Deep Learning Applications

Syllabus & Reference

Computer vision, Speech Recognition, Natural Language Processing, Decision Making process.

- Reference:
  Deep Learning (Adaptive Computation and Machine Learning series),
  Ian Goodfellow , Yoshua Bengio, Aaron Courville,
  Francis Bach, 2017, MIT Press
-

Large Scale Deep Learning

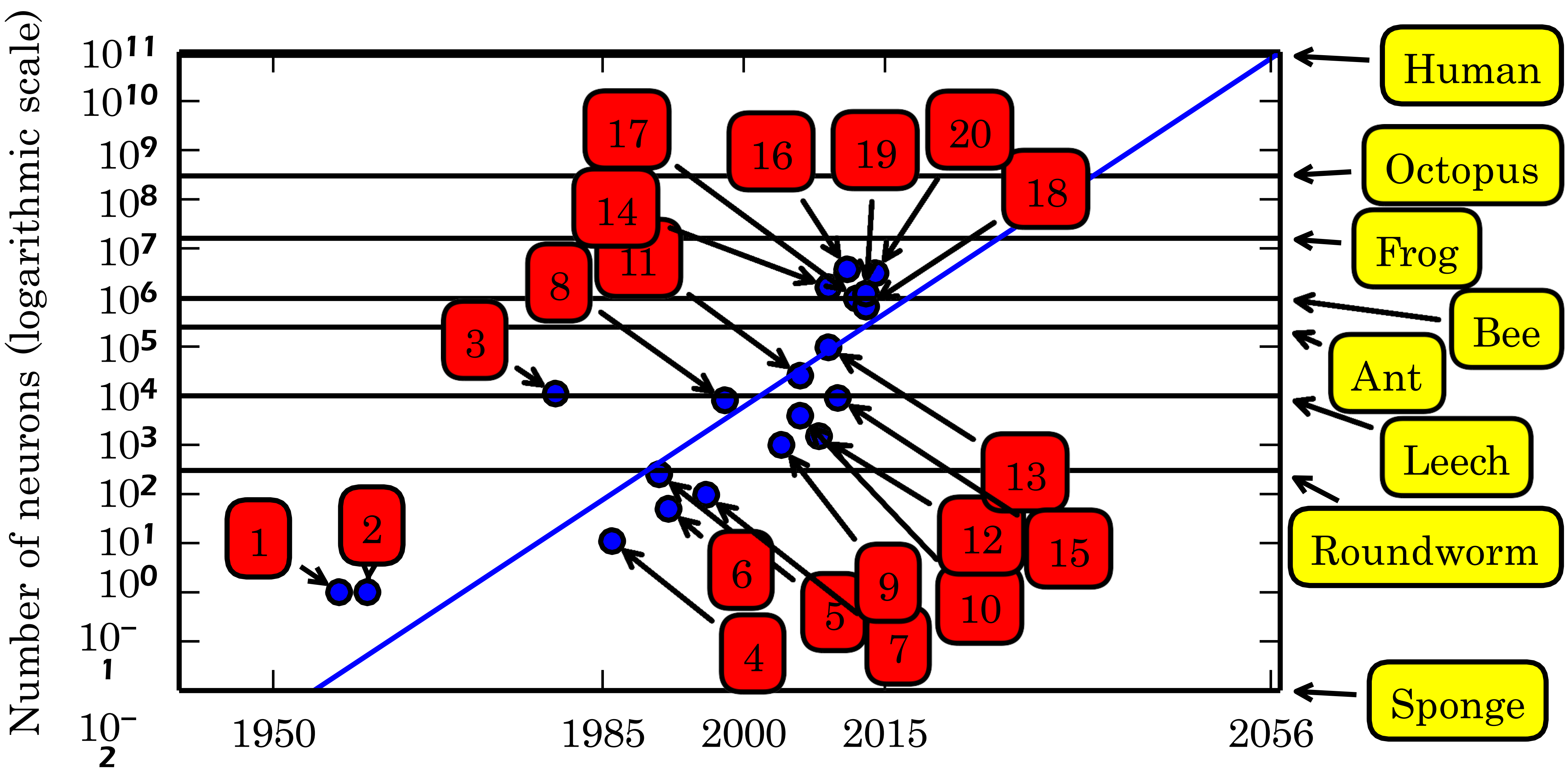


Figure 1.11

Fast Implementations

- CPU
  - Exploit fixed point arithmetic in CPU families where this offers a speedup
  - Cache-friendly implementations
- GPU
  - High memory bandwidth
  - No cache
  - Warps must be synchronized
- TPU
  - Similar to GPU in many respects but faster
  - Often requires larger batch size
  - Sometimes requires reduced precision

