Large Scale Scene-Text Extraction

Viswanath Sivakumar

Facebook AI Research (FAIR)

Agenda

1. **Introduction**
2. Text Detection
3. Text Recognition
4. Performance Optimizations
5. Additional Topics


NEUROSCIENCE AND AI:
INTEGRATING THE UNEXPLORED AREAS
www.jsbconference.com

Sentence	JEB NEUROSCIENCE AND AI: INTEGRATING THE UNEXPLORED AREAS www.jsbconference.com							
Languages	English	0.99999994039536						
Crop	Word	Detection confidence	Extraction confidence	TopLeftX	TopLeftY	Height	Width	Angle
	JEB	0.2686	**0.5602**	**531.6**	**10.9**	**14.1**	**18.2**	0
	NEUROSCIENCE	**0.9021**	**0.8566**	**63.5**	**216**	**34.5**	**286.5**	0
	AND	**0.9089**	**0.9915**	**361**	**216.8**	**32.8**	**86.9**	-0.4
	AI:	**0.8879**	**0.7287**	**456.9**	**217.2**	**33.7**	**54.1**	**0.7**
	INTEGRATING	**0.7018**	**0.9626**	**136**	**259.7**	**17.5**	**105.9**	0
	THE	**0.8127**	**0.9877**	**244.6**	**260.4**	**15.7**	**30**	0
	UNEXPLORED	**0.8127**	**0.9667**	**277.5**	**259.6**	**17.5**	**103.4**	0
	AREAS	**0.7018**	**0.9823**	**384.4**	**259**	**17.7**	**49.9**	0
	www.jsbconference.com	0.4041	0.478	**435.5**	**284.8**	**10.5**	**113.7**	-0.4

Motivated by the need to identify policy-violating text on images

Began work around July 2017

2+ billion images / day

100+ languages

Currently used by a wide variety of Facebook teams:

- Hate-speech detection
- Spam detection
- Image Search
- News Feed Ranking
- ...

OCR as a Computer Vision & Machine Learning Problem

Overall Task: Given an image, correctly **extract** the text in the image

From a ML standpoint, there are two primary subtasks:

Task 1: **Detect** the text. *Where is the text in this image?*

Task 2: **Recognize** the text. *What does the text say?*

Other Problems @ Facebook

Filtering: which images should we process with OCR?

Scale: efficiently process >1B images per day

Speed: inference times matter for Integrity

Languages: extract text in every language

A few major challenges

- ***The need for very precise object detection***
  (half of a bulldog still looks like a dog; half of "bulldog" looks "bull")

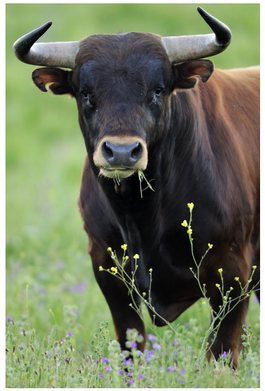
- ***Text of different…***
  - sizes
  - fonts
  - alphabets
  - orientations
  - scripts
  - styles (curved)

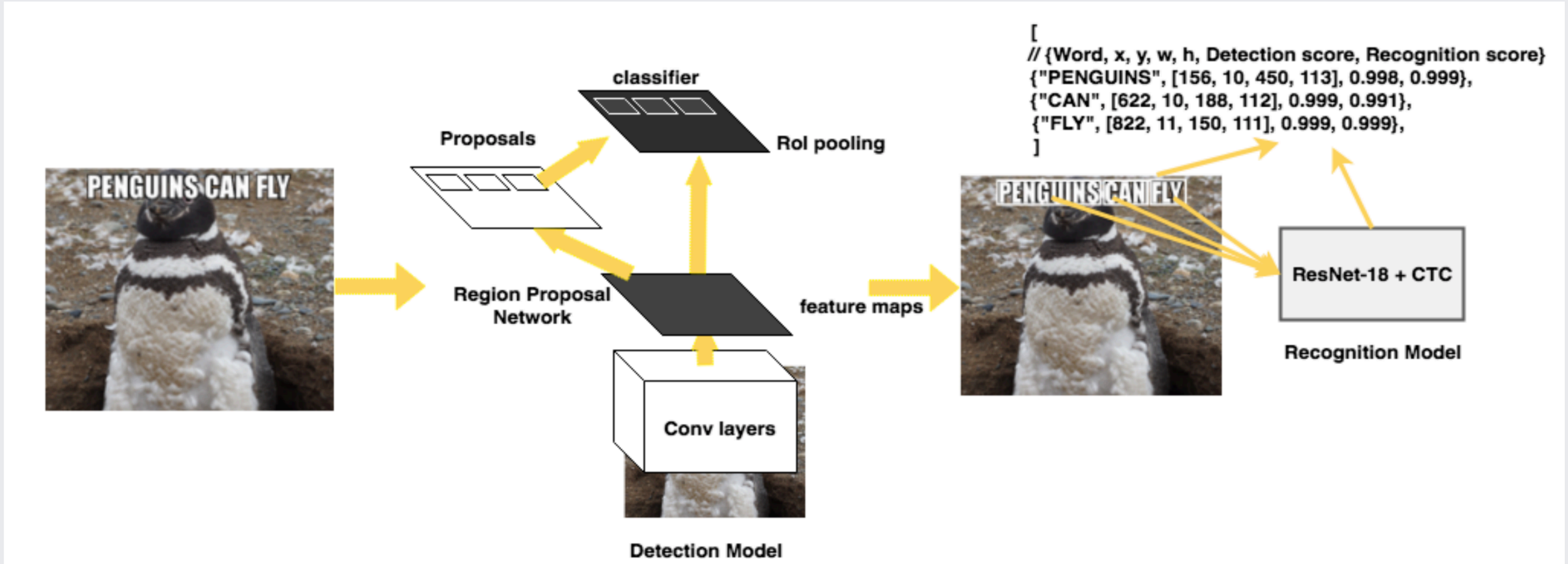
- ***Occluded (blocked/masked) & Obfuscated (unclear/unintelligible) Text***

bulldog

Classes for image

bulldog	dog	petshop	pet	english
puppy	petfood	bull	dogfood	creche
microchip	haldol	doggie	peludo	kennel

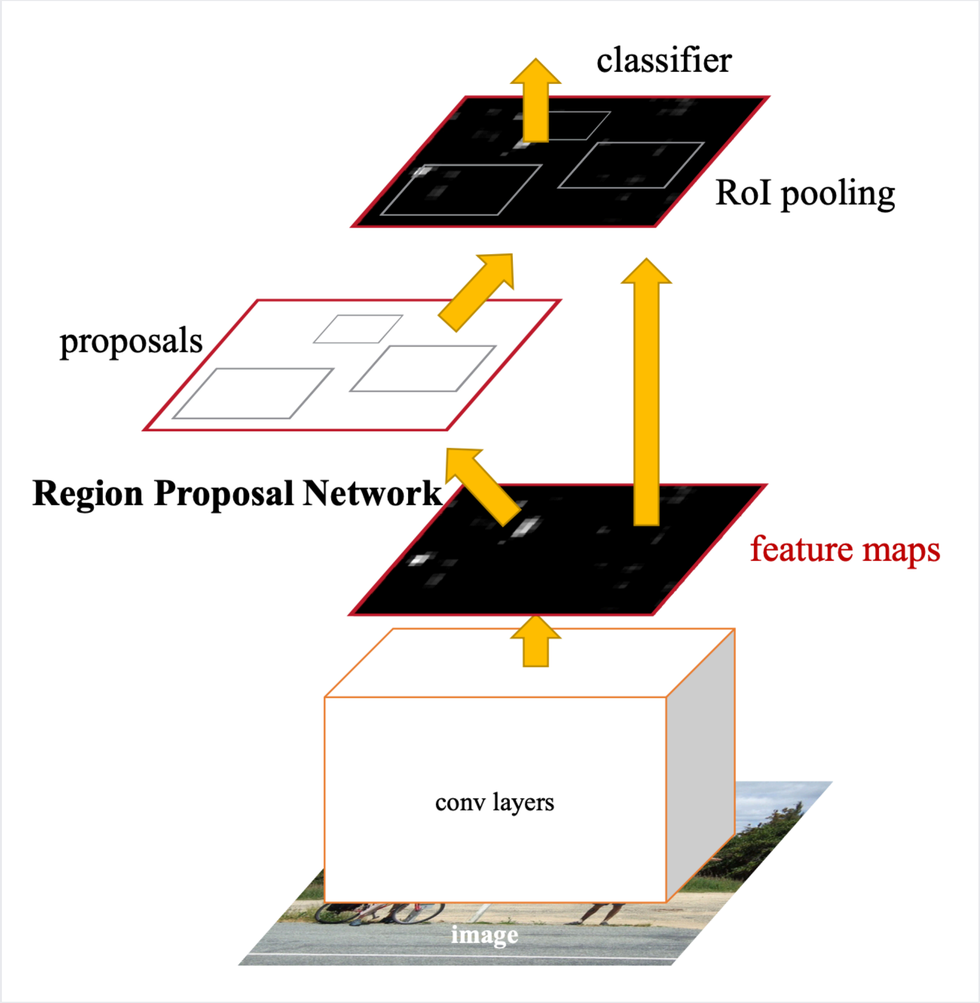


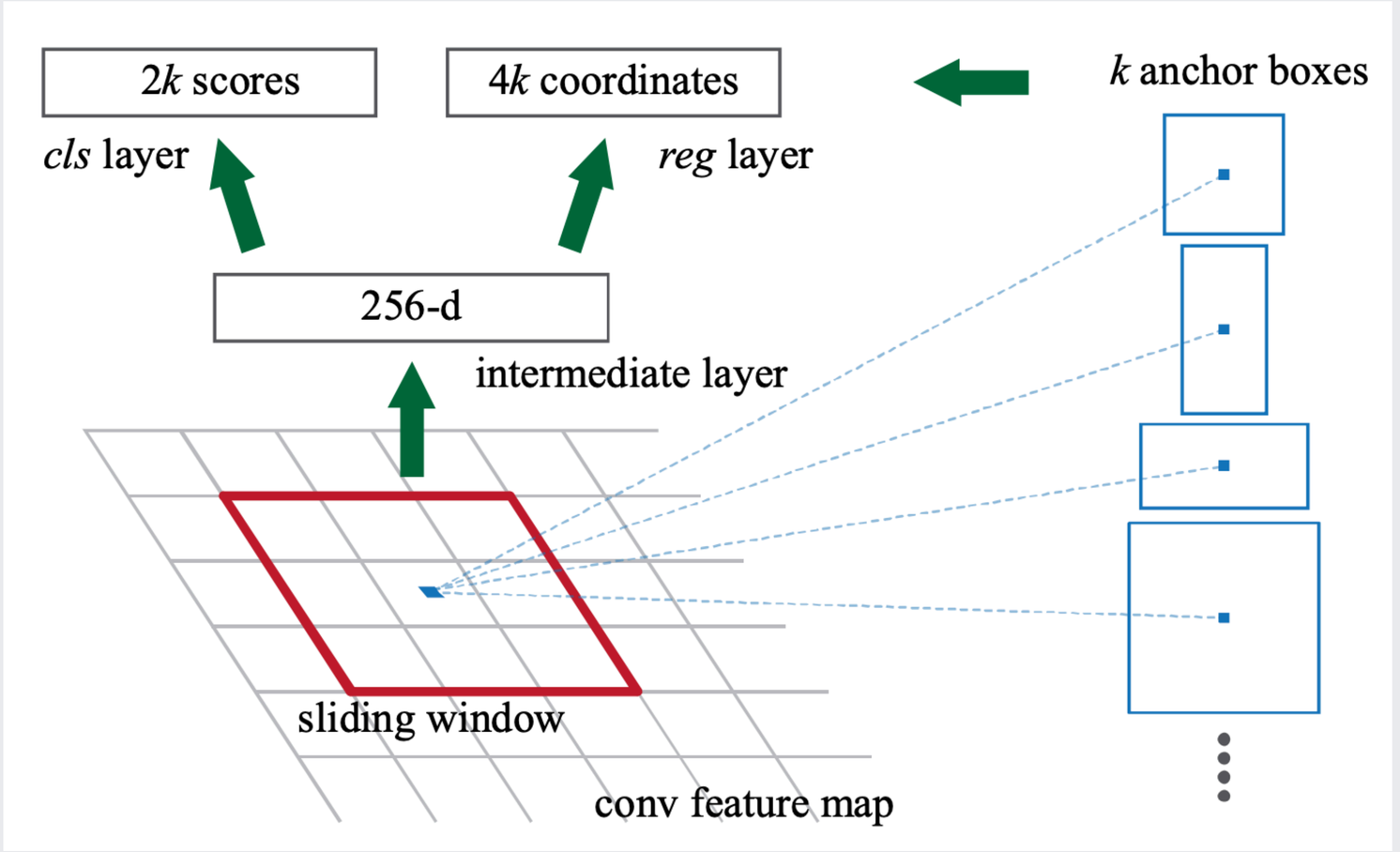


Large scale system for text detection and recognition in images, KDD 2018,
Viswanath Sivakumar, Albert Gordo, Fedor Borisyuk

Agenda

1. Introduction
2. **Text Detection**
3. Text Recognition
4. Performance Optimizations
5. Additional Topics

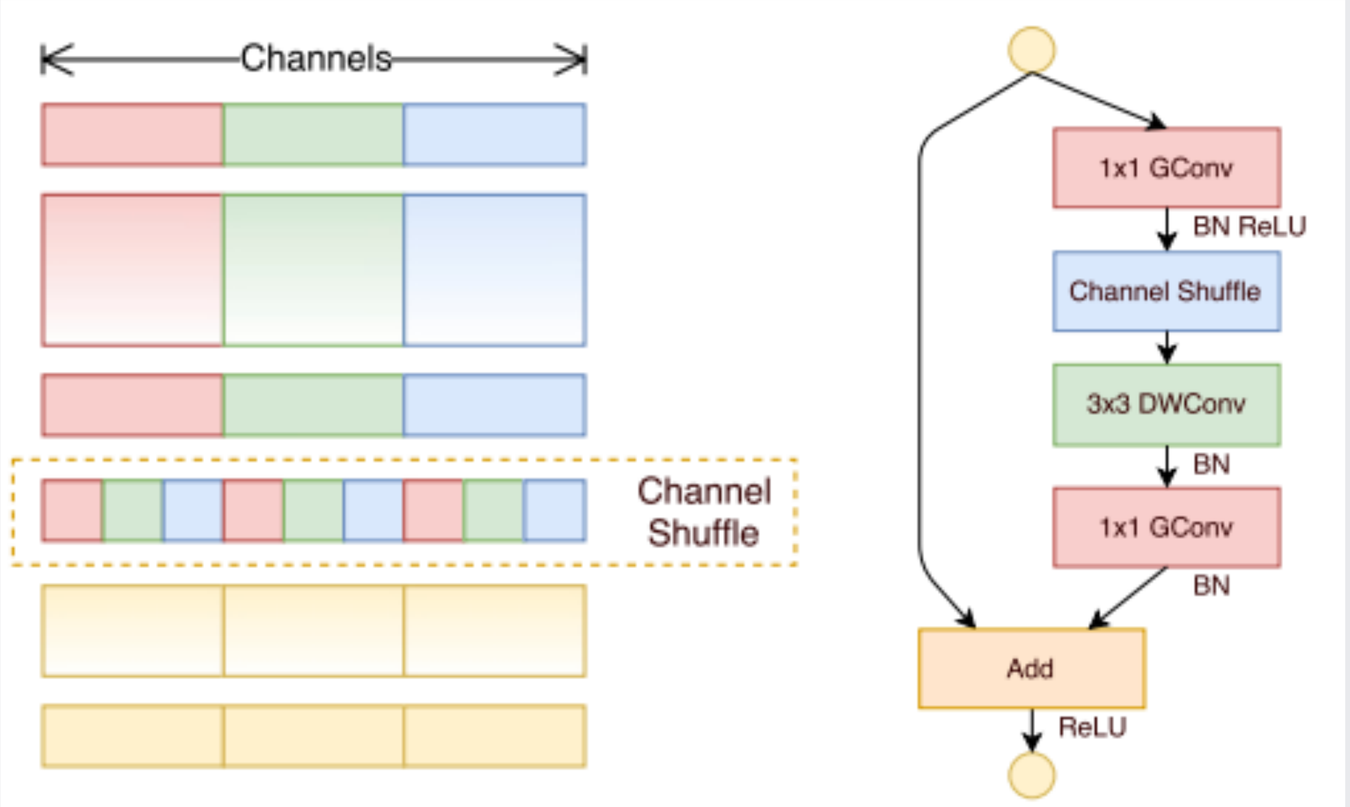




Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Ren et al.

