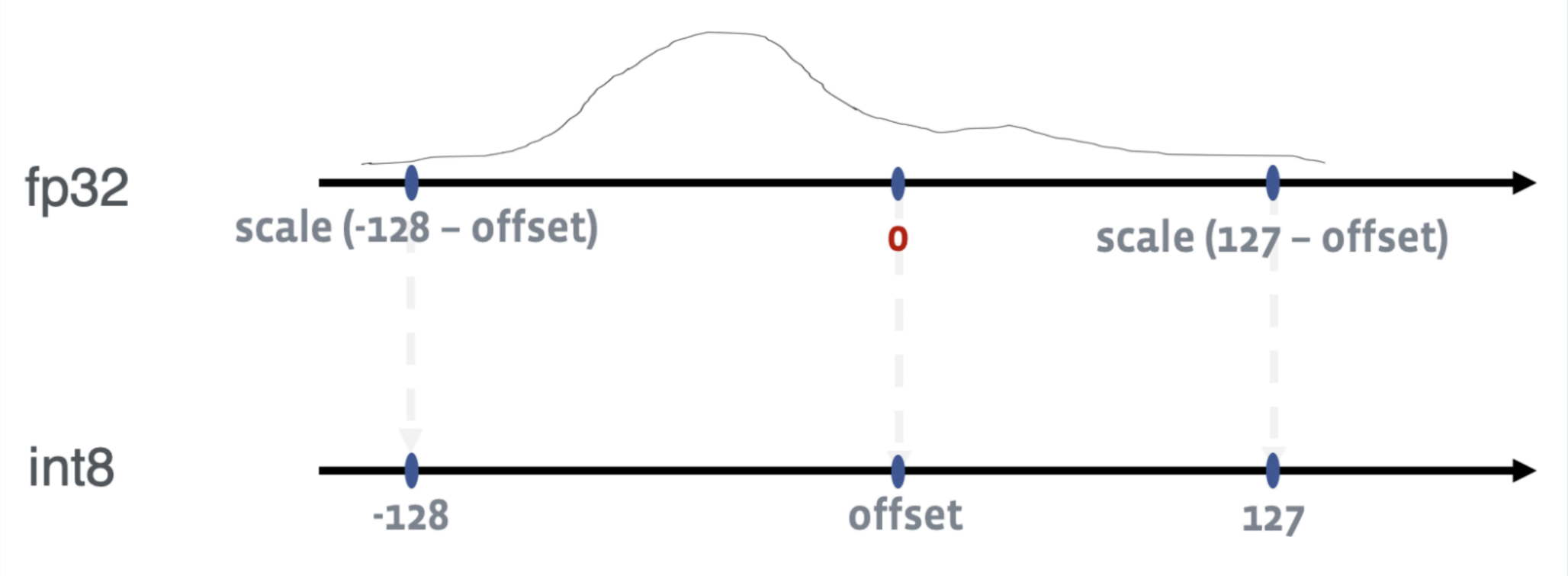
Inference

1B images/day x 5 sec/image = Lots of servers!