Distributed Implementations

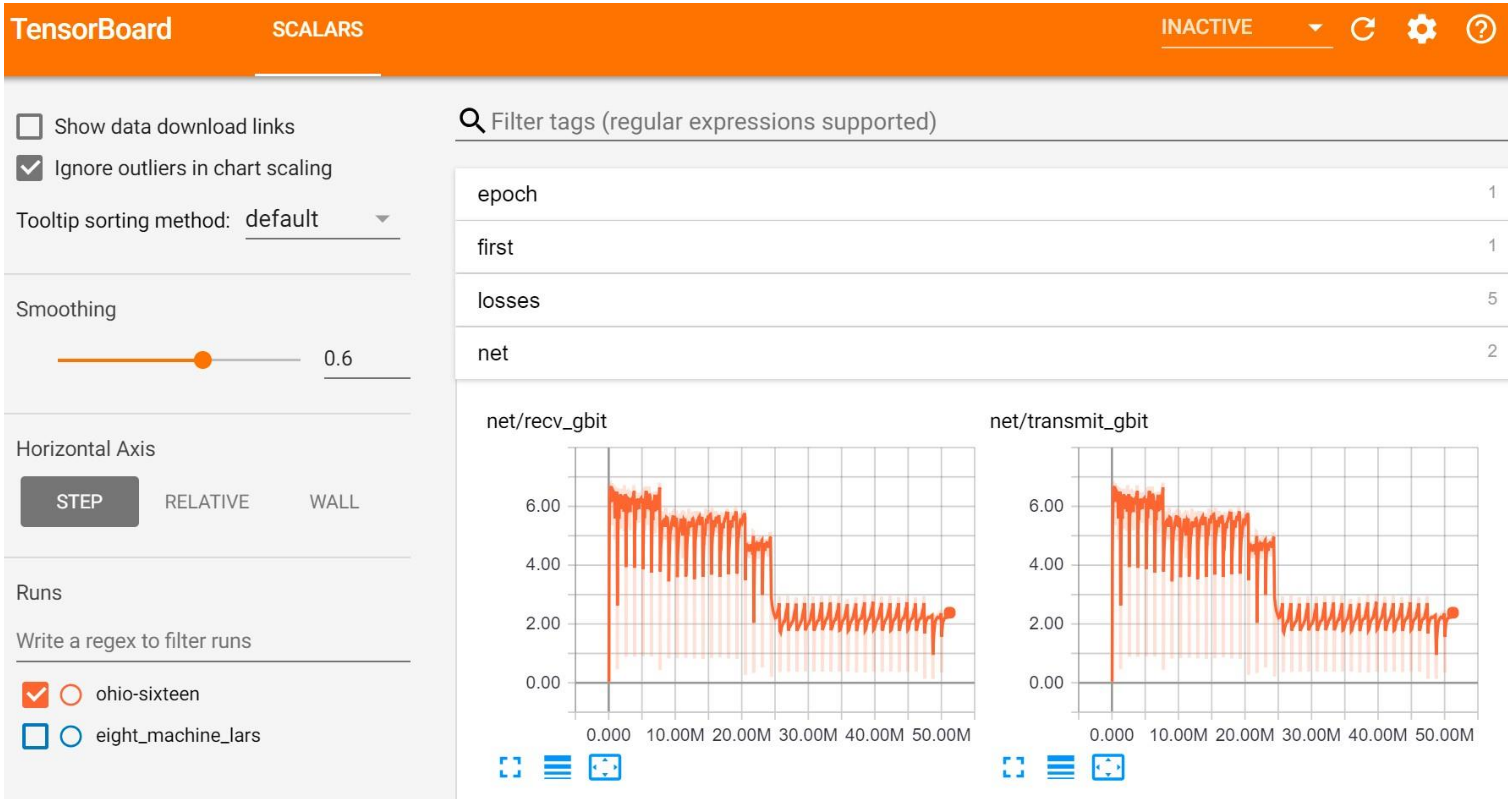
- Distributed
  - Multi-GPU
  - Multi-machine
- Model parallelism
- Data parallelism
  - Trivial at test time
  - Synchronous or asynchronous SGD at train time

Synchronous SGD

```
# Calculate the gradients for each model tower.
tower_grads = []
with tf.variable_scope(tf.get_variable_scope()):
  for i in xrange(FLAGS.num_gpus):
    with tf.device('/gpu:%d' % i):
      with tf.name_scope('%s_%d' % (cifar10.TOWER_NAME, i)) as scope:
        # Dequeues one batch for the GPU
        image_batch, label_batch = batch_queue.dequeue()
        # Calculate the loss for one tower of the CIFAR model. This function
        # constructs the entire CIFAR model but shares the variables across
        # all towers.
        loss = tower_loss(scope, image_batch, label_batch)

        # Reuse variables for the next tower.
        tf.get_variable_scope().reuse_variables()

        # Calculate the gradients for the batch of data on this CIFAR tower.
        grads = opt.compute_gradients(loss)

        # Keep track of the gradients across all towers.
        tower_grads.append(grads)

# We must calculate the mean of each gradient. Note that this is the
# synchronization point across all towers.
grads = average_gradients(tower_grads)
```