Faster-RCNN modified

- ResNext -> MobileNet -> ShuffleNet
- 3x3 Conv -> Depthwise
- 1x1 Conv -> Group Conv + Channel Shuffle
- Channel Shuffle prevents local connectivity
- Additional tricks such as more aspect ratios for region proposal

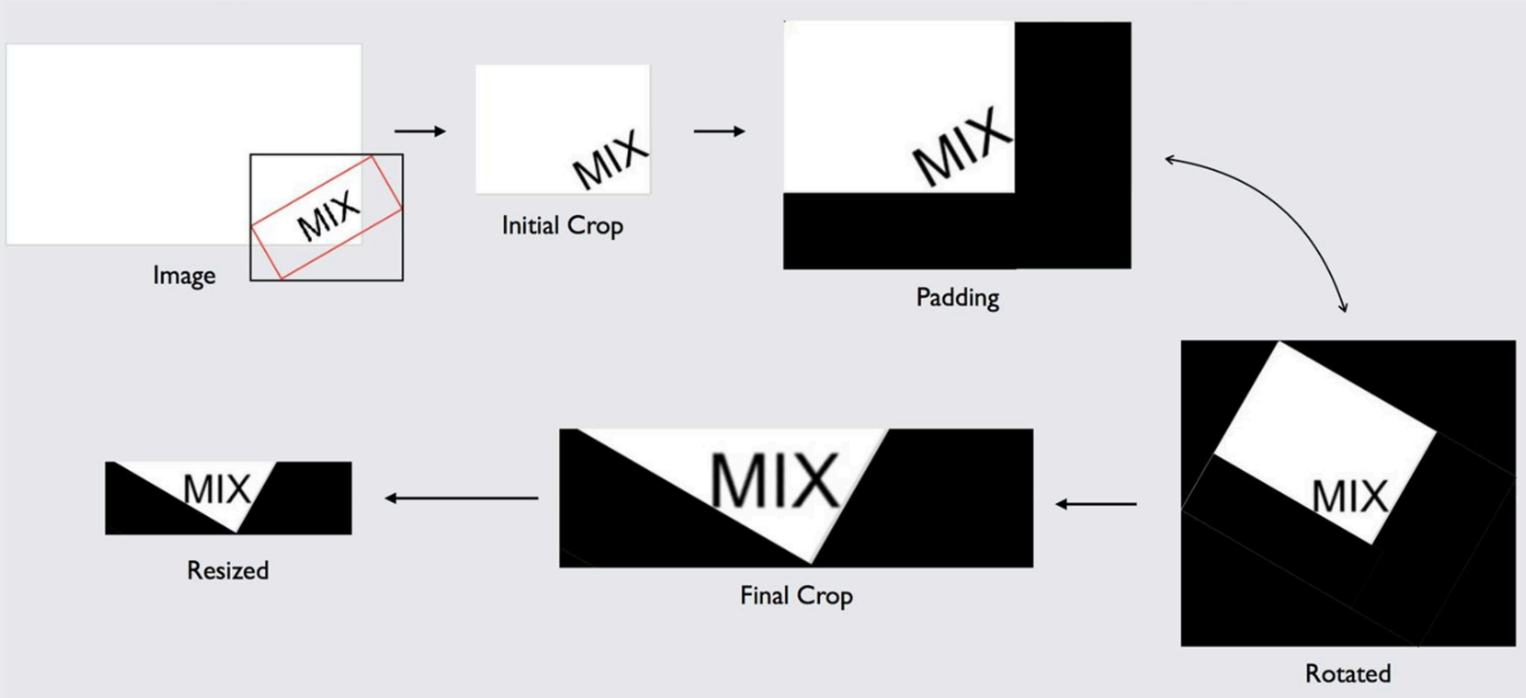


ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices, Zhang et al.

Orientations

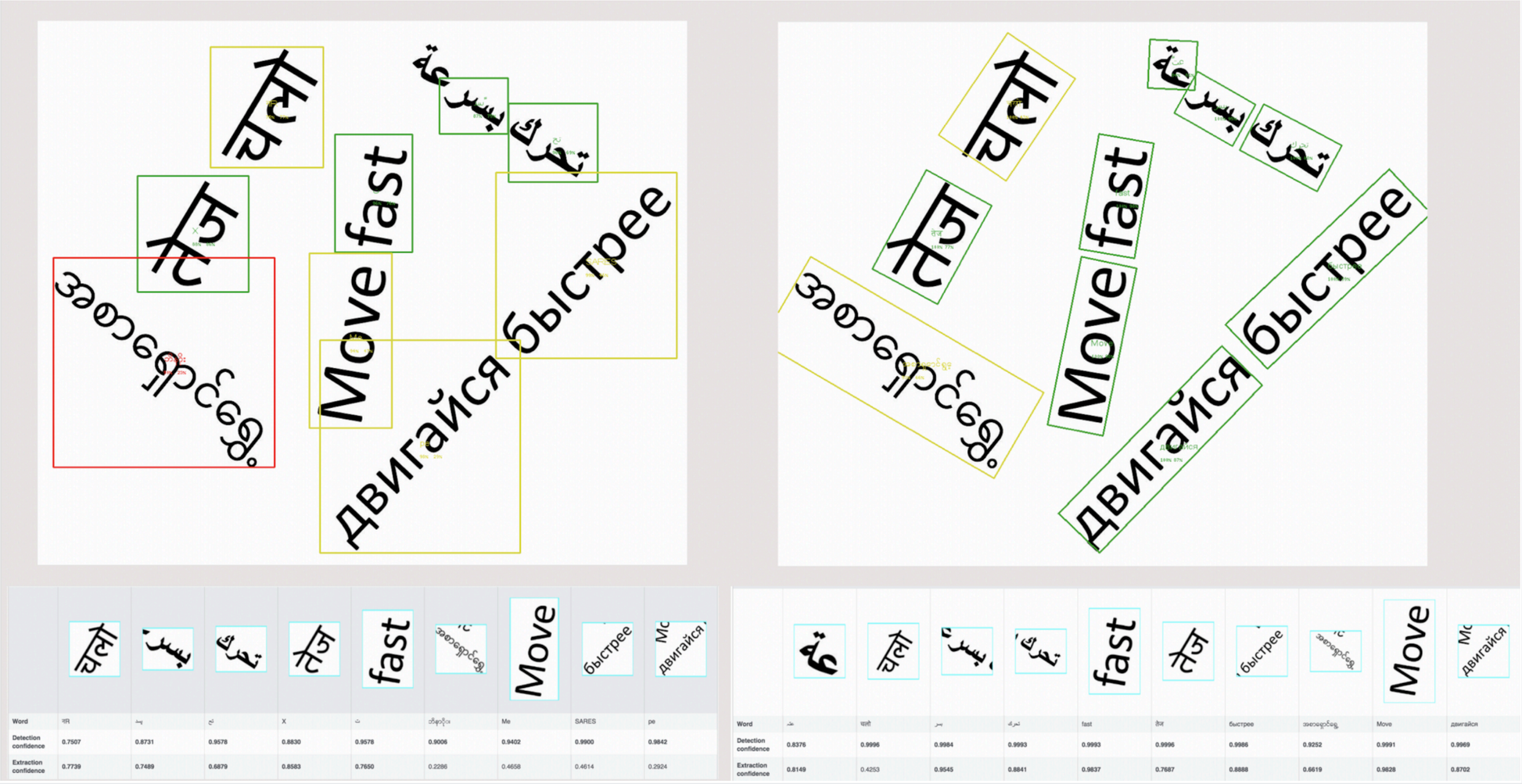
Extend Region Proposals with angle:

- More Anchors: |Aspect ratios| * |Scales| * |Orientations|
- Region of Interest alignment with affine transformation.
- Angles for region proposal quantized at 30 degree granularity.
- Regression to correct for +15/-15 delta.
- Tricky implementation, lots of edge cases!



Improving Rotated Text Detection with Rotation Region Proposal Networks, Jing Huang, **Viswanath Sivakumar**, Mher Mnatsakanyan, Guan Pang

Orientations



Open-sourced via PyTorch: https://github.com/pytorch/pytorch/tree/master/caffe2/operators

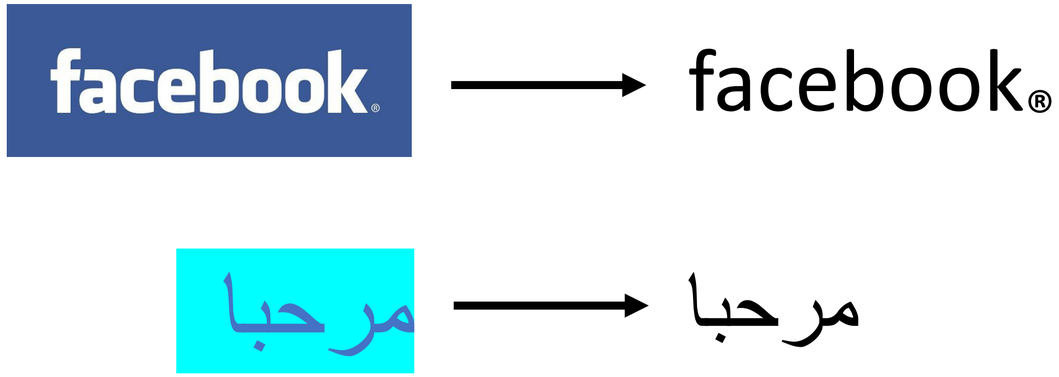
Agenda

1. Introduction
2. Text Detection
3. **Text Recognition**
4. Performance Optimizations
5. Additional Topics

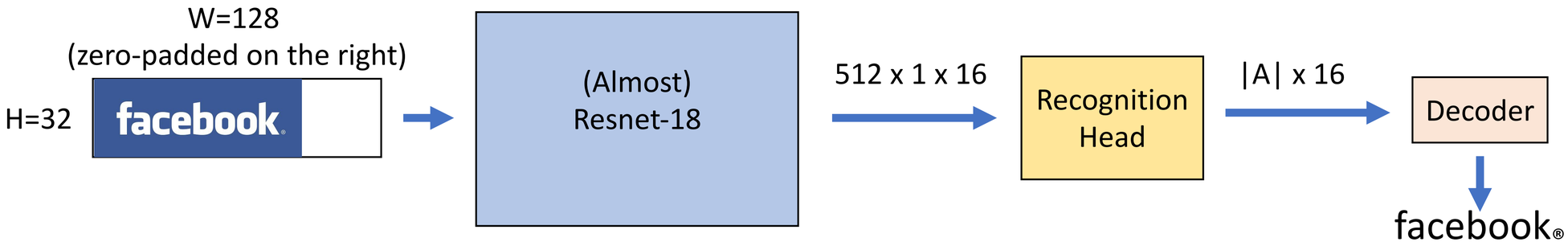
Goal: Text recognition

→ facebook®

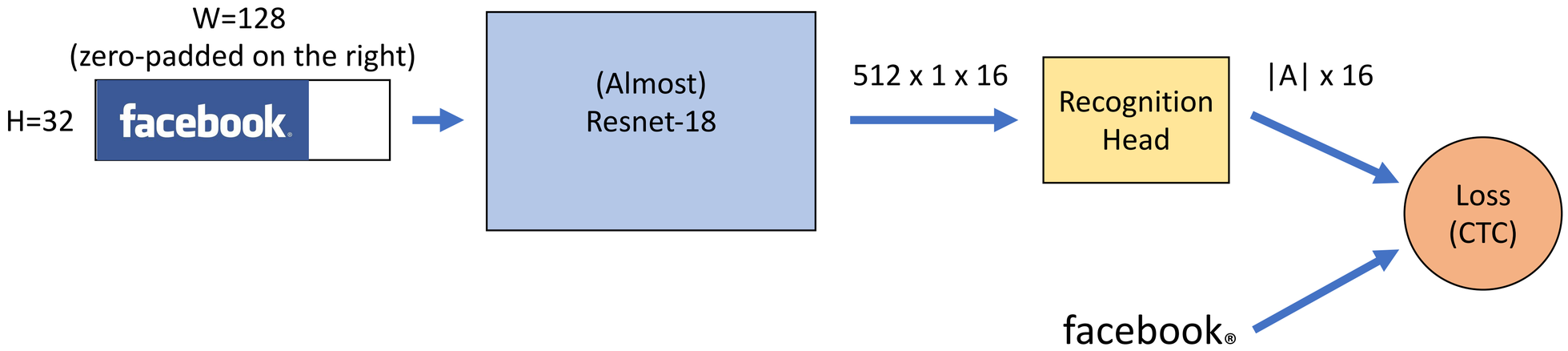
Goal: Text recognition (in many languages)



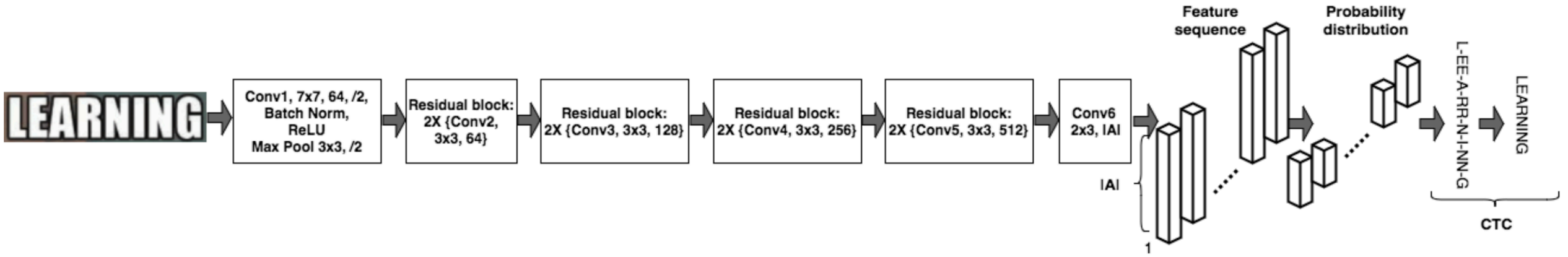
(Mostly our) current OCR



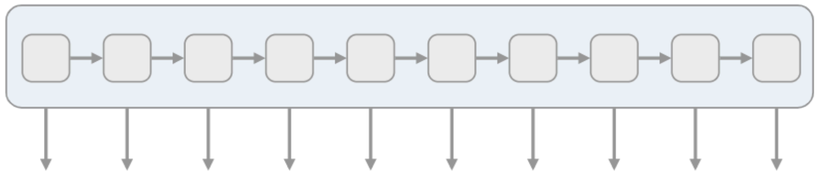
(Mostly our) current OCR



Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. Graves et al. ICML 2006.