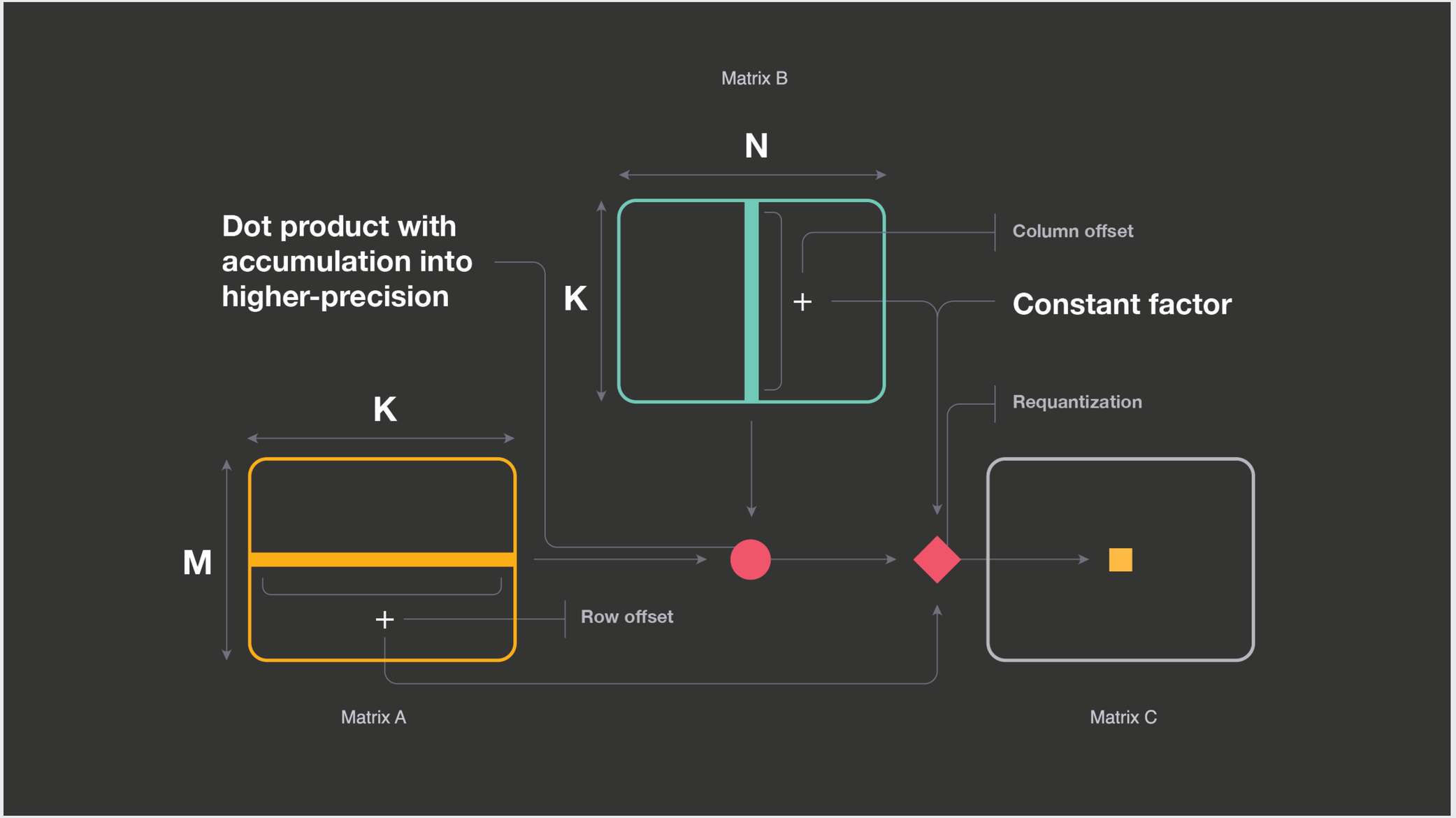
Int8 Quantization

Quantization: $x_q = clip(round(x/scale) + offset, -128,127)$

De-quantization: $x = scale \times (x_q - offset)$



Int8 Quantization



But what about accuracy?

- No quantization error for 0
- Fuse Convolution and ReLU
- L2 Error Minimization vs Min-Max
- Don't quantize the first layer

Initial accuracy gap: 5%

After: 0.2%

Deep Learning Inference in Facebook Data Centers: Characterization, Performance Optimizations and Hardware Implications, Park et al.

Int8 Quantization

Great for memory bandwidth bound systems!

But what about CPU?

Intel Architecture	fp32 GEMM with 32-bit accumulation	int8 GEMM with 32-bit accumulation	int8 GEMM with 16-bit accumulation
Broadwell	**32 ops/cycle**	**32 ops/cycle**	**64 ops/cycle**
Skylake	**64 ops/cycle**	**64 ops/cycle**	**192 ops/cycle**

16-bit accumulation is faster!
But what about accuracy drop due to overflows?