TensorFlow tutorial

Example: ImageNet in 18 minutes for $40

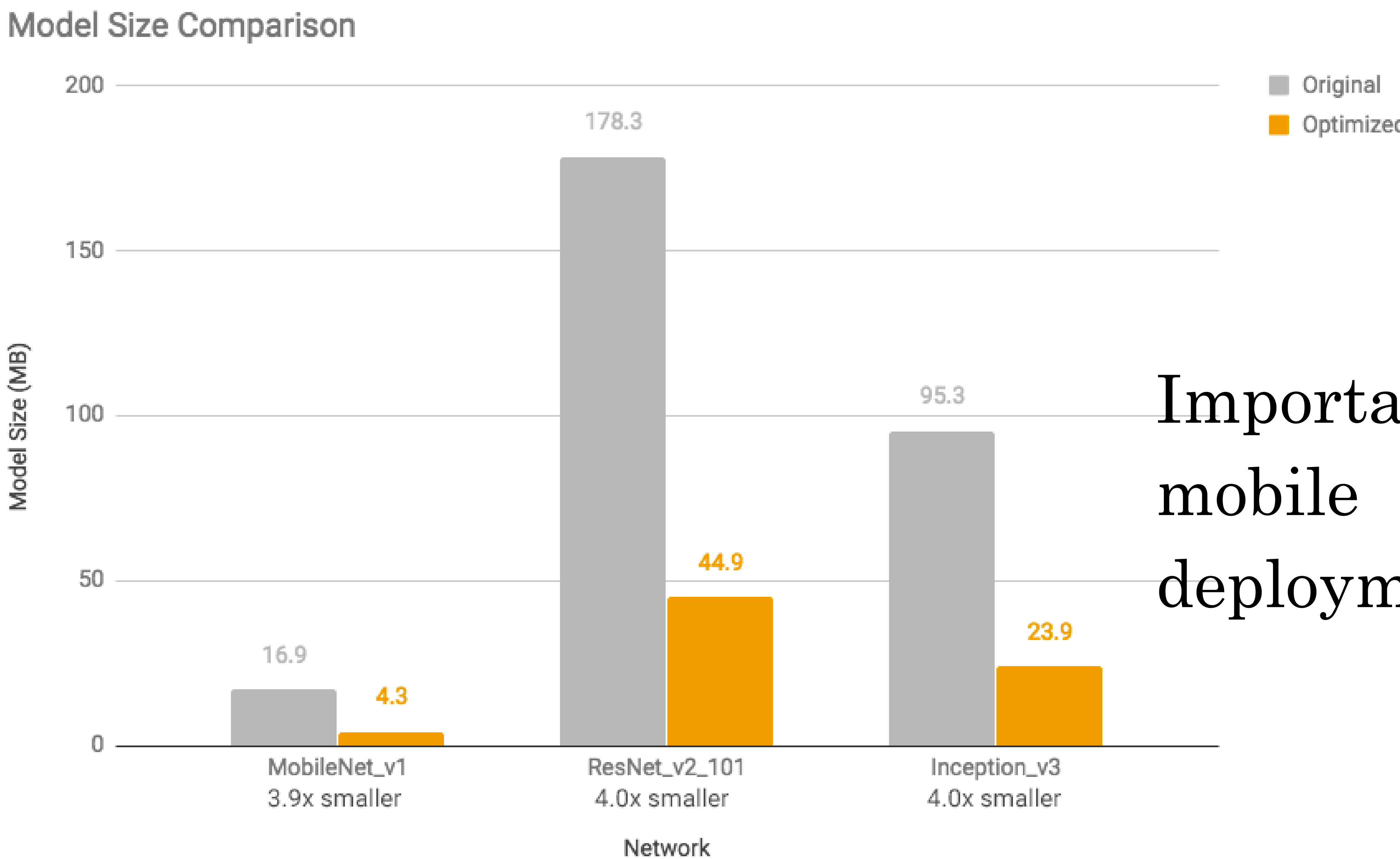


Blog post

Model Compression

- Large models often have lower test error
  - Very large model trained with dropout
  - Ensemble of many models
- Want small model for low resource use at test time
- Train a small model to mimic the large one
  - Obtains better test error than directly training a small model