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



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

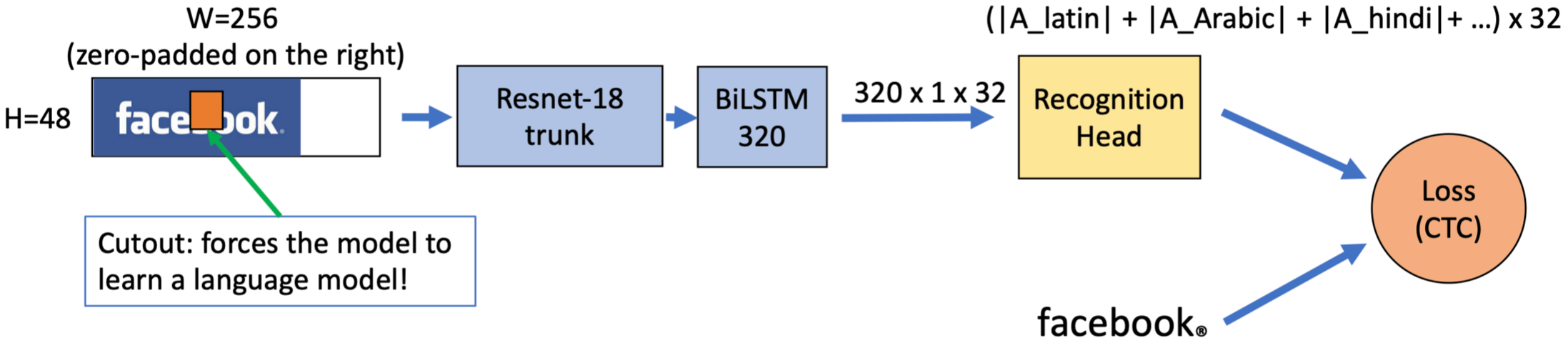
computing the **probability** for a single alignment step-by-step.

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/

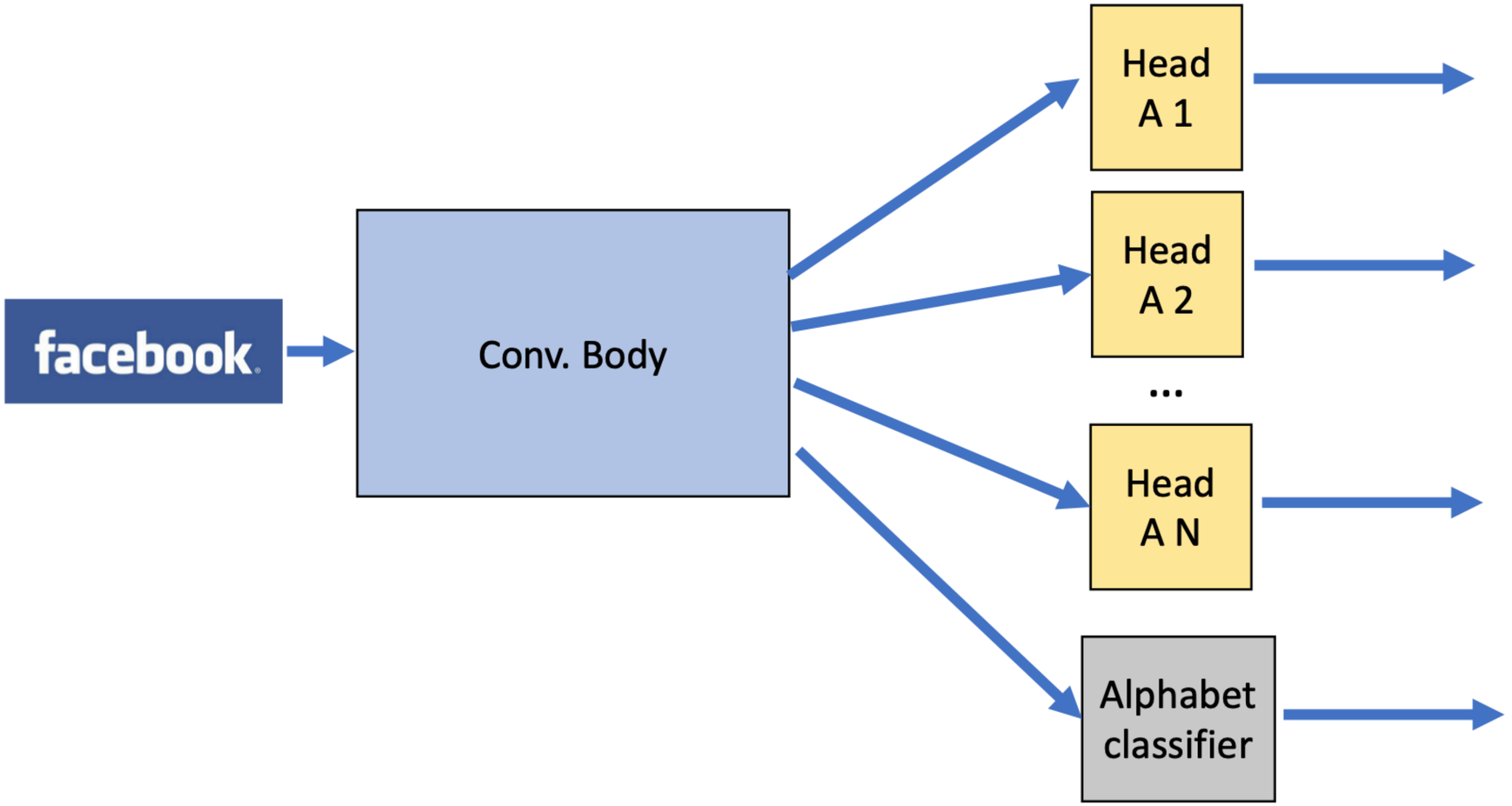
Multilanguage recognition



- Combine alphabets and train jointly
- Alternative: multihead model, one recognition head per encoding
  - Reduces label space, simplifying the classification task, but
  - One needs to choose the right head. Non-trivial
  - Trunk is still shared
- Hybrid approach: Main head plus auxiliary heads (e.g., English only)

Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. Graves et al. ICML 2006.

Multilanguage recognition

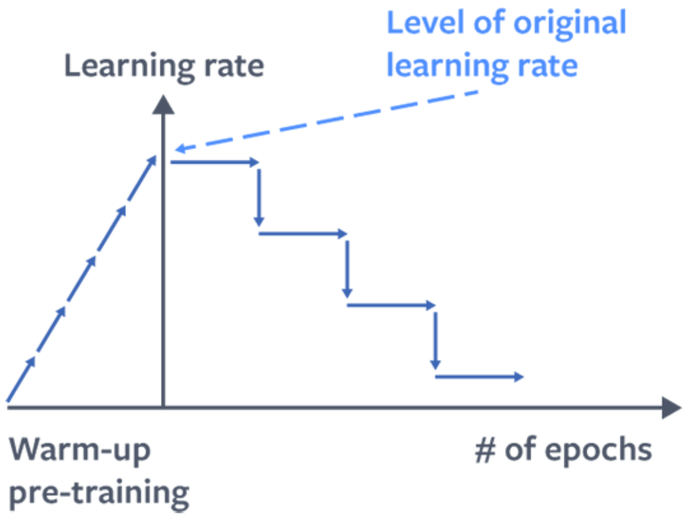


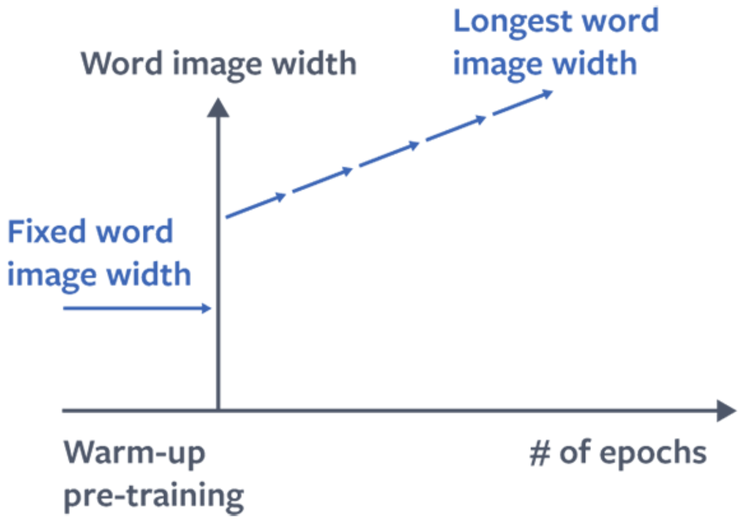
Other ~~tricks~~ well-principled design choices:

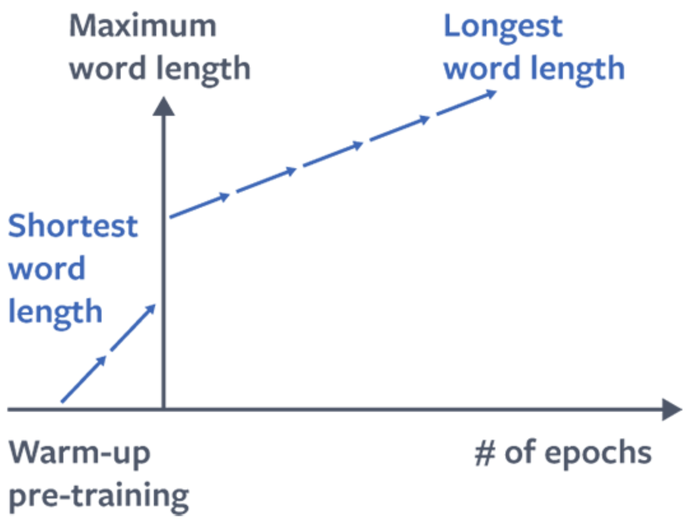
- Stretch words by a constant factor (~1.5) + jitter during training

facebook

- To recognize right-to-left languages: Flip the label, recognize left-to-right, flip the transcription
- Pretrain on synthetic data, finetune on real data.
- Learning rate warm up (only for synthetic pretraining)
- Small batch sizes (~64) seem to generalize better
- At test time, batch all words of the same image







Agenda

1. Introduction
2. Text Detection
3. Text Recognition
4. **Performance Optimizations**
5. Additional Topics

Inference Performance

2 billion images/day, 3 to 5 seconds per image => Lots of servers!

Text Recognition

- Small images => More compute bound than memory b/w
- Can "batch" all words in an image
- 20% increase in throughput

Intel Math Kernel Library

=> Intra-op parallelism

=> Helps to some extent if compute bound

Inference Performance

2 billion images/day, 3 to 5 seconds per image => Lots of servers!

Text Recognition

- Small images => More compute bound than memory b/w
- Can "batch" all words in an image
- 20% increase in throughput

Intel Math Kernel Library

=> Intra-op parallelism

=> Helps to some extent if compute bound

Text Detection

- High-res images (>= 800 x 800 px) for high accuracy
- Batching doesn't help:
  - Memory b/w bound system => batching leads to further bottleneck
  - Different aspect-ratios => Padding affects accuracy

Int8 Quantization

Idea: Can we get away with 8-bit integer weights and activations instead of 32-bit floats?

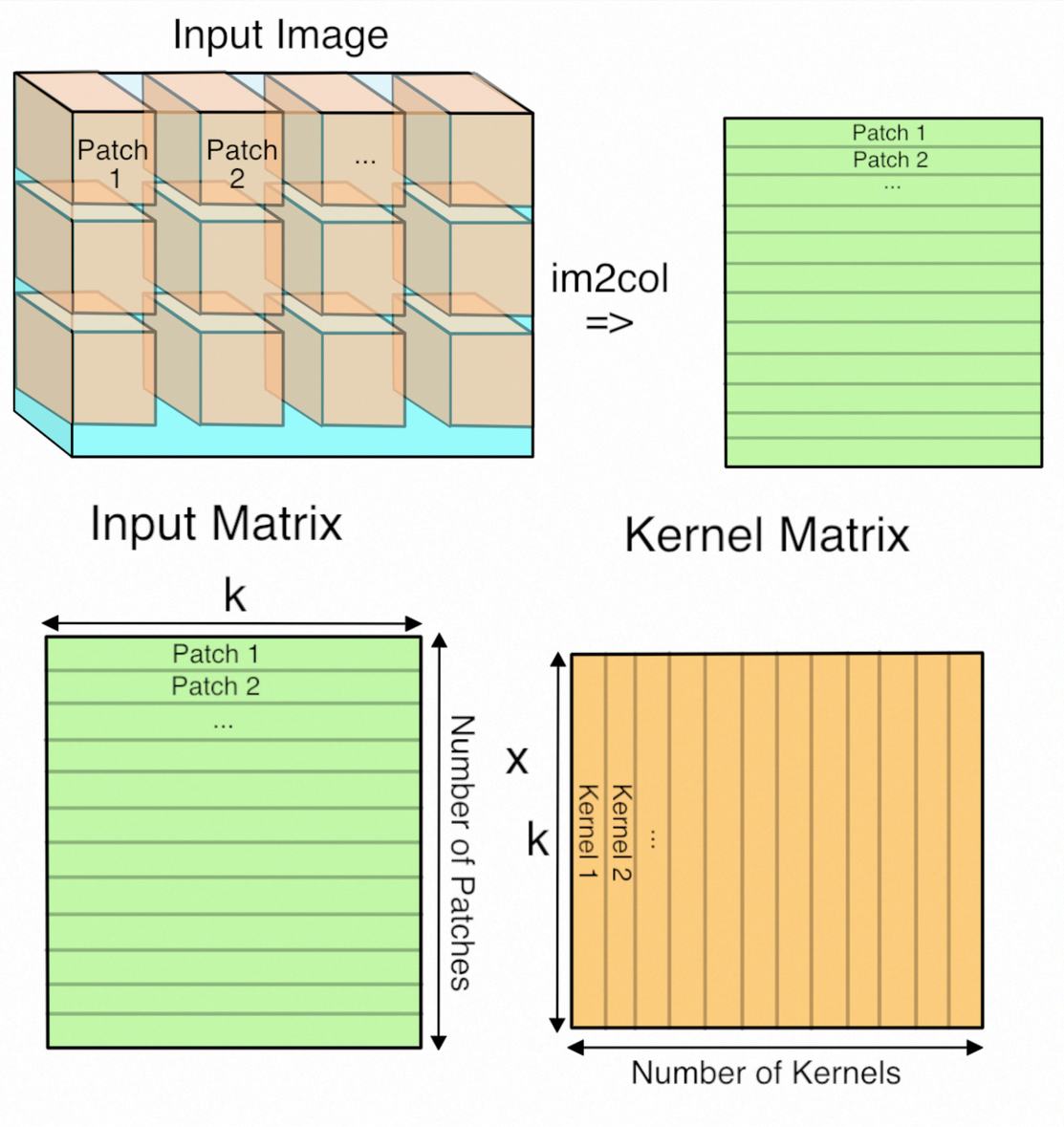
Int8 Quantization

Idea: Can we get away with 8-bit integer weights and activations instead of 32-bit floats?