Outlier-Aware Quantization

A block of unsigned int8 activation matrix X

0	10	0	10
30	200	10	250
40	0	0	0
0	20	10	30

x

A block of signed int8 weight matrix W^T

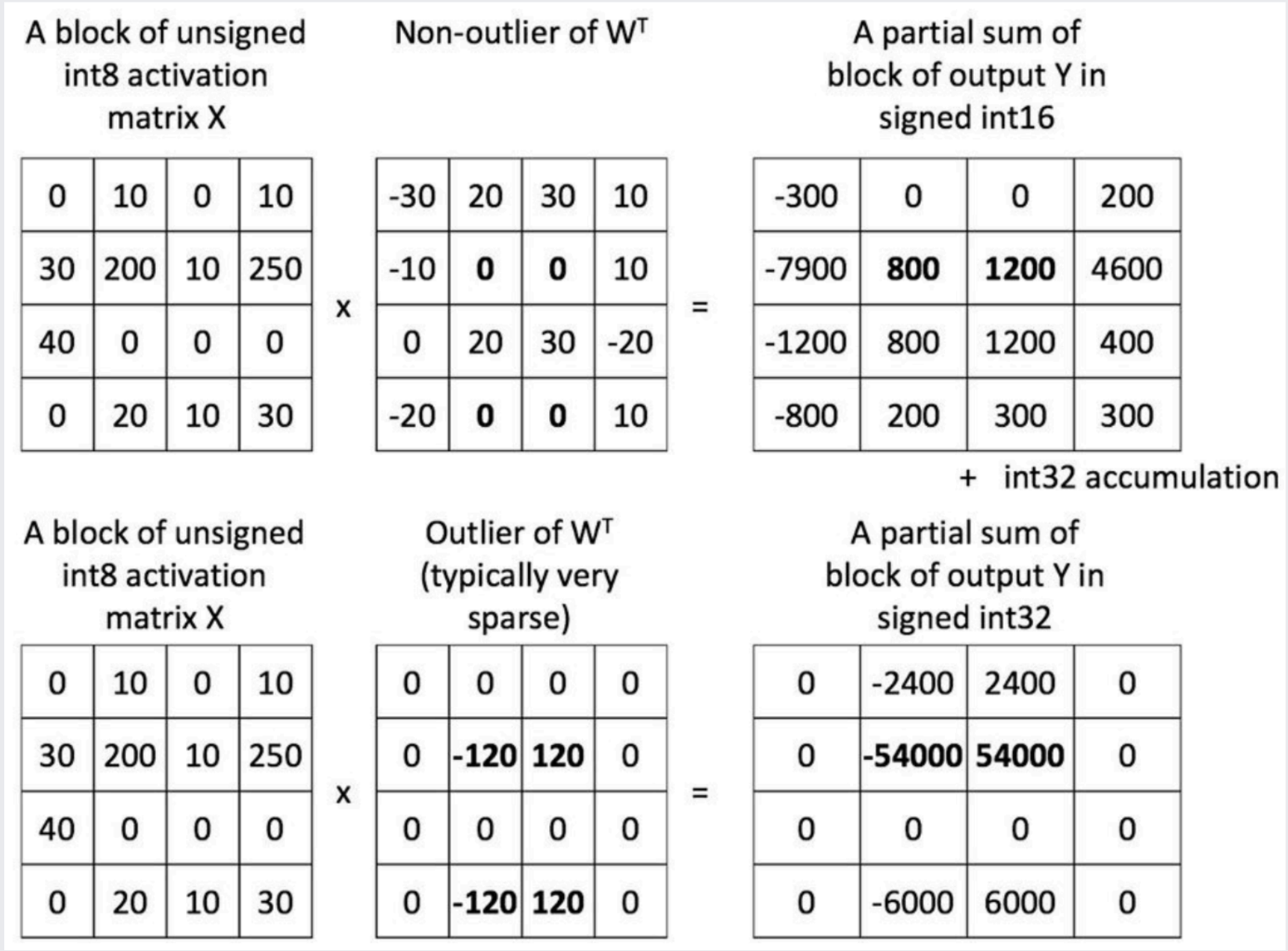
-30	20	30	10
-10	**-120**	**120**	10
0	20	30	-20
-20	**-120**	**120**	10

=

A partial sum of block of output Y in signed int16

-300	-2400	2400	200
-7900	**-53200**	**55200**	4600
-1200	800	1200	400
-800	-5800	6300	300

Outlier-Aware Quantization



W = Dense_W + Sparse_W

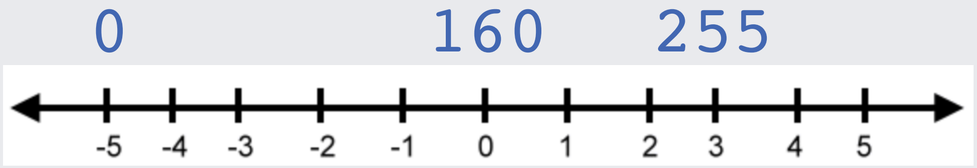
W: [-128, 127]
Dense_W: [-64, 63]
Sparse_W: Outliers