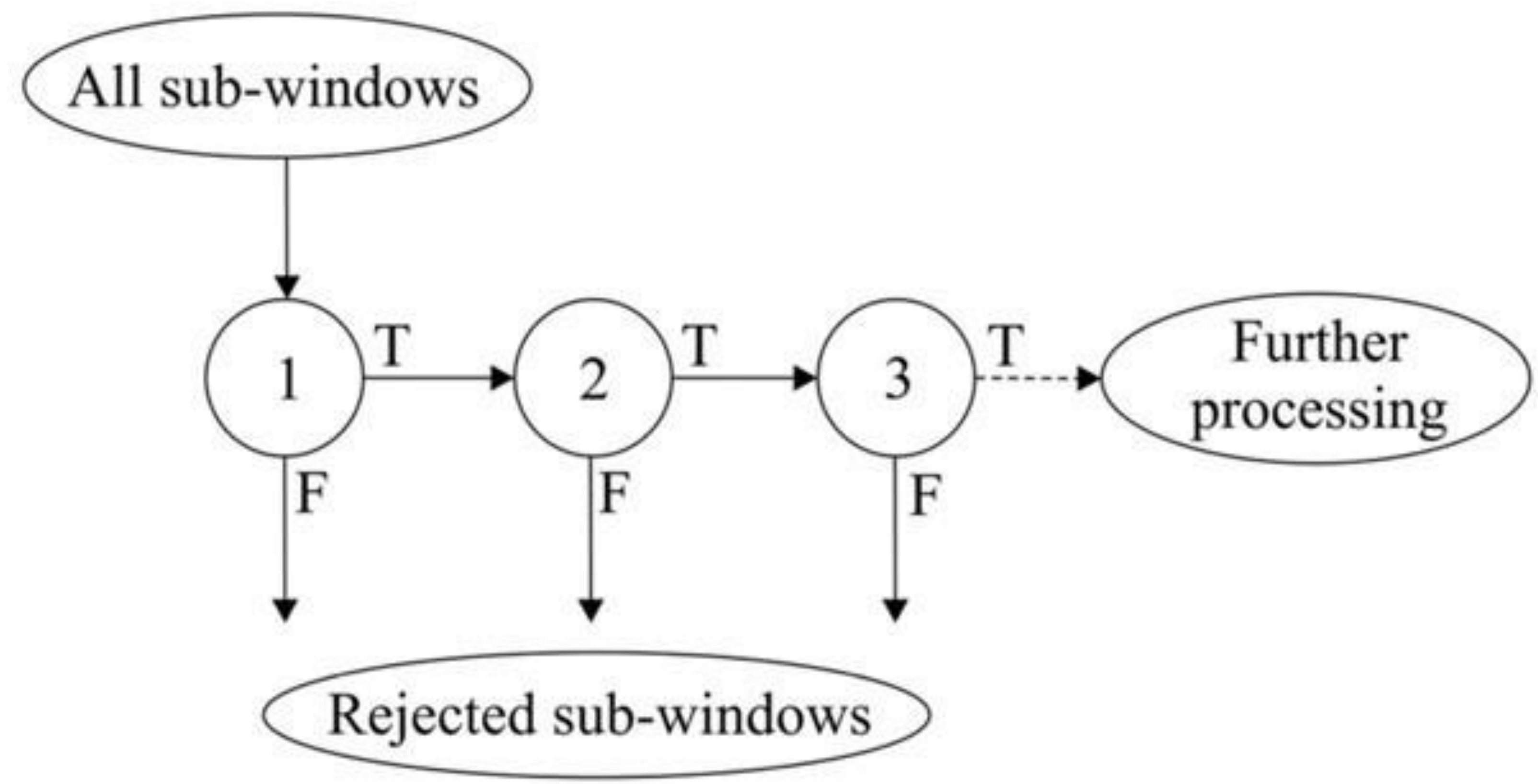
Quantization



Important for mobile deployment

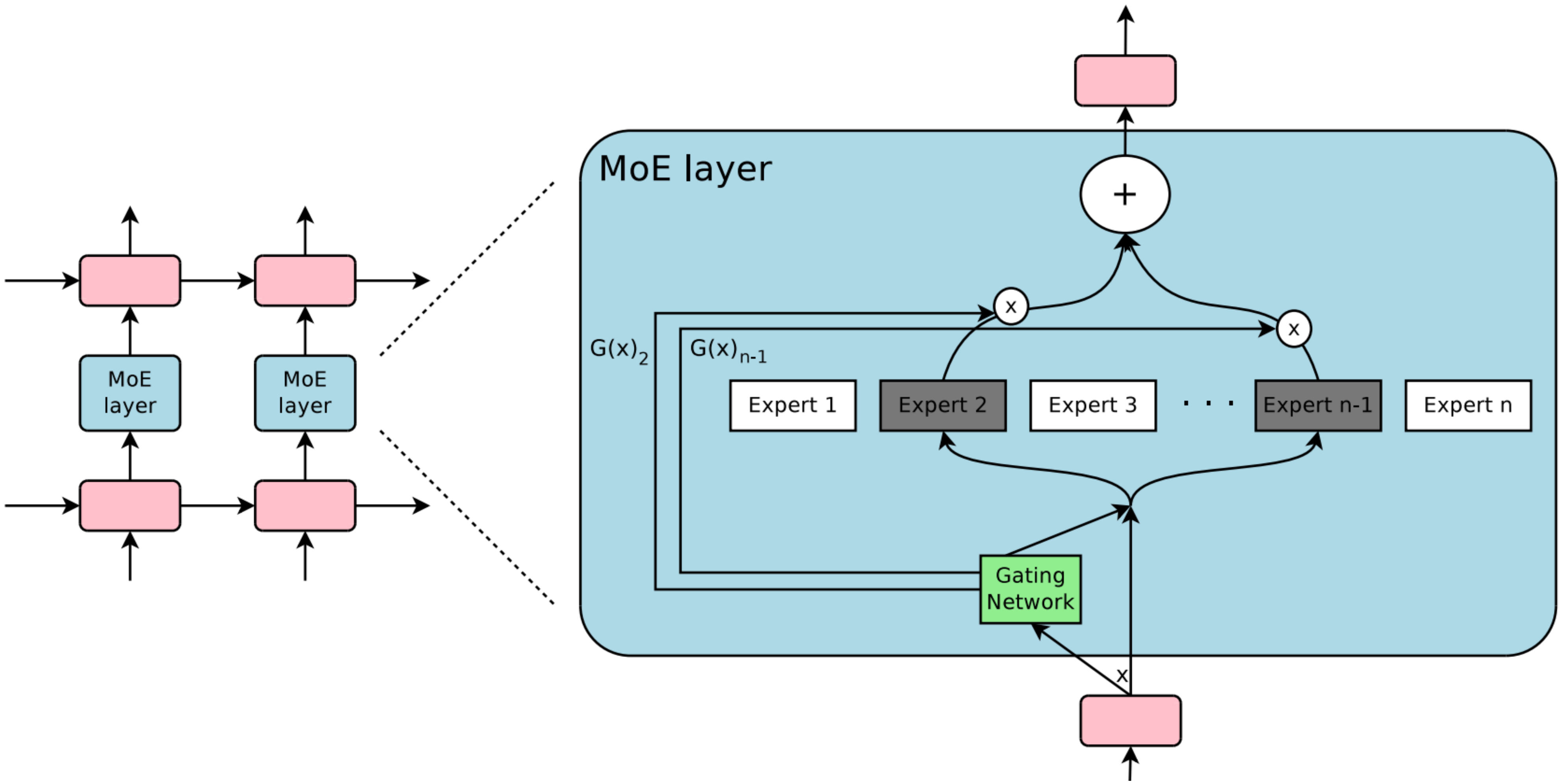
(TensorFlow Lite)