Why?

- Research shows large neural networks have redundant representations
- 4x theoretical Memory Bandwidth reduction
- Faster integer arithmetic (more FLOPS) on Intel Skylake
- Production ML models need to support before custom ASIC: Training workflows, accuracy tradeoffs

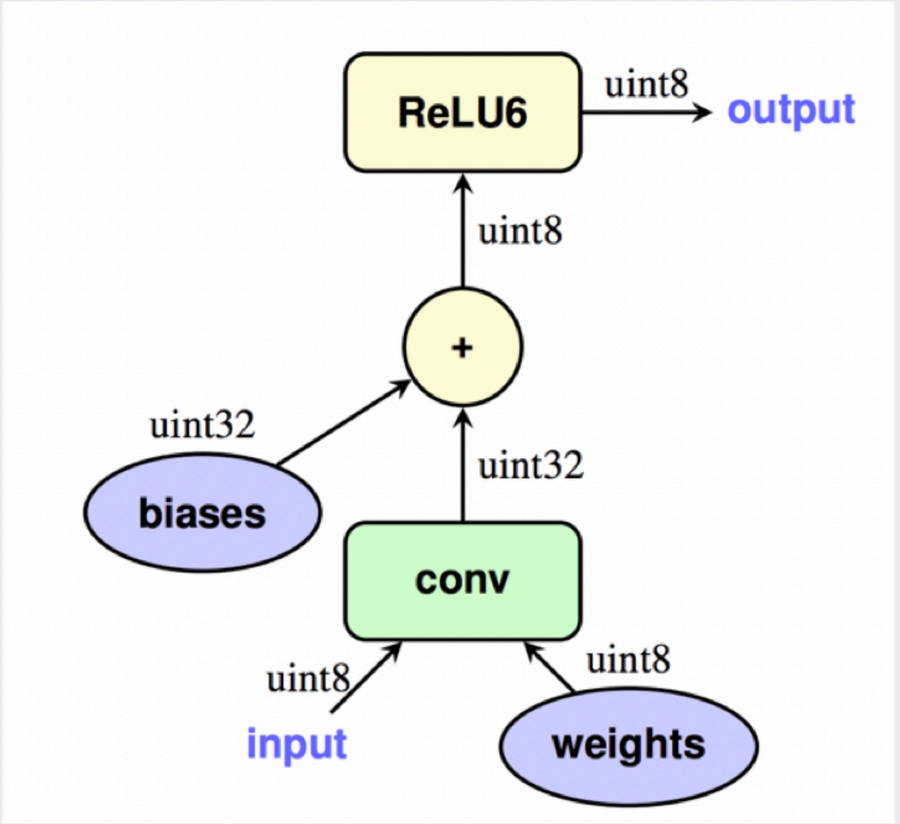
Convolution as GEMM



- GEMM: General Matrix Multiply
- Convolution = im2col + GEMM
- 32-bit floating point weights, activations and GEMM
- Hi-res image
- => Large input matrix to GEMM
- => Large output activations
- => Activations don’t fit in LLC
- => More memory I/O for GEMM
- Moreover, compute-optimized CNN architectures for mobile (ShuffleNet) cause greater Mem B/W boundedness

Int8 Quantization

- Linear quantization (lookup tables perform poorly on H/W)
- fp32 tensor values linearly mapped to [0, 255] uint8_t range
- real_value = scale * (quantized_value - offset)
- Post-training quantization: Simpler than training with quantization
- One (scale, offset) pair pre-computed for each tensor
- float32 min(Tensor) -> 0, float32 max(Tensor) -> 255
- fp32 GEMM -> int8 GEMM with int32 accumulation



Int8 Quantization - Accuracy

Goal: Minimize accuracy drop compared to fp32.

1. Zero must be exactly mapped to a value in [0, 255] (no quantization error).

2. Fuse Conv-Relu ops:
   Conv output = [-a, b]
   Relu(x) = max(0, x)
   Conv-Relu output = [0, b].

   Don’t waste precious uint8_t range quantizing [-a, 0) which will be thrown away by Relu. Allocate the entire [0, 255] range to Conv-Relu output.

3. L2 Error Minimization: Robust to outliers

   By default, min/max of tensor mapped to [0, 255]. Instead, generate histogram of tensor values and pick the best range that minimizes L2 reconstruction error.

4. Don’t quantize first Conv: Highly sensitive to accuracy, very cheap (~20ms)

Int8 Quantization - Accuracy

Goal: Minimize accuracy drop compared to fp32.

1. Zero must be exactly mapped to a value in [0, 255] (no quantization error).

2. Fuse Conv-Relu ops:
   Conv output = [-a, b]
   Relu(x) = max(0, x)
   Conv-Relu output = [0, b].

   Don’t waste precious uint8_t range quantizing [-a, 0) which will be thrown away by Relu. Allocate the entire [0, 255] range to Conv-Relu output.

3. L2 Error Minimization: Robust to outliers

   By default, min/max of tensor mapped to [0, 255]. Instead, generate histogram of tensor values and pick the best range that minimizes L2 reconstruction error.

4. Don’t quantize first Conv: Highly sensitive to accuracy, very cheap (~20ms)

Initial accuracy gap: 5%

After: 0.2%

Int8 Quantization - Implementation

DNNLOWP: https://github.com/pytorch/pytorch/tree/master/caffe2/quantization/server

Targeted for inference in AVX2/AVX512 servers

Hand-optimized for Intel Broadwell and Skylake compared to Intel MKL-DNN int8 libraries:

- Highly optimized 3x3 int8 Depthwise Conv kernel => Total Conv runtime from 516ms to 330ms
- Output rescaling fused into the GEMM kernel for temporal reuse of intermediate int32 buffers => Significant Mem B/W reduction
- AVX2 optimized transpose for ChannelShuffle => 168ms to 60ms
- Share intermediate im2col buffer across Conv ops => Memory usage reduction

Int8 Quantization - Results

- 2.4x throughput increase
- 2x drop in inference time (2s to 1.2s at p50, 5s to 2.5s at p99)
- 42% drop in memory bandwidth

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

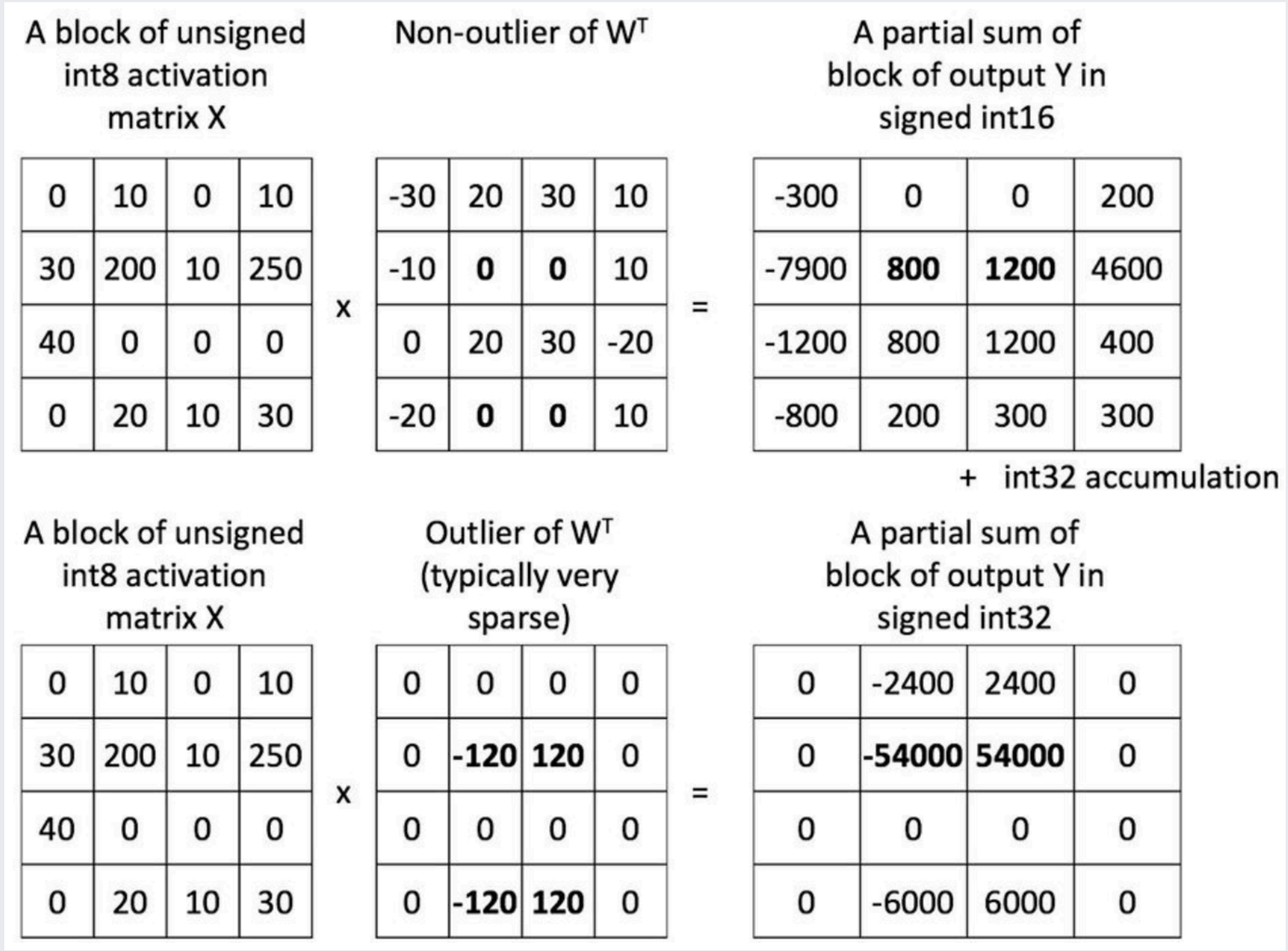
-30	20	30	10
-10	**-120**	**120**	10
0	20	30	-20
-20	**-120**	**120**	10

=

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)

Outlier-Aware Quantization

Key: Important to map 0.0 (fp32) to 0 (int8) in W

- Otherwise Sparse_W won't really be sparse
- Assumption: Histogram of W peaks around 0

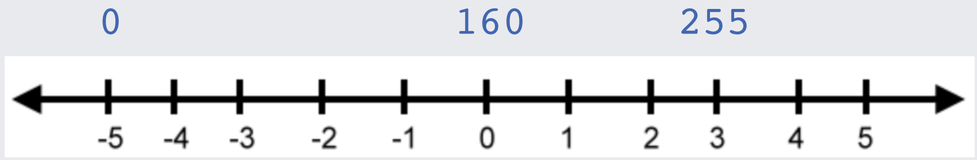
Outlier-Aware Quantization

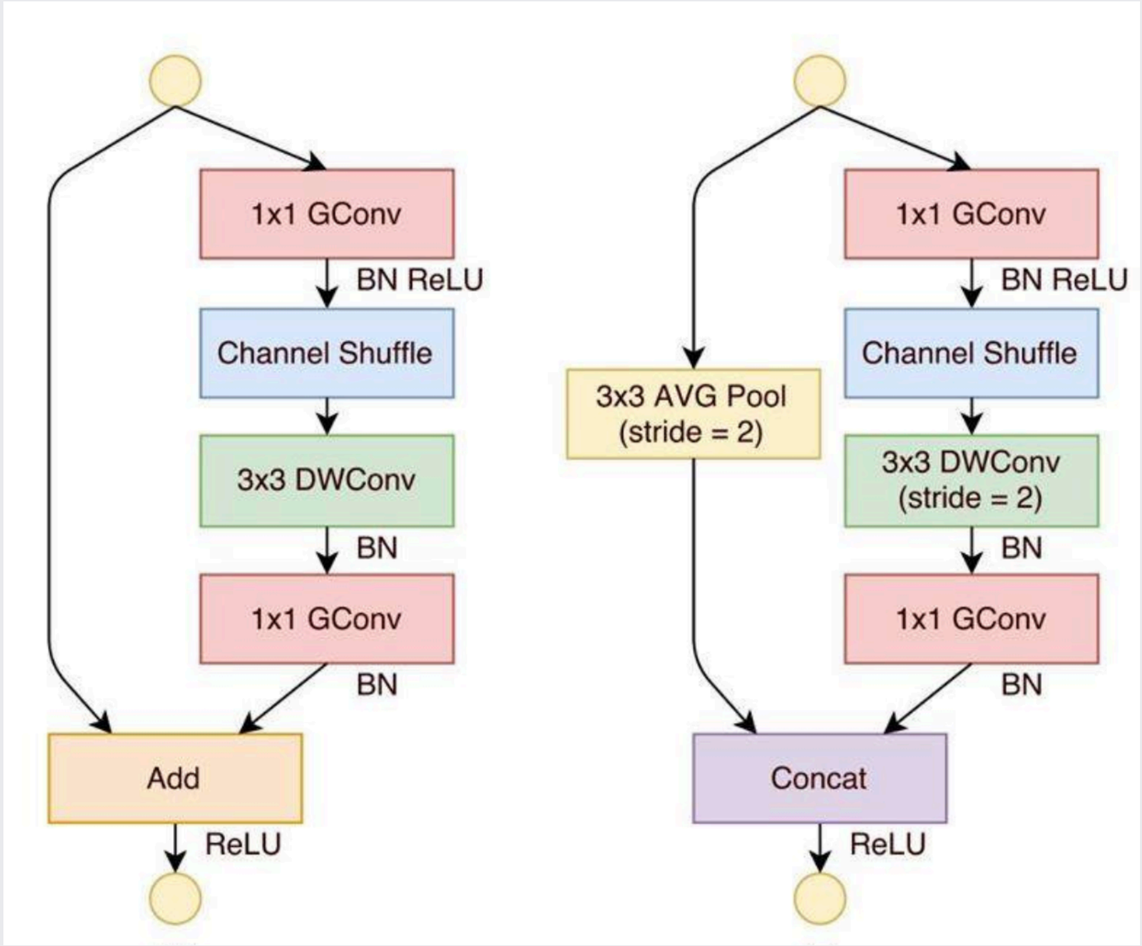
Key: Important to map 0.0 (fp32) to 0 (int8) in W

- Otherwise Sparse_W won't really be sparse
- Assumption: Histogram of W peaks around 0

ShuffleNet specific-problem:

- Not all Conv are followed by ReLU
- => Less sparsity in activation matrix
- => Bigger loss in accuracy