X * W =
X * Dense_W (16-bit acc)
+
X * Sparse_W (32-bit acc)

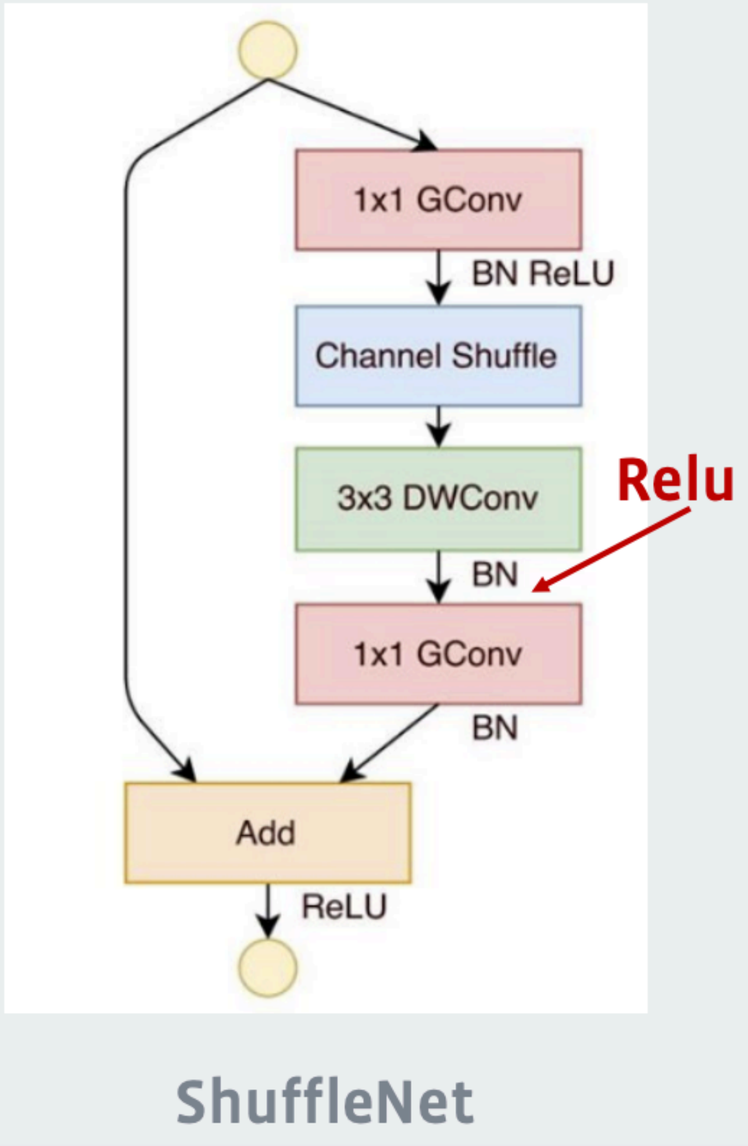
Model Co-Design

Outlier-aware quantization

- Int8 Quantization with 16-bit accumulation
- Further CPU speedup



Conv

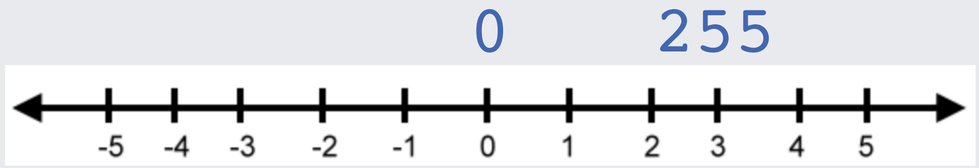