Dynamic Structure: Cascades



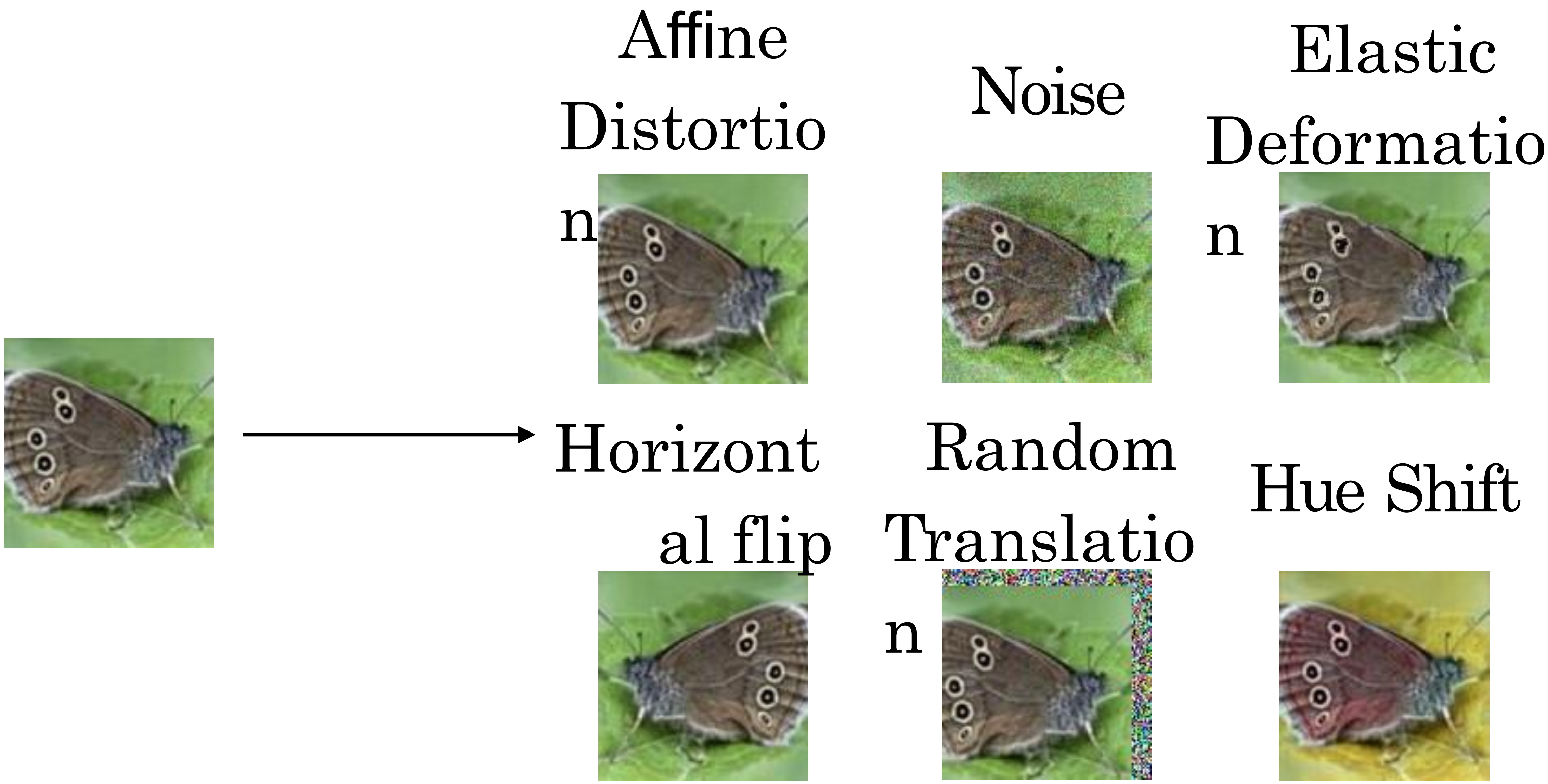
(Viola and Jones, 2001)

Dynamic Structure



<u>Outrageously Large Neural Networks</u>

Dataset Augmentation for Computer Vision



Generative Modeling: Sample Generation

Training Data (CelebA)

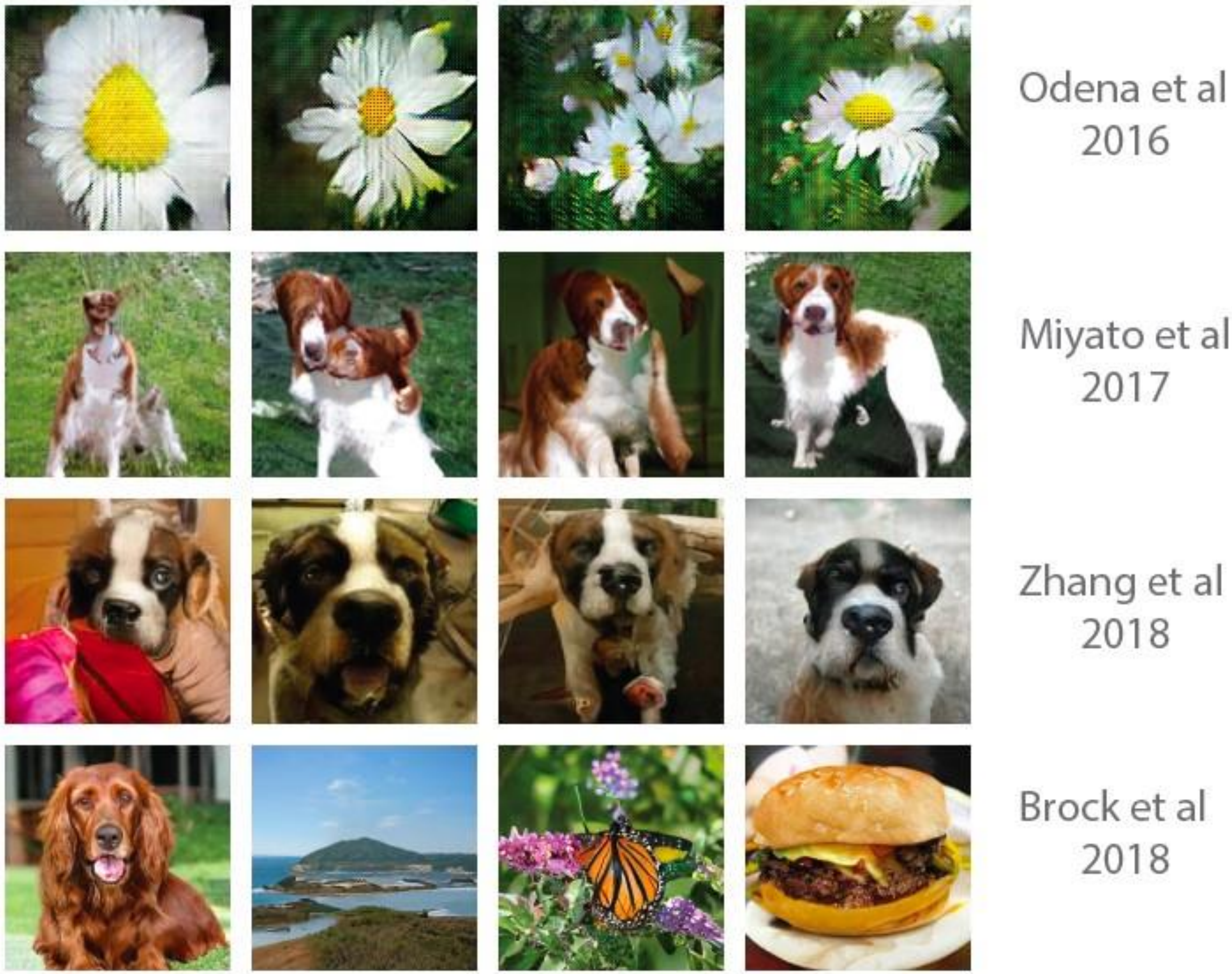
Sample Generator (Karras et al, 2017)