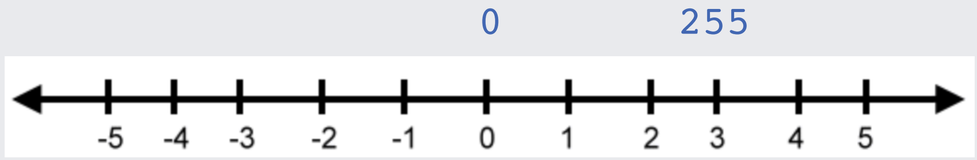
Outlier-Aware Quantization

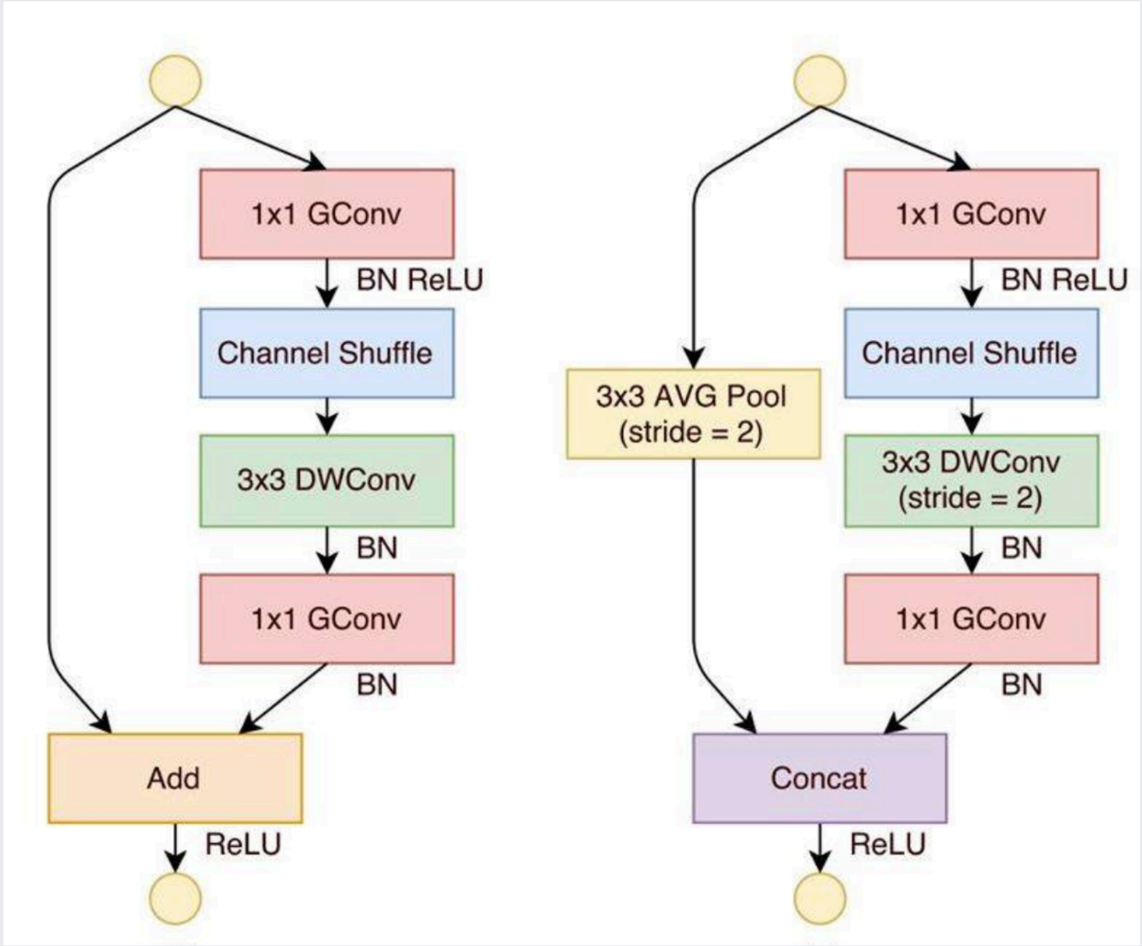
Key: Important to map 0.0 (fp32) to 0 (int8) in W

- Otherwise Sparse_W won't really be sparse
- Assumption: Histogram of W peaks around 0

ShuffleNet specific-problem:

- Not all Conv are followed by ReLU
- => Less sparsity in activation matrix
- => Bigger loss in accuracy
- Just introduce ReLU and retrain!





- Further 2x drop in inference time
- 25% drop in CPU-util
- 11% increase in throughput

Agenda

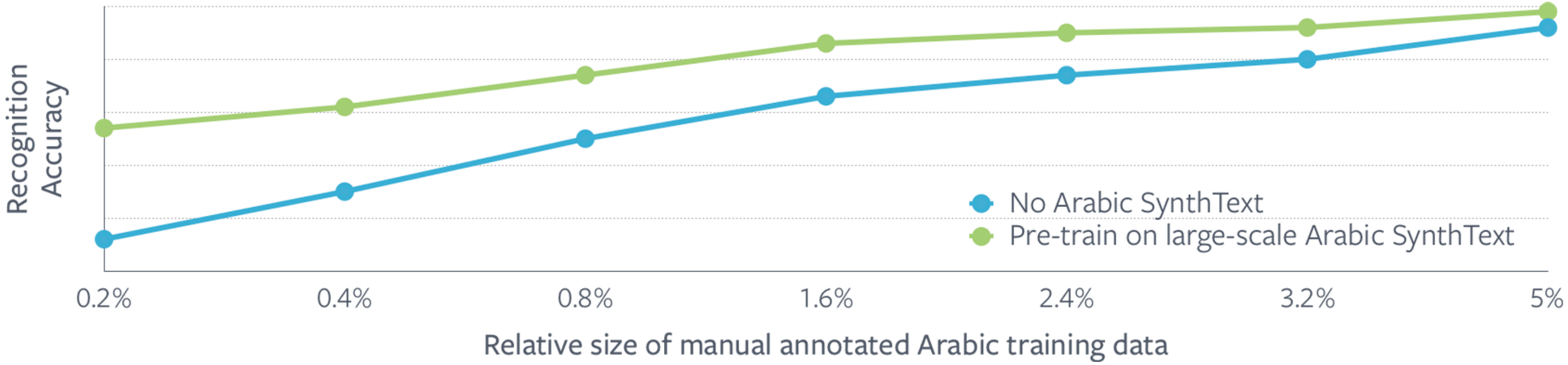
1. Introduction
2. Text Detection
3. Text Recognition
4. Performance Optimizations
5. **Additional Topics**

SynthText

Code for generating synthetic text images as described in "Synthetic Data for Text Localisation in Natural Images", Ankush Gupta, Andrea Vedaldi, Andrew Zisserman, CVPR 2016.

Synthetic Scene-Text Image Samples

Synthetic Data for Text Localisation in Natural Images, Gupta et al.


Recognition Accuracy
No Arabic SynthText
Pre-train on large-scale Arabic SynthText
0.2%
0.4%
0.8%
1.6%
2.4%
3.2%
5%
Relative size of manual annotated Arabic training data

Training

Classic data-parallel training

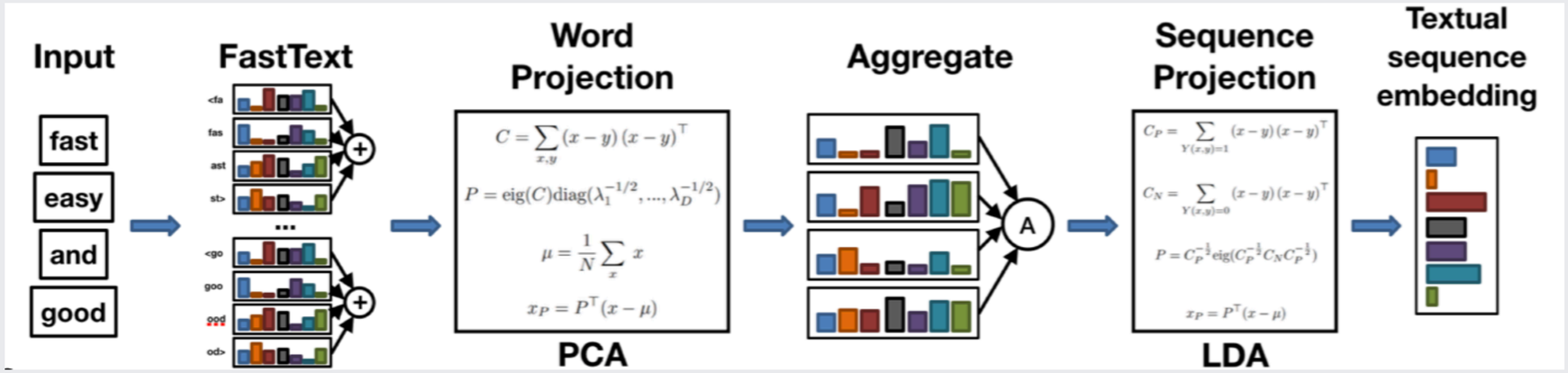
- Scale learning rate linearly with total batch size
- Trained on ~32 GPUs in parallel

ImageNet shows no accuracy drop until total batch size = 8192

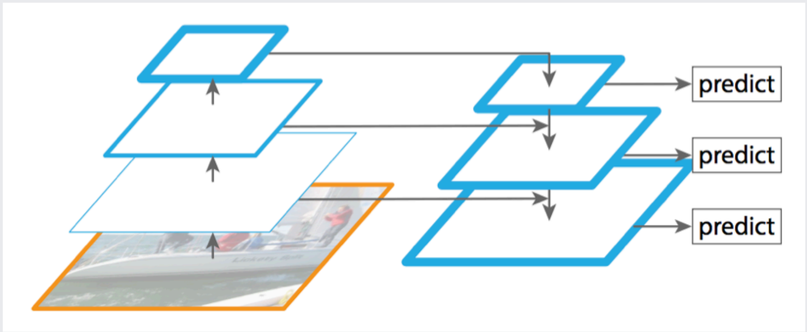
- Ref: Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, Goyal et al.

Additional Topics

Learning (multi-lingual) image embeddings that contain text information



Feature Pyramid Networks for detecting small text



Feature Pyramid Networks for Object Detection, Lin et al.

Detecting warped text by learning pixel-wise affine transformation

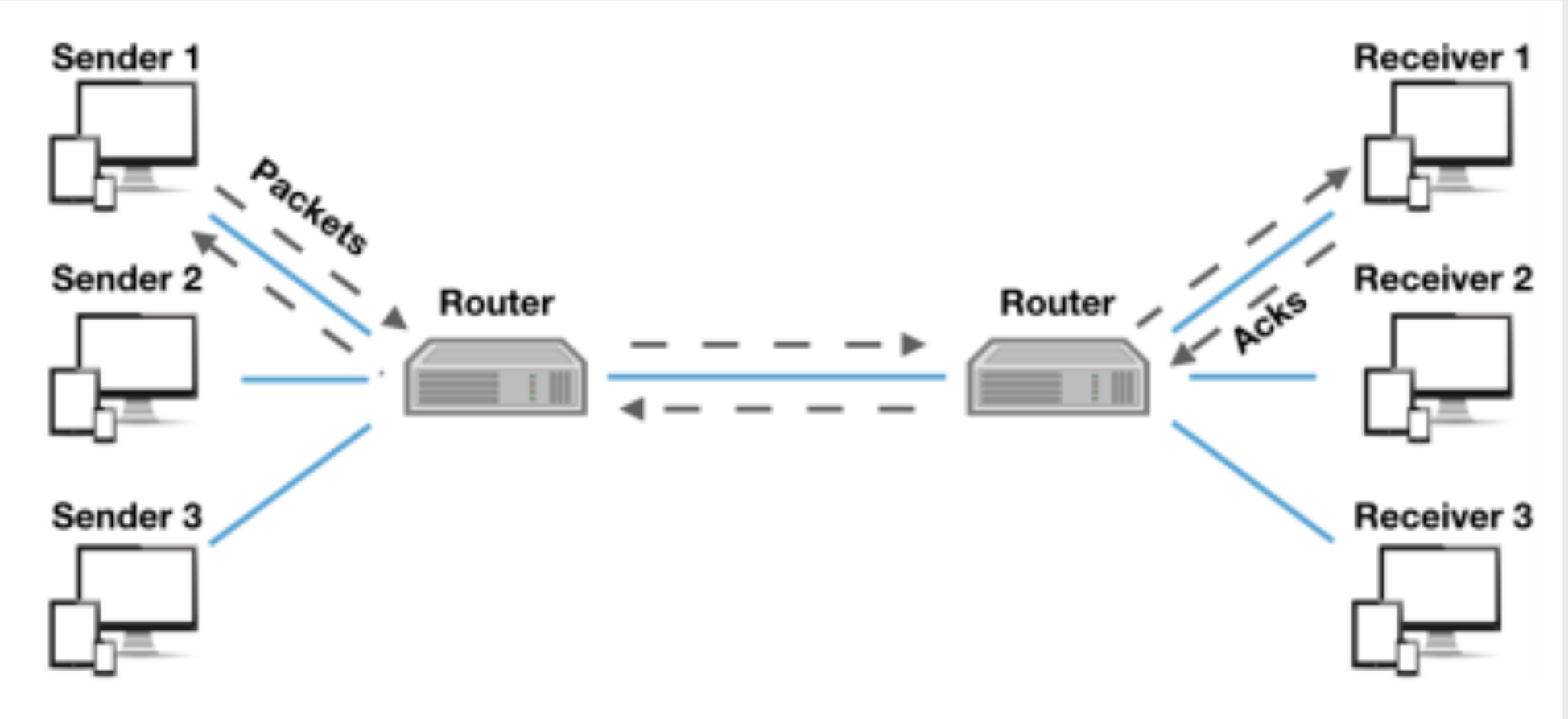
Huge credits to my colleagues at Facebook AI Applied Research!