ShuffleNet

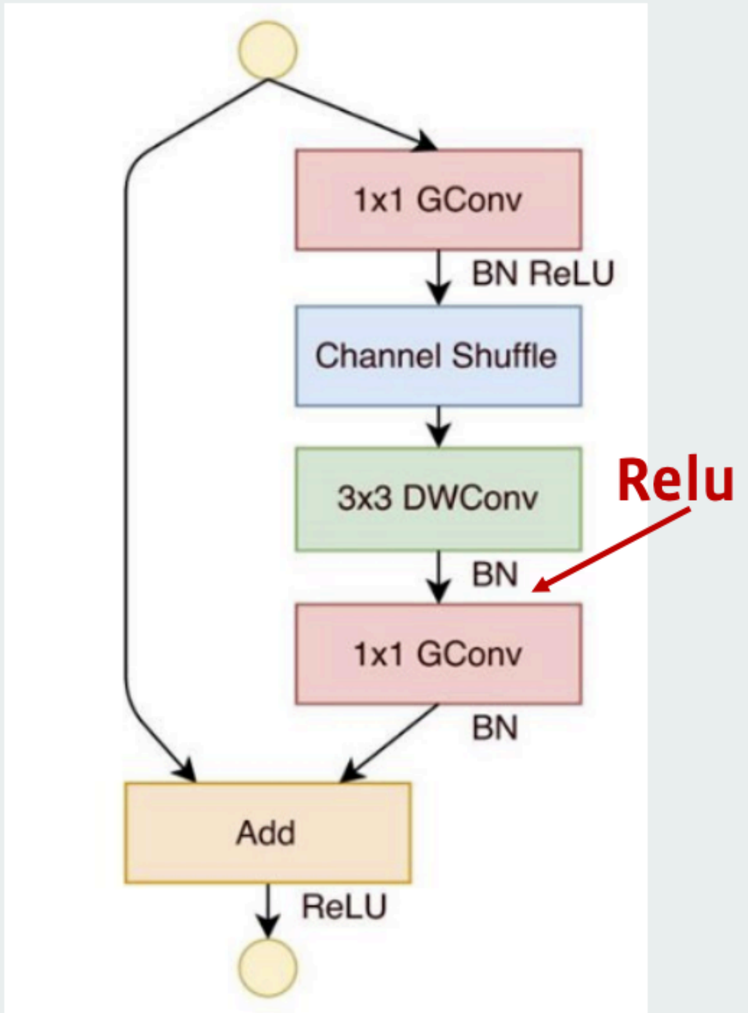
Model Co-Design

Outlier-aware quantization

- Int8 Quantization with 16-bit accumulation
- Further CPU speedup



Conv-ReLU

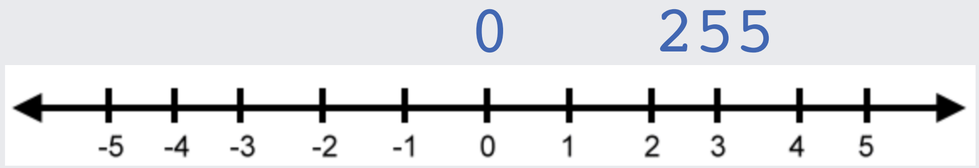


ShuffleNet

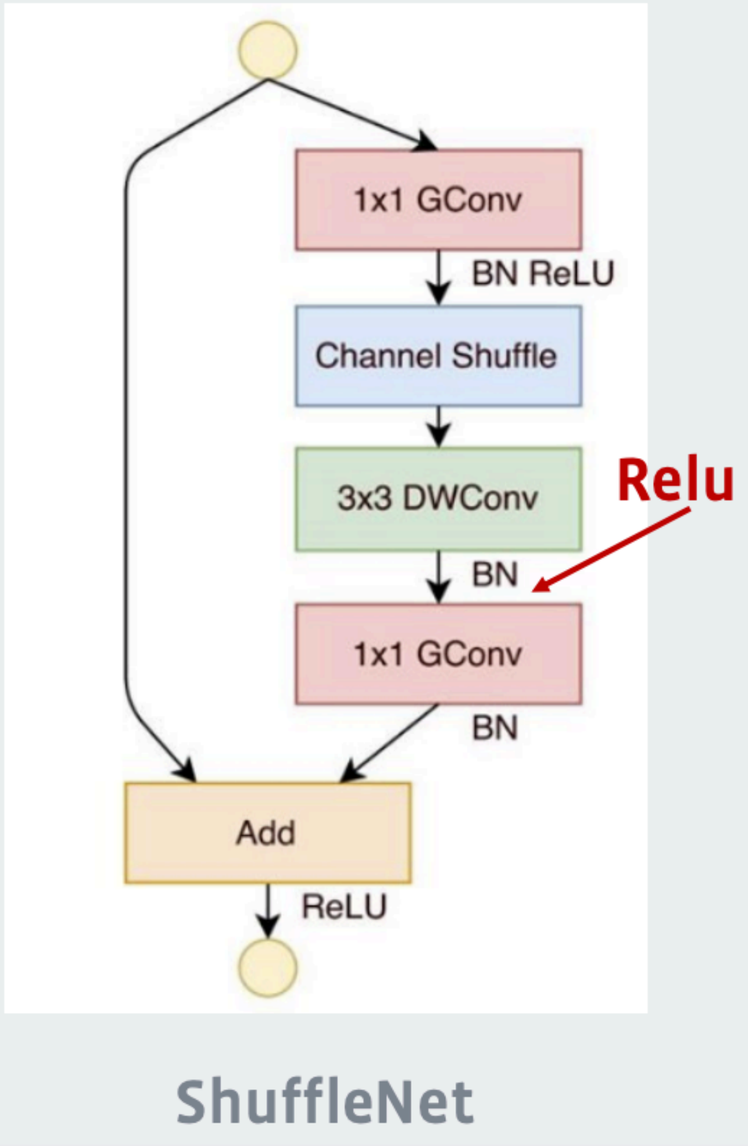