Covered in Part III

Underlies many graphics and speech

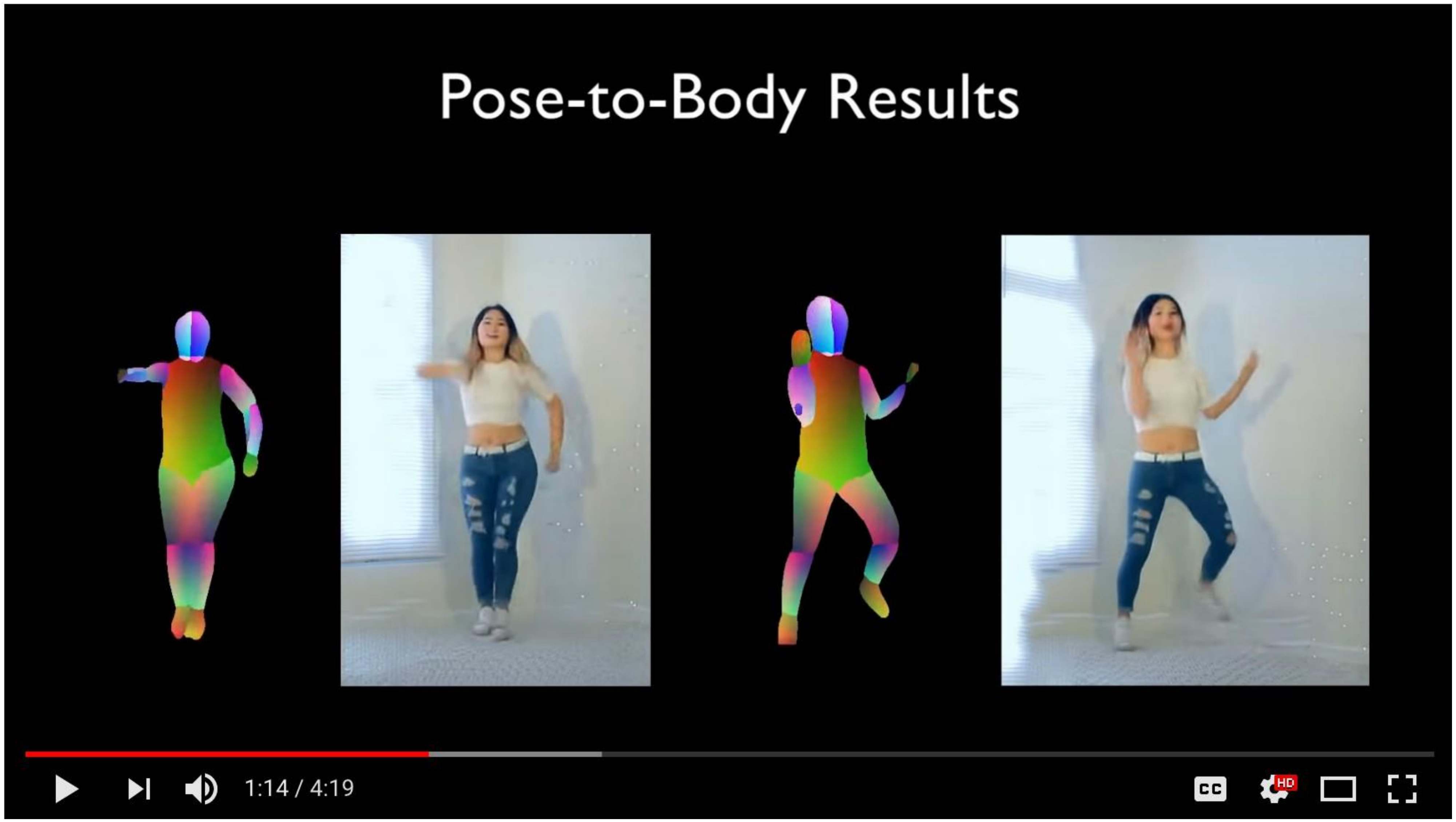
Progressed rapidly after the book was written