Reinforcement Learning for TCP Congestion Control

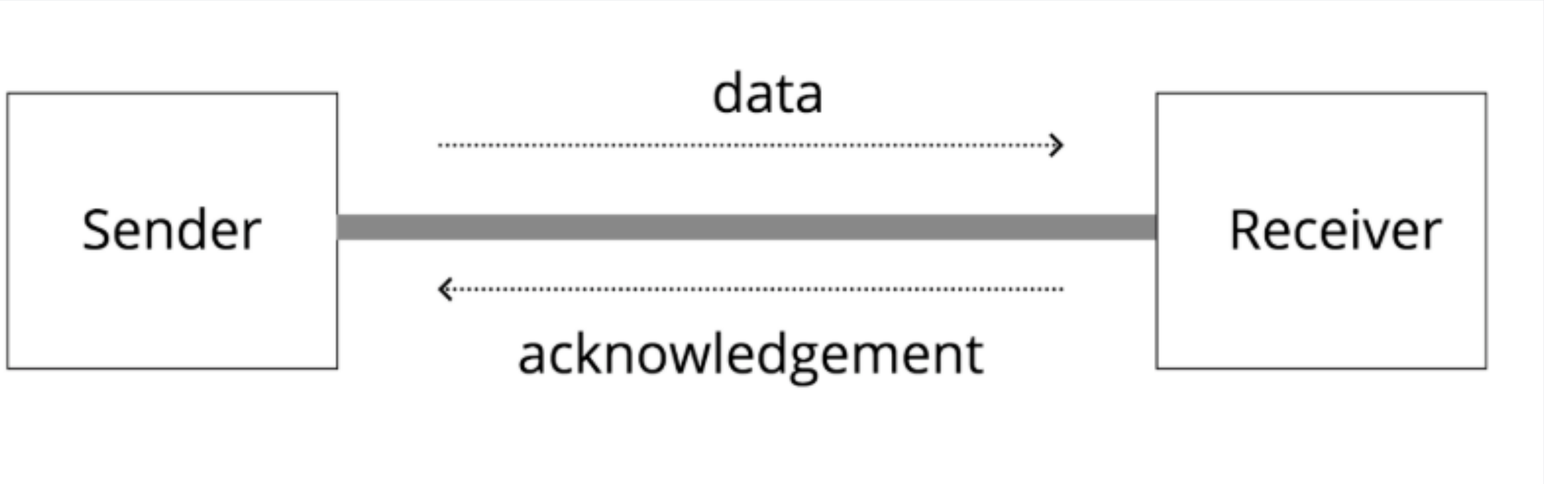
Advisors:

Prof. Sebastian Riedel, FAIR and University College London
Prof. Joelle Pineau, FAIR and McGill University

Network Congestion



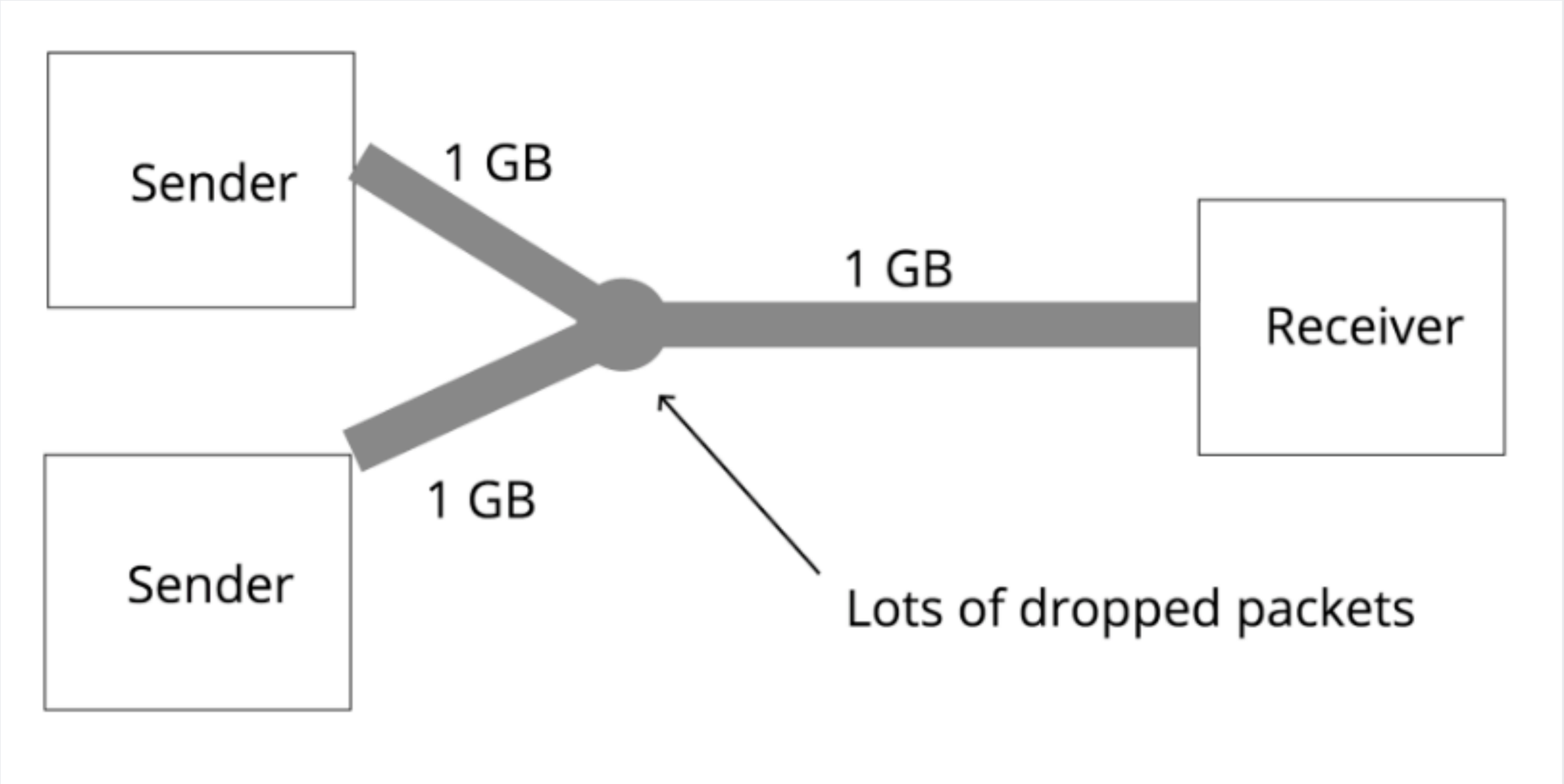
Computer Networks 101



ACK tells you:

- The largest packet number received
- Measured round-trip time (RTT)
- ...

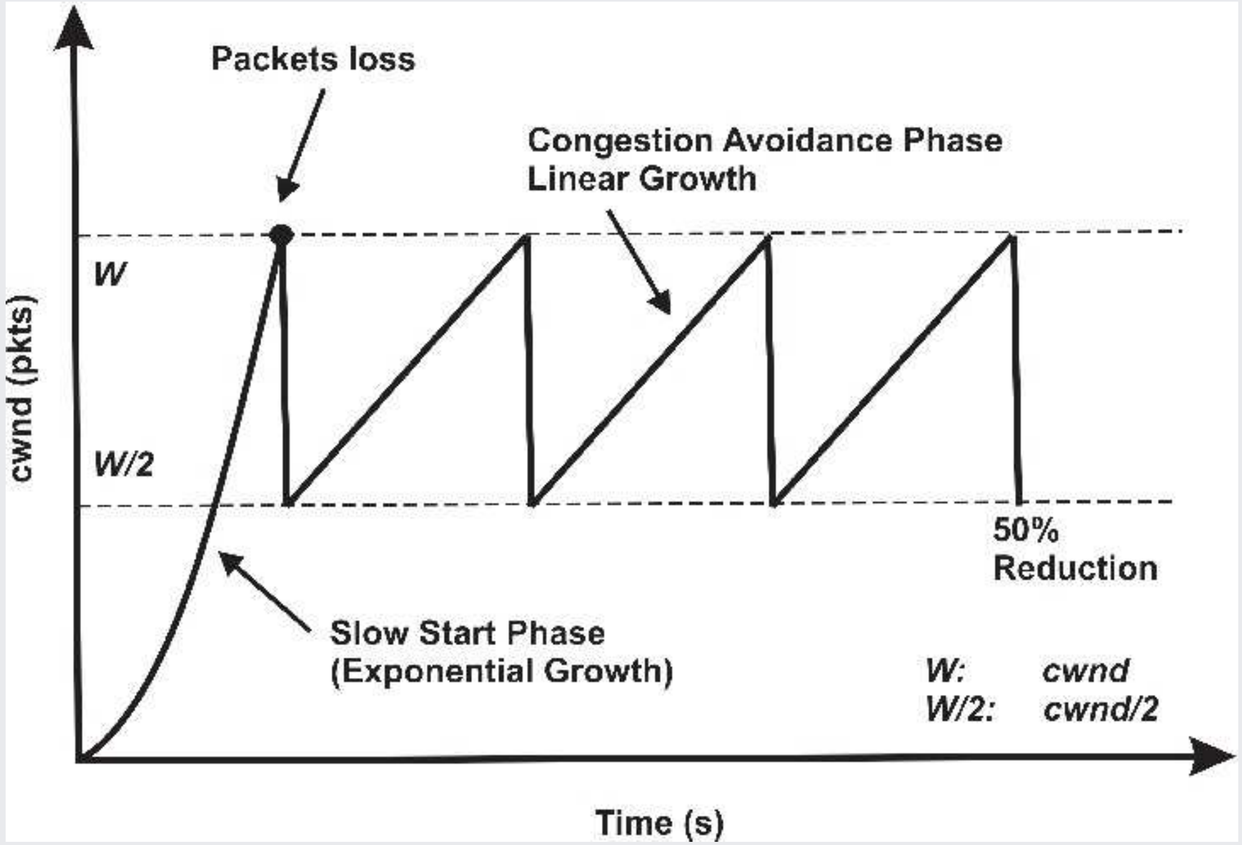
Computer Networks 101



Traditional Congestion Control

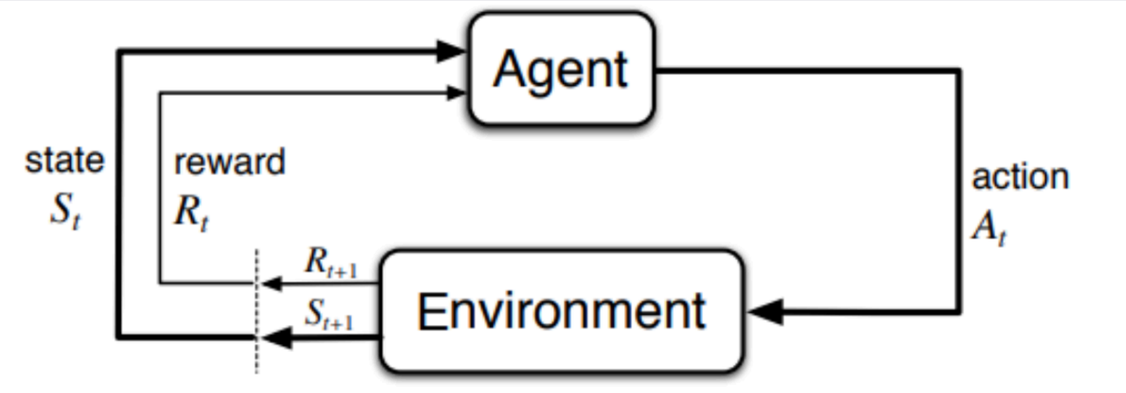
- Hand-engineered
- Decentralized
- Reactive, not predictive
- Often too conservative

30+ years of active research!



RL Basics

<S, A, R, S'>

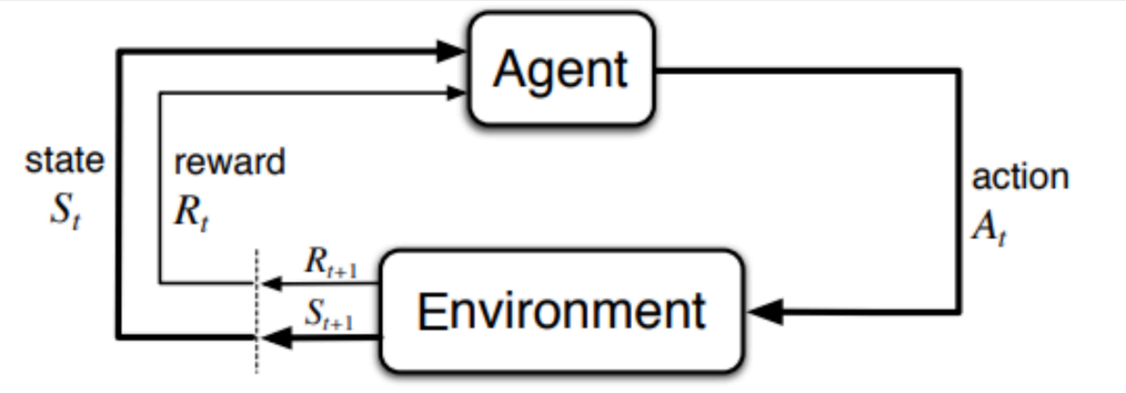


Markov Decision Process (MDP)

- Next state S' depends only on current state S and action A

RL Basics

<S, A, R, S'>



Policy Gradient Methods:

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$$

Markov Decision Process (MDP)

- Next state S' depends only on current state S and action A

Congestion Control as RL

States:

Network conditions:

- Round-trip time
- Bytes sent
- Packets dropped
- Network delay
- ...

Heterogeneous state space

```
struct TransportInfo {
  std::chrono::microseconds srtt;
  std::chrono::microseconds rttvar;
  std::chrono::microseconds lrtt;
  std::chrono::microseconds mrtt;
  uint64_t writableBytes;
  uint64_t congestionWindow;
  uint64_t pacingBurstSize;
  std::chrono::microseconds pacingInterval;
  uint32_t packetsRetransmitted;
  uint32_t timeoutBasedLoss;
  std::chrono::microseconds pto;
  uint64_t bytesSent;
  uint64_t bytesAcked;
  uint64_t bytesRecvd;
  uint64_t totalBytesRetransmitted;
  uint32_t ptoCount;
  uint32_t totalPTOCount;
  PacketNum largestPacketAckedByPeer;
  PacketNum largestPacketSent;
};
```

Congestion Control as RL

States:

Network conditions:

- Round-trip time
- Bytes sent
- Packets dropped
- Network delay
- ...

Heterogeneous state space

Action:

- Modify congestion window (cwnd)
- Fixed update: [cwnd / 2, cwnd - x, cwnd, cwnd + x, cwnd * 2]

```
struct TransportInfo {
  std::chrono::microseconds srtt;
  std::chrono::microseconds rttvar;
  std::chrono::microseconds lrtt;
  std::chrono::microseconds mrtt;
  uint64_t writableBytes;
  uint64_t congestionWindow;
  uint64_t pacingBurstSize;
  std::chrono::microseconds pacingInterval;
  uint32_t packetsRetransmitted;
  uint32_t timeoutBasedLoss;
  std::chrono::microseconds pto;
  uint64_t bytesSent;
  uint64_t bytesAcked;
  uint64_t bytesRecvd;
  uint64_t totalBytesRetransmitted;
  uint32_t ptoCount;
  uint32_t totalPTOCount;
  PacketNum largestPacketAckedByPeer;
  PacketNum largestPacketSent;
};
```

Congestion Control as RL

States:

Network conditions:

- Round-trip time
- Bytes sent
- Packets dropped
- Network delay
- ...

Heterogeneous state space

Action:

- Modify congestion window (cwnd)
- Fixed update: [cwnd / 2, cwnd - x, cwnd, cwnd + x, cwnd * 2]

reward = f(throughput) - alpha * g(delay)

```
struct TransportInfo {
  std::chrono::microseconds srtt;
  std::chrono::microseconds rttvar;
  std::chrono::microseconds lrtt;
  std::chrono::microseconds mrtt;
  uint64_t writableBytes;
  uint64_t congestionWindow;
  uint64_t pacingBurstSize;
  std::chrono::microseconds pacingInterval;
  uint32_t packetsRetransmitted;
  uint32_t timeoutBasedLoss;
  std::chrono::microseconds pto;
  uint64_t bytesSent;
  uint64_t bytesAcked;
  uint64_t bytesRecvd;
  uint64_t totalBytesRetransmitted;
  uint32_t ptoCount;
  uint32_t totalPTOCount;
  PacketNum largestPacketAckedByPeer;
  PacketNum largestPacketSent;
};
```

Congestion Control as RL

States:

Network conditions:

- Round-trip time
- Bytes sent
- Packets dropped
- Network delay
- ...