Model Co-Design

Outlier-aware quantization

- Int8 Quantization with 16-bit accumulation
- Further CPU speedup



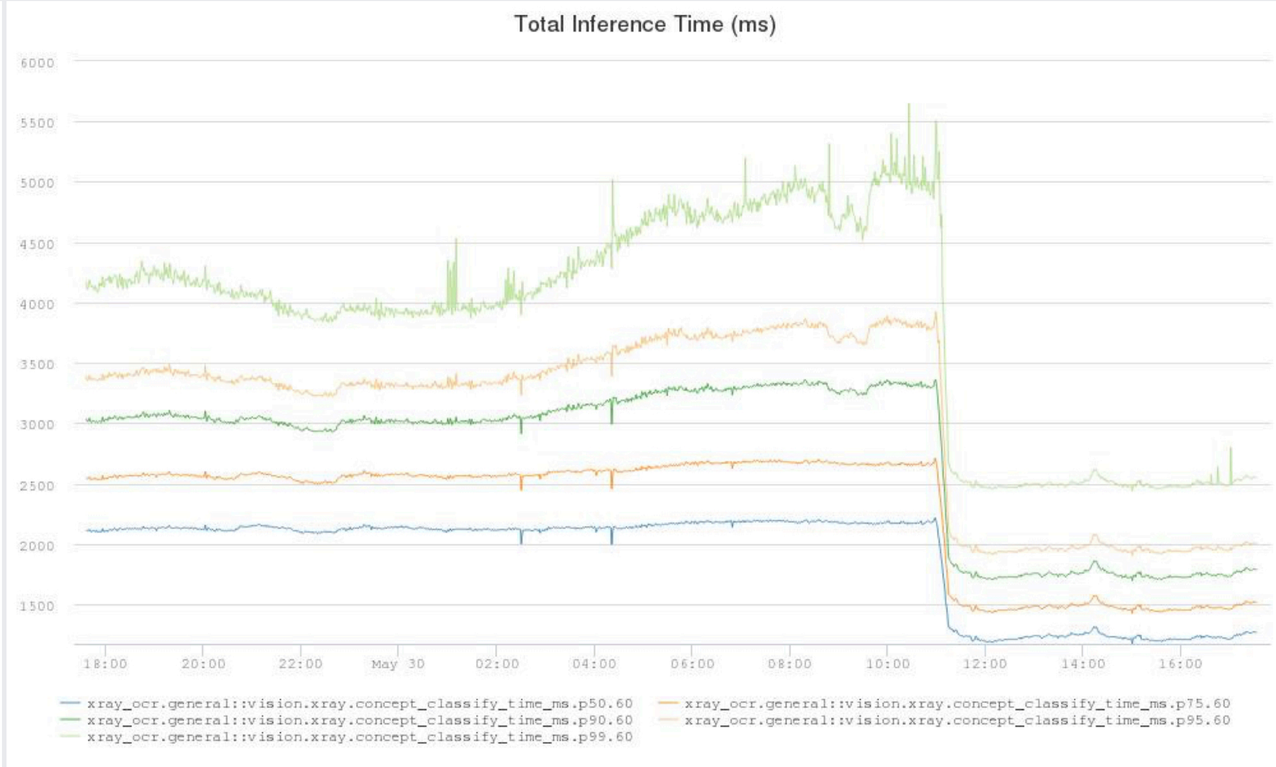
Conv-ReLU



ShuffleNet

Co-design your model and efficiency optimizations together!