Graphics



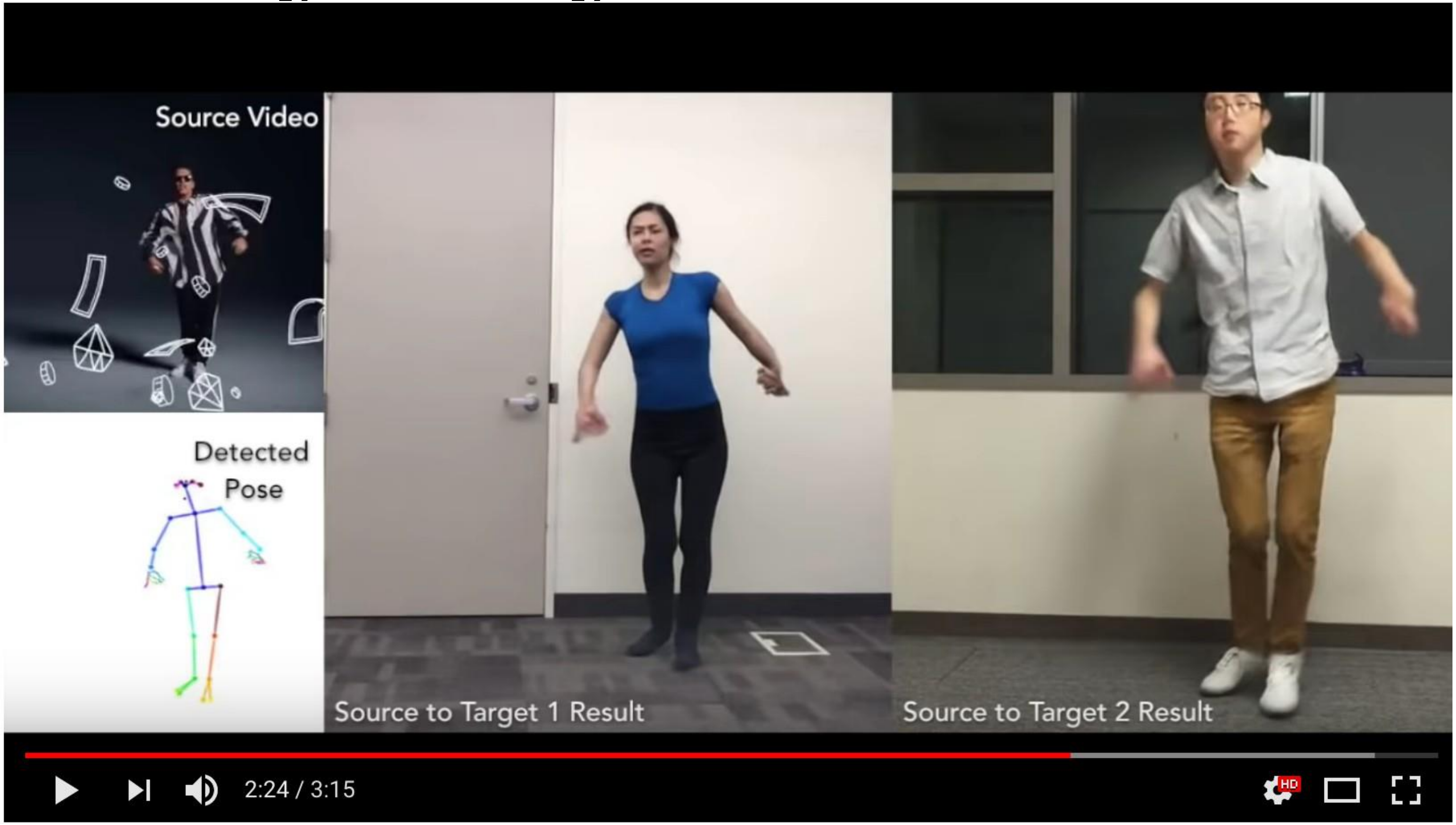
(Table by Augustus Odena)

Video Generation



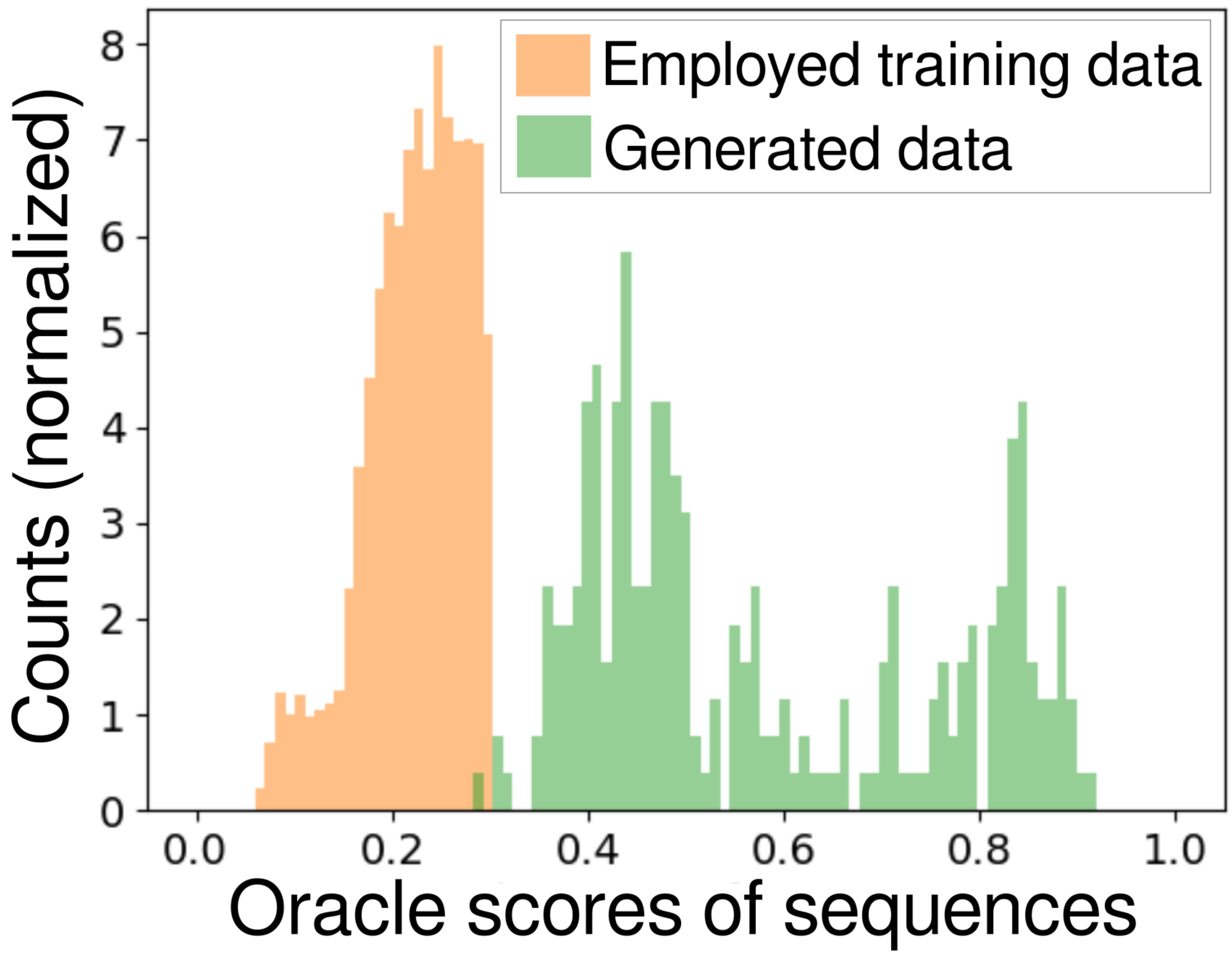
(Wang et al, 2018)