Heterogeneous state space

Action:

- Modify congestion window (cwnd)
- Fixed update: [cwnd / 2, cwnd - x, cwnd, cwnd + x, cwnd * 2]

reward = f(throughput) - alpha * g(delay)

Partially Observable MDP (POMDP)!

```
struct TransportInfo {
  std::chrono::microseconds srtt;
  std::chrono::microseconds rttvar;
  std::chrono::microseconds lrtt;
  std::chrono::microseconds mrtt;
  uint64_t writableBytes;
  uint64_t congestionWindow;
  uint64_t pacingBurstSize;
  std::chrono::microseconds pacingInterval;
  uint32_t packetsRetransmitted;
  uint32_t timeoutBasedLoss;
  std::chrono::microseconds pto;
  uint64_t bytesSent;
  uint64_t bytesAcked;
  uint64_t bytesRecvd;
  uint64_t totalBytesRetransmitted;
  uint32_t ptoCount;
  uint32_t totalPTOCount;
  PacketNum largestPacketAckedByPeer;
  PacketNum largestPacketSent;
};
```

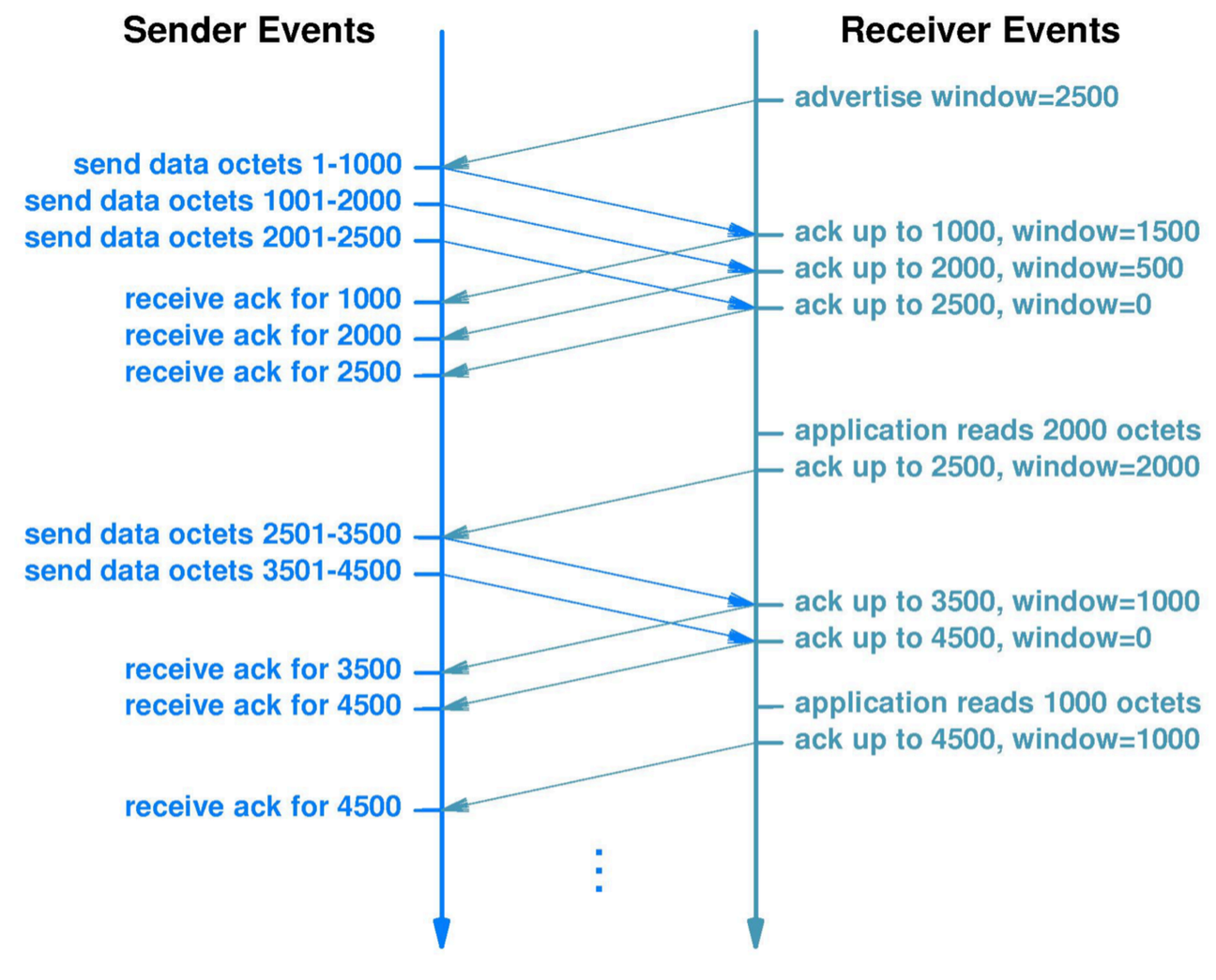
Traditional RL Environments

```
env = ...
state = env.reset()
for _ in range(1000):
  action = model(state)
  state, reward, done = env.step(action)
  if done:
    state = env.reset()
```

Traditional RL Environments

```
env = ...
state = env.reset()
for _ in range(1000):
  action = model(state)  # Blocks the environment
  state, reward, done = env.step(action)
  if done:
    state = env.reset()
```

Continuous Environment



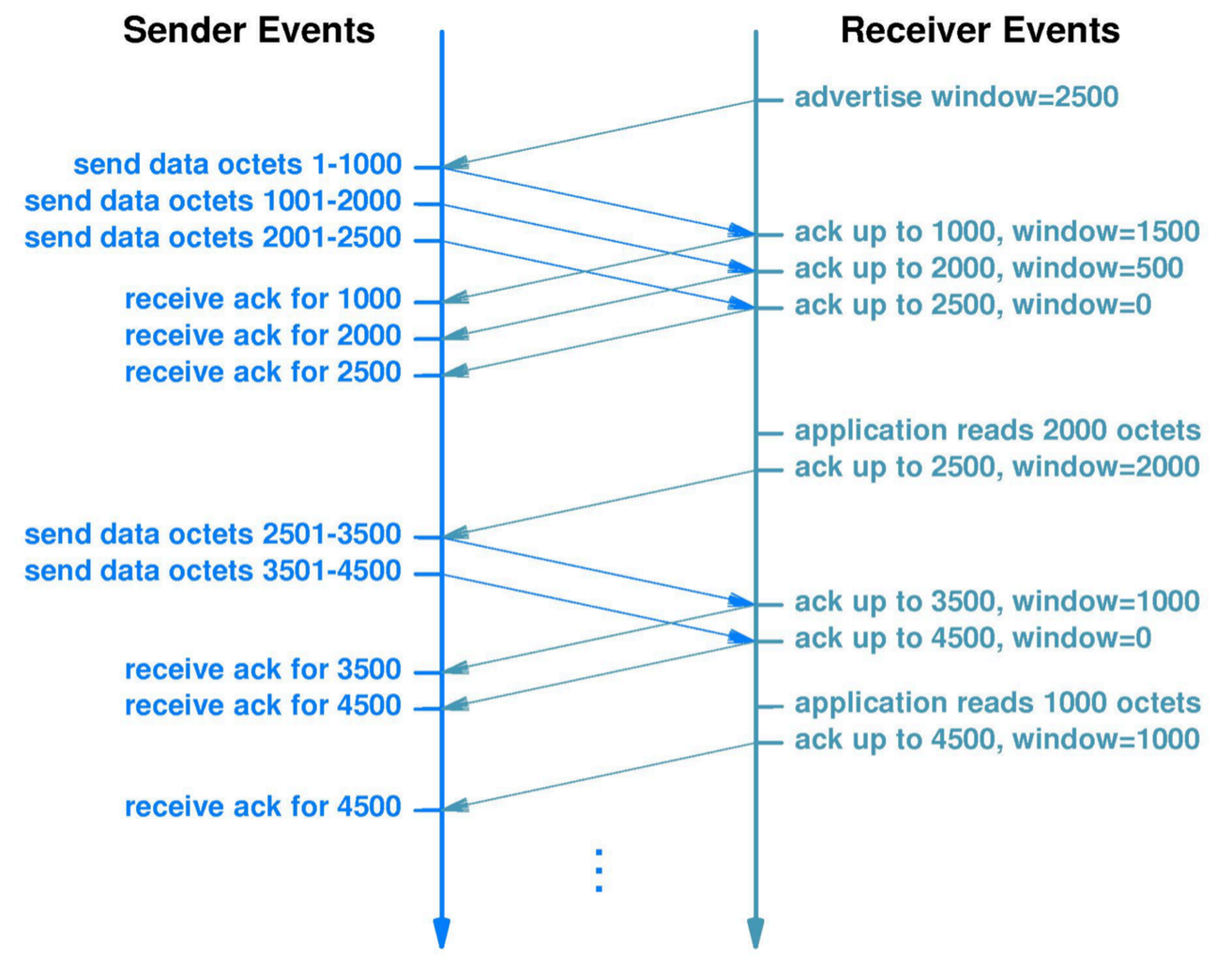
Typical RL Env:

- <S, A, R, S'>
- <R, S'> corresponds to <S, A>

Async Env:

- Sequence of S' and R'
- Non-blocking agent
- Another step/action A' taken before obtaining S' corresponding to A.

Continuous Environment



Typical RL Env:

- <S, A, R, S'>
- <R, S'> corresponds to <S, A>

Async Env:

- Sequence of S' and R'
- Non-blocking agent
- Another step/action A' taken before obtaining S' corresponding to A.

Delayed MDP!

<S, A, R, S', k>

k = constant delay factor

Actor-Critic Methods

Policy Gradient Loss

$$maximize_\theta \, E_\pi[R(\tau)]$$

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$$

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

$$\nabla_\theta E_\pi[R(\tau)] = E_\pi[\nabla_\theta log\, \pi_\theta(\tau)\, R(\tau)]$$

Intuition: Nudge params towards higher probability for a trajectory with high reward.

Problem: High variance in $R(\tau)$.

Actor-Critic Methods

Policy Gradient Loss

$$maximize_\theta \, E_\pi[R(\tau)]$$

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$$

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

$$\nabla_\theta E_\pi[R(\tau)] = E_\pi[\nabla_\theta log\, \pi_\theta(\tau)\, R(\tau)]$$

Intuition: Nudge params towards higher probability for a trajectory with high reward.

Problem: High variance in $R(\tau)$.

Actor-Critic

Key: Estimate $R(\tau)$ with a model.

- This model is called the value function
- Trained using L2-loss

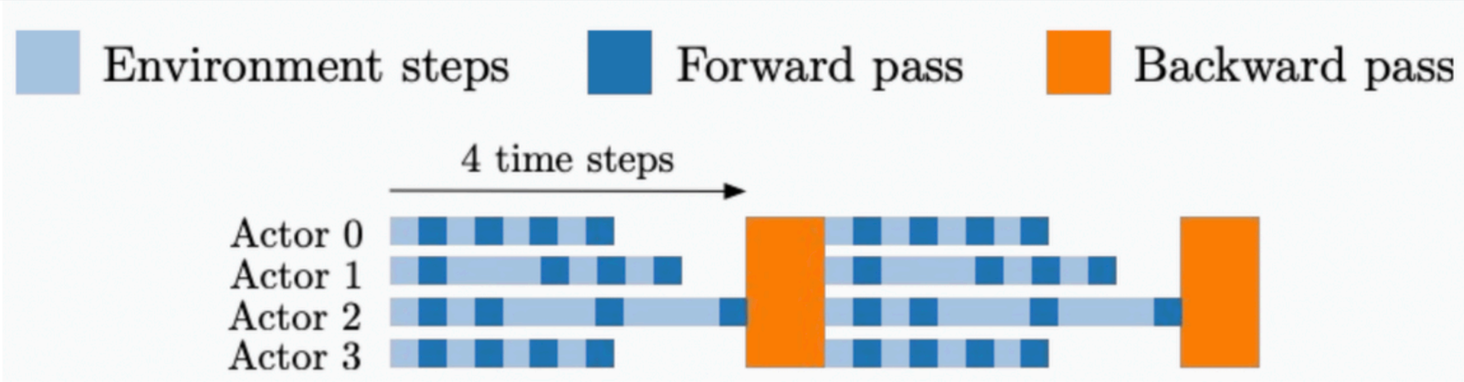
$$\nabla_\theta E_\pi[R(\tau)] = E_\pi[\nabla_\theta log\, \pi_\theta(\tau)\, V(s)]$$

Two models:

- Actor: Policy function $\pi_\theta(a \mid s)$ - next action
- Critic: Value function $V(s)$ - estimated reward