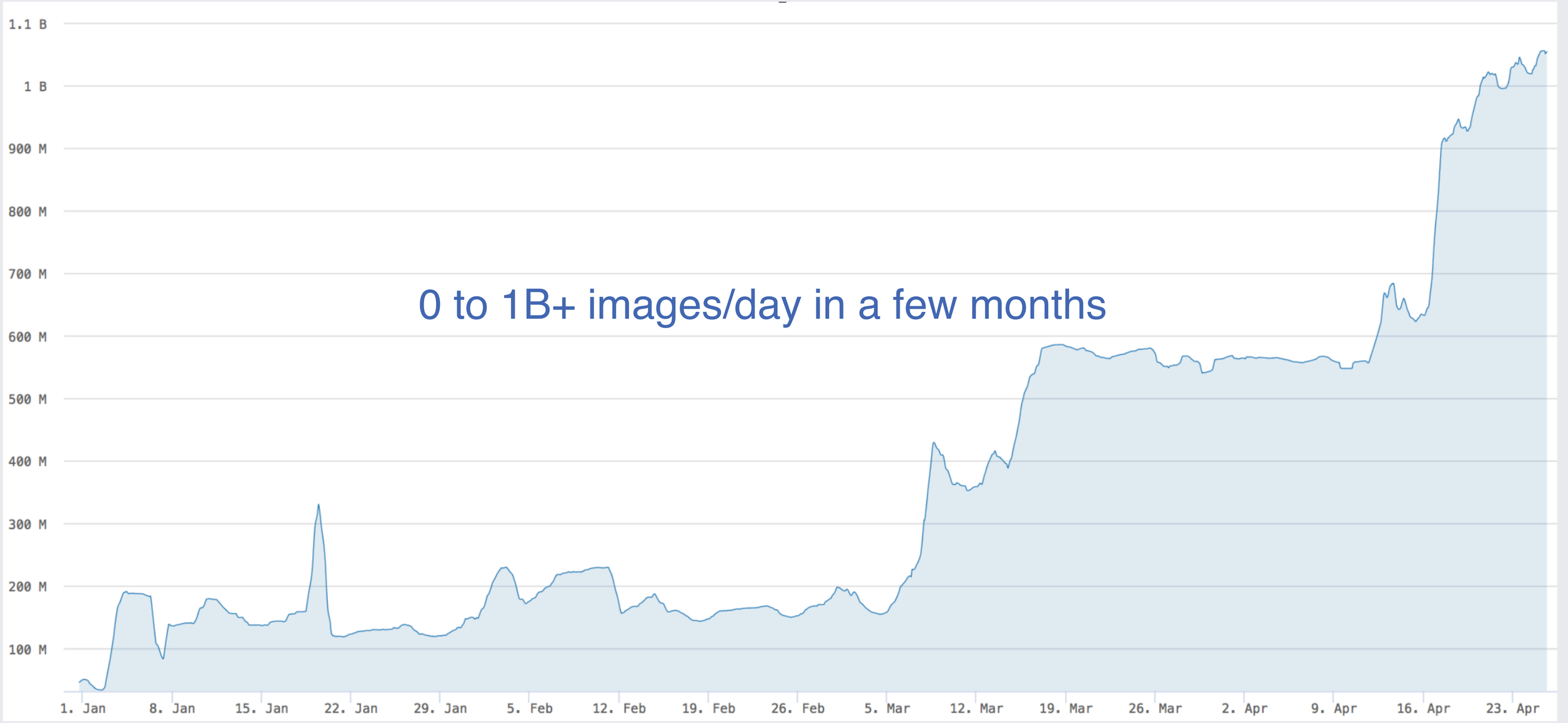
Int8 Quantization



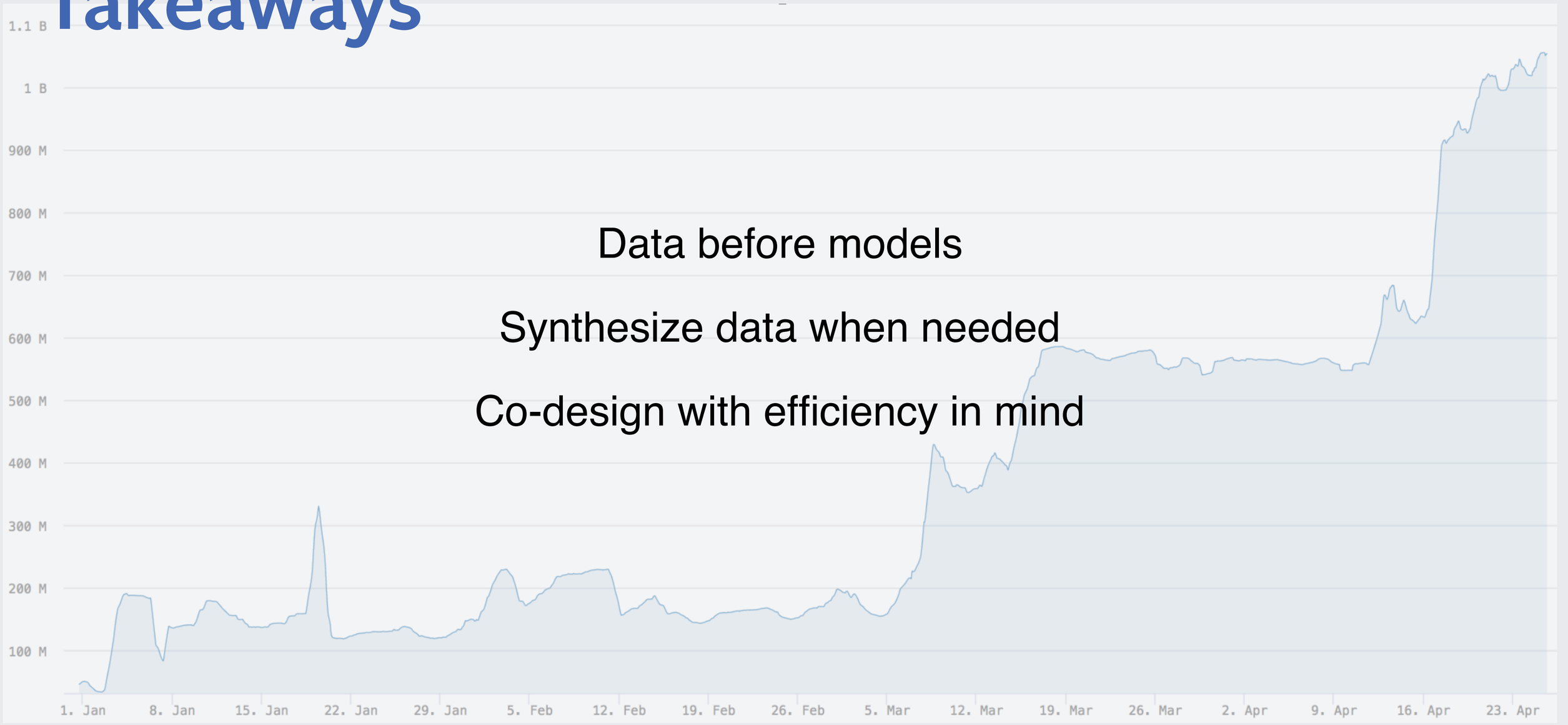
2.4x images/server

2x faster

github.com/pytorch/FBGEMM


1.1 B
1 B
900 M
800 M
700 M
600 M
500 M
400 M
300 M
200 M
100 M
0 to 1B+ images/day in a few months
1. Jan
8. Jan
15. Jan
22. Jan
29. Jan
5. Feb
12. Feb
19. Feb
26. Feb
5. Mar
12. Mar
19. Mar
26. Mar
2. Apr
9. Apr
16. Apr
23. Apr

Takeaways



Data before models

Synthesize data when needed

Co-design with efficiency in mind

Takeaways

Data before models

Synthesize data when needed

Co-design with efficiency in mind

Thank You!