Everybody Dance Now!



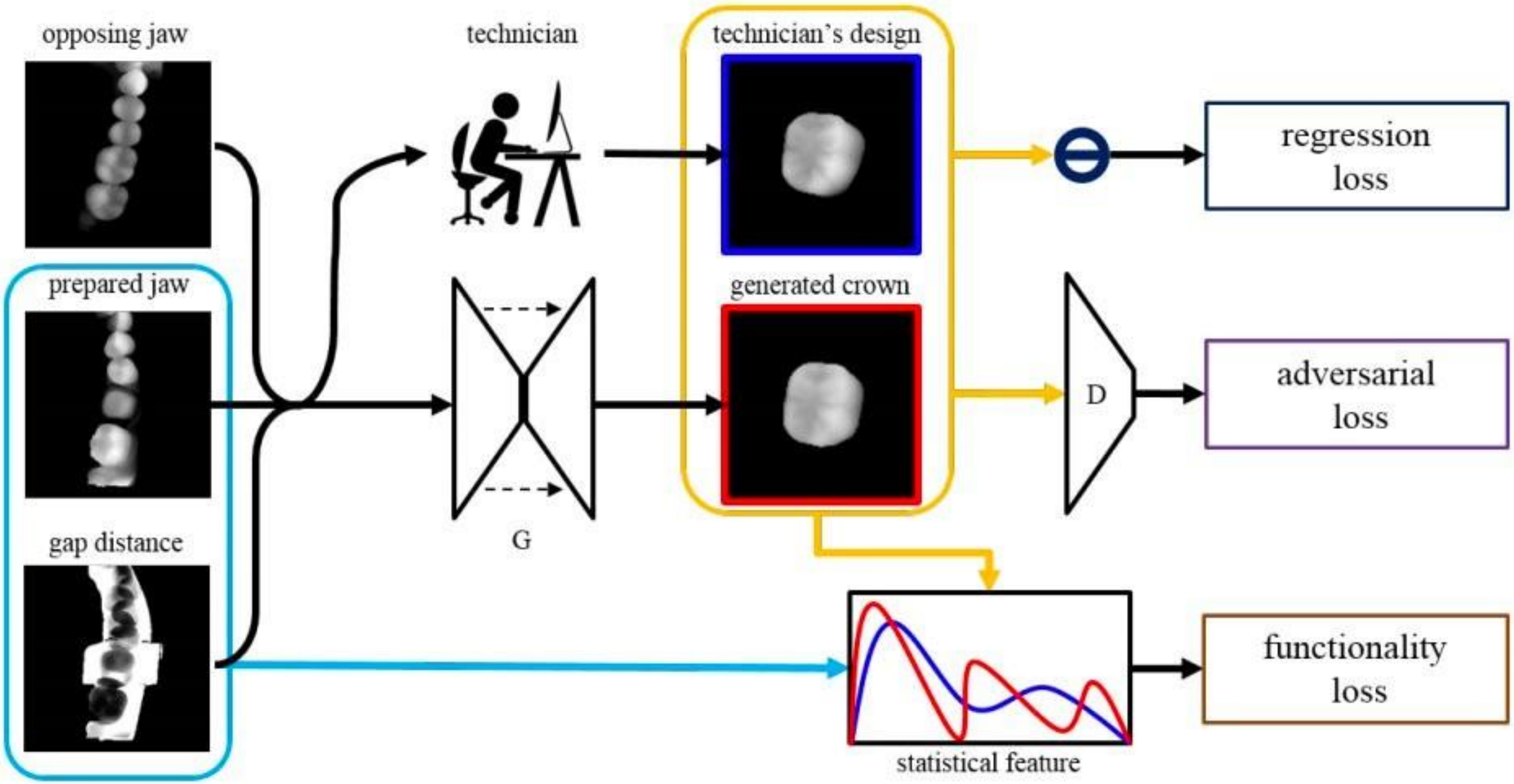
(Chan et al 2018)

Model-Based Optimization



(Killoran et al, 2017)

Designing Physical Objects



(Hwang et al 2018)

Attention Mechanisms