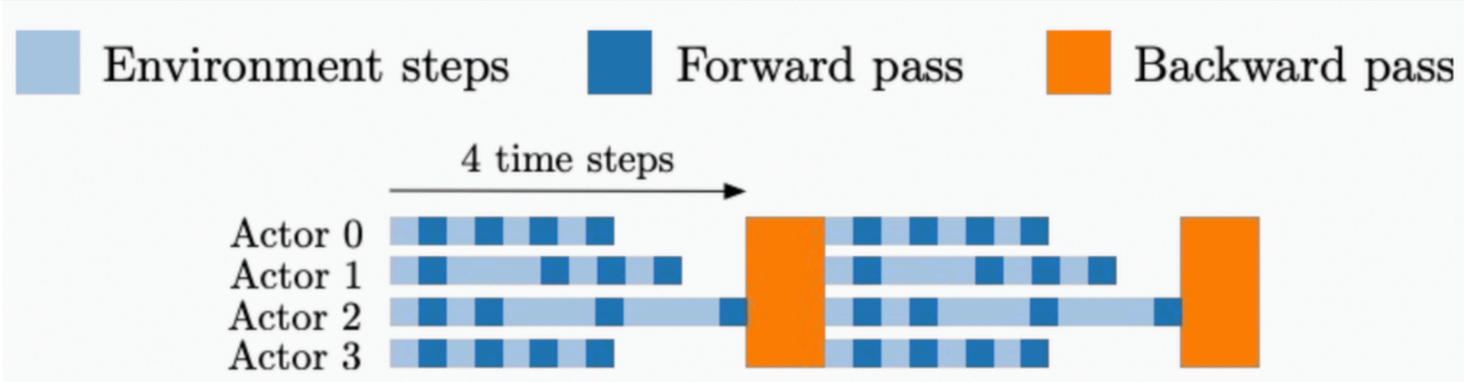
Asynchronous Off-Policy Training

Synchronous Actor-Critic

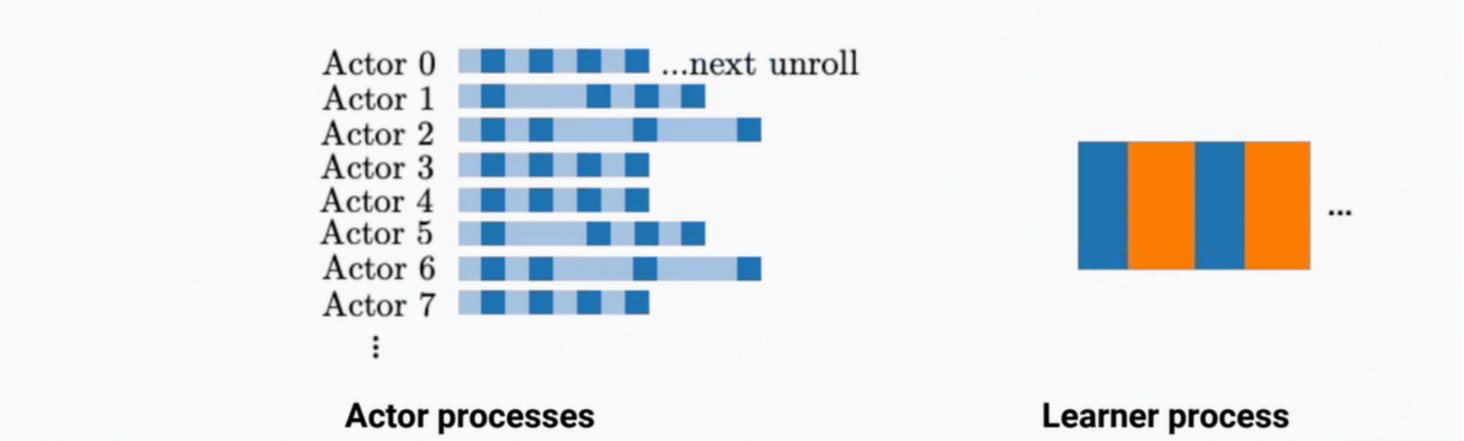


Asynchronous Off-Policy Training

Synchronous Actor-Critic



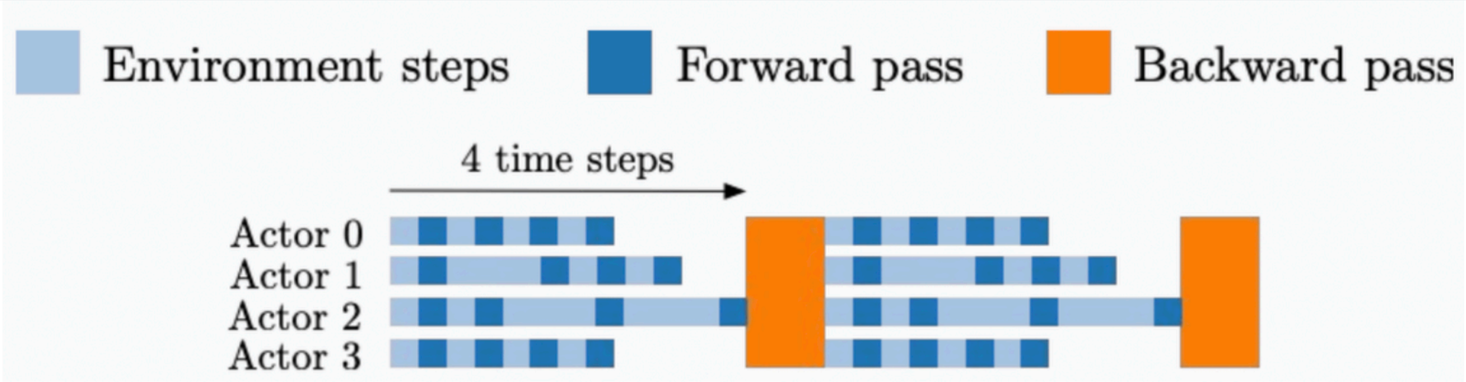
Asynchronous Actor-Critic



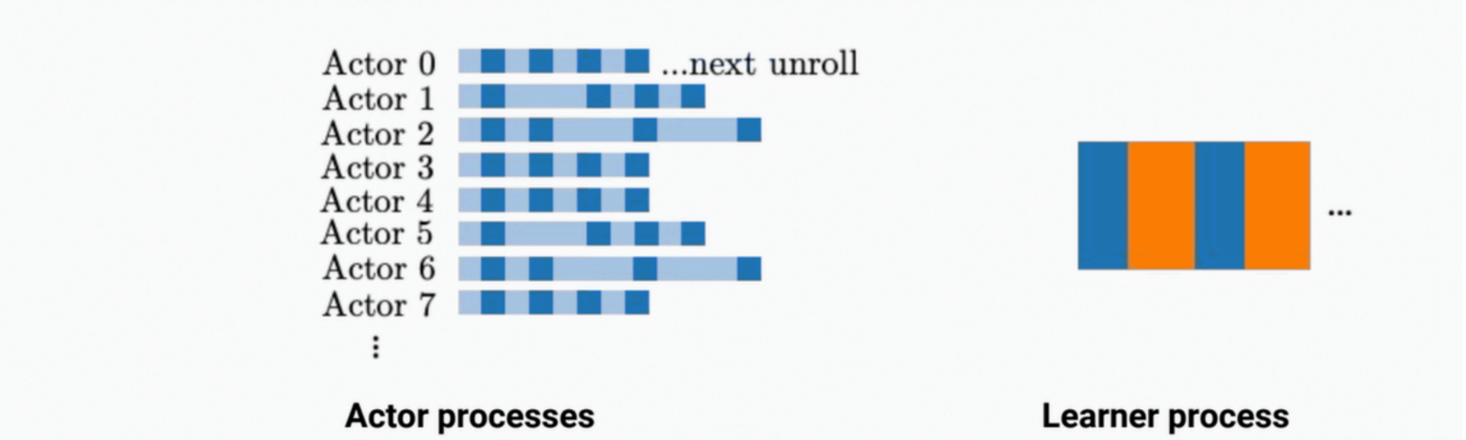
IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures, Espeholt et al.

Asynchronous Off-Policy Training

Synchronous Actor-Critic



Asynchronous Actor-Critic



Problem:

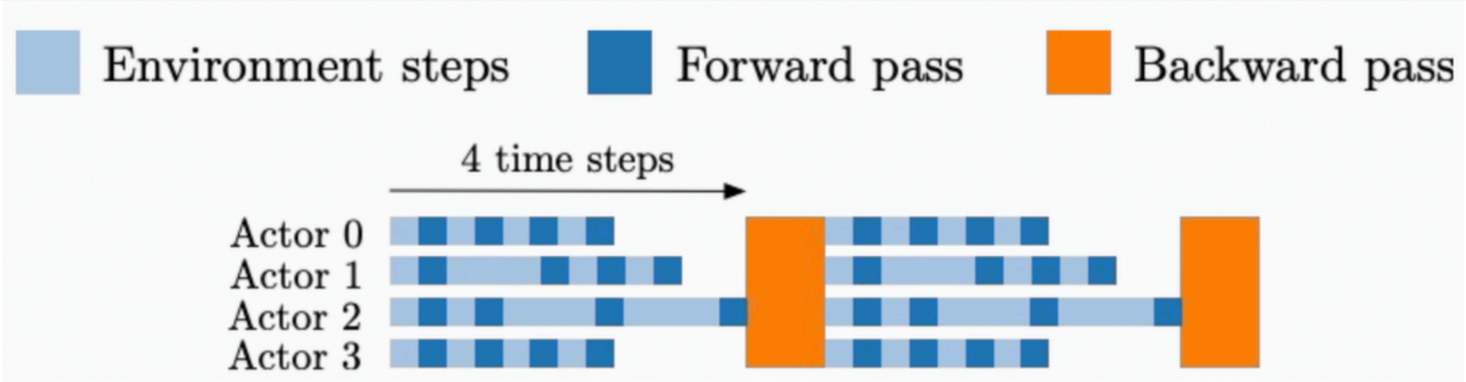
- Actor policy is older than learner policy
- Trajectories are sampled from older policy $\mu(a \mid s)$ instead of target policy $\pi(a \mid s)$.

$$\nabla_\theta E_\pi[R(\tau)] = E_\mu[\nabla_\theta log\, \pi_\theta(\tau)\, V(s)]$$

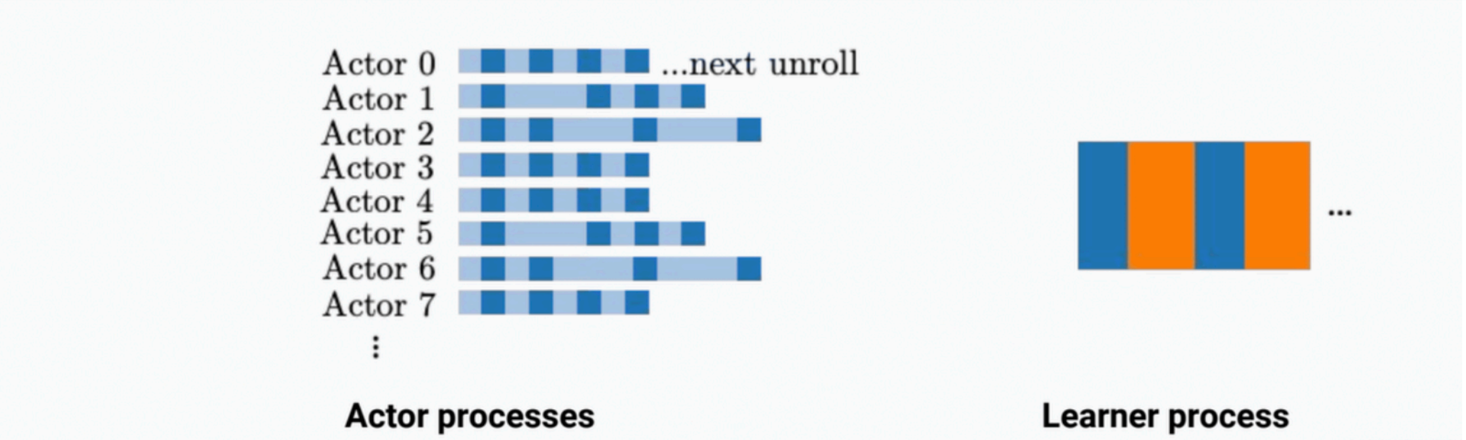
IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures, Espeholt et al.

Asynchronous Off-Policy Training

Synchronous Actor-Critic



Asynchronous Actor-Critic



Problem:

- Actor policy is older than learner policy
- Trajectories are sampled from older policy $\mu(a \mid s)$ instead of target policy $\pi(a \mid s)$.

$$\nabla_\theta E_\pi[R(\tau)] = E_\mu[\nabla_\theta log\, \pi_\theta(\tau)\, V(s)]$$

Fixed by importance sampling

$$\nabla_\theta E_\pi[R(\tau)] = E_\mu[\frac{\pi(\tau)}{\mu(\tau)} \nabla_\theta log\, \pi_\theta(\tau)\, V(s)]$$

IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures, Espeholt et al.

Open Research Questions

State / ACK aggregation:

- Large variance in Inter-ACK arrival time: ~10ms to 200ms
- Cannot act on every single ACK
- RL Agent acts at a fixed clock-rate
- ACK aggregation: RNN, min/max/mean/var, etc.

LSTM over trajectory:

- Systems identification in variable environments

Can we leverage domain-info during training?

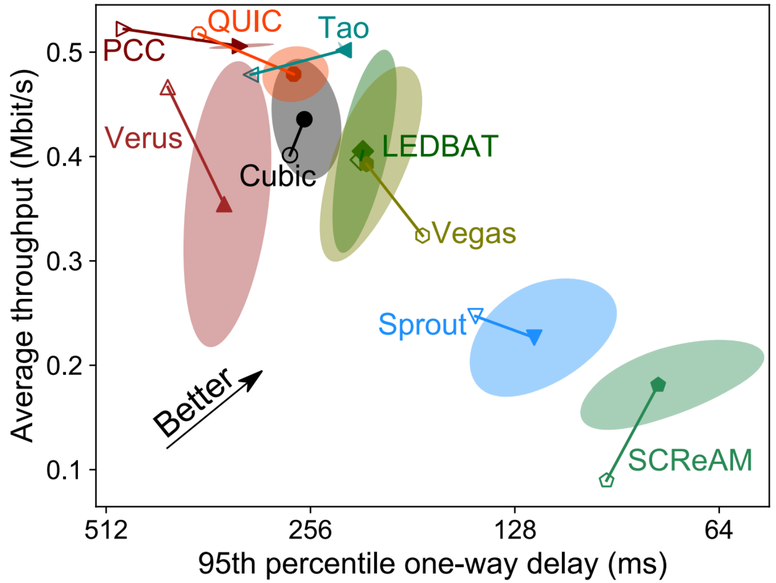
- Improves baseline / critic
- "Zero-out" during testing (not needed for policy)
- Used in environments like StartCraft

Our Environment

I. QUIC Networking Stack (https://github.com/facebookincubator/mvfst):
- Handles Facebook production traffic
- UDP-based
- Application-layer (as opposed to Kernel) congestion control

II. IMPALA:
- Centralized learner on GPU
- Massively parallel actors / environments
- RPC-based interaction with actors / environments

III. Pantheon Network Emulator (https://pantheon.stanford.edu/):
- Tunneled client-server setup
- Bayesian Optimization to calibrate network emulators



(a) Nepal to AWS India (wireless), 1 flow, 10 trials. Mean replication error: 19.1%. P188.

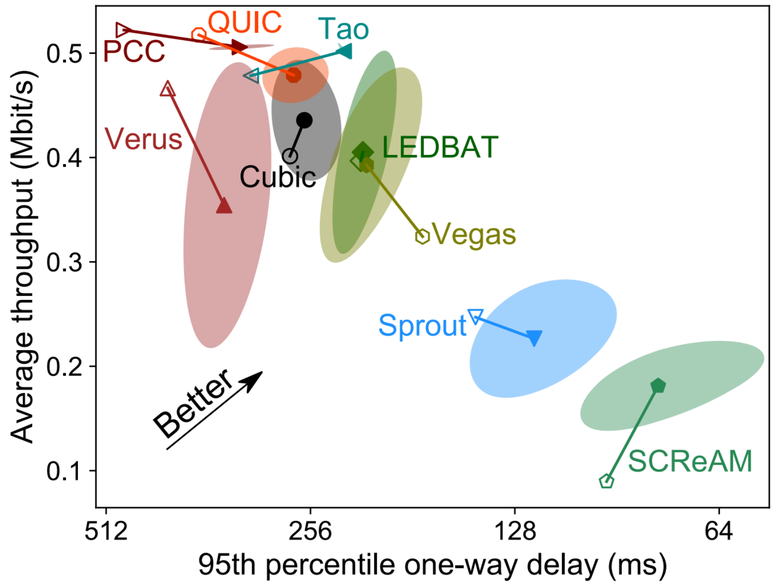
Pantheon: the training ground for Internet congestion-control research, Yan et al.

Our Environment

I. QUIC Networking Stack (https://github.com/facebookincubator/mvfst):
- Handles Facebook production traffic
- UDP-based
- Application-layer (as opposed to Kernel) congestion control

II. IMPALA:
- Centralized learner on GPU
- Massively parallel actors / environments
- RPC-based interaction with actors / environments

III. Pantheon Network Emulator (https://pantheon.stanford.edu/):
- Tunneled client-server setup
- Bayesian Optimization to calibrate network emulators



(a) Nepal to AWS India (wireless), 1 flow, 10 trials. Mean replication error: 19.1%. P188.

Pantheon: the training ground for Internet congestion-control research, Yan et al.

Soon to be open-sourced!

Thank You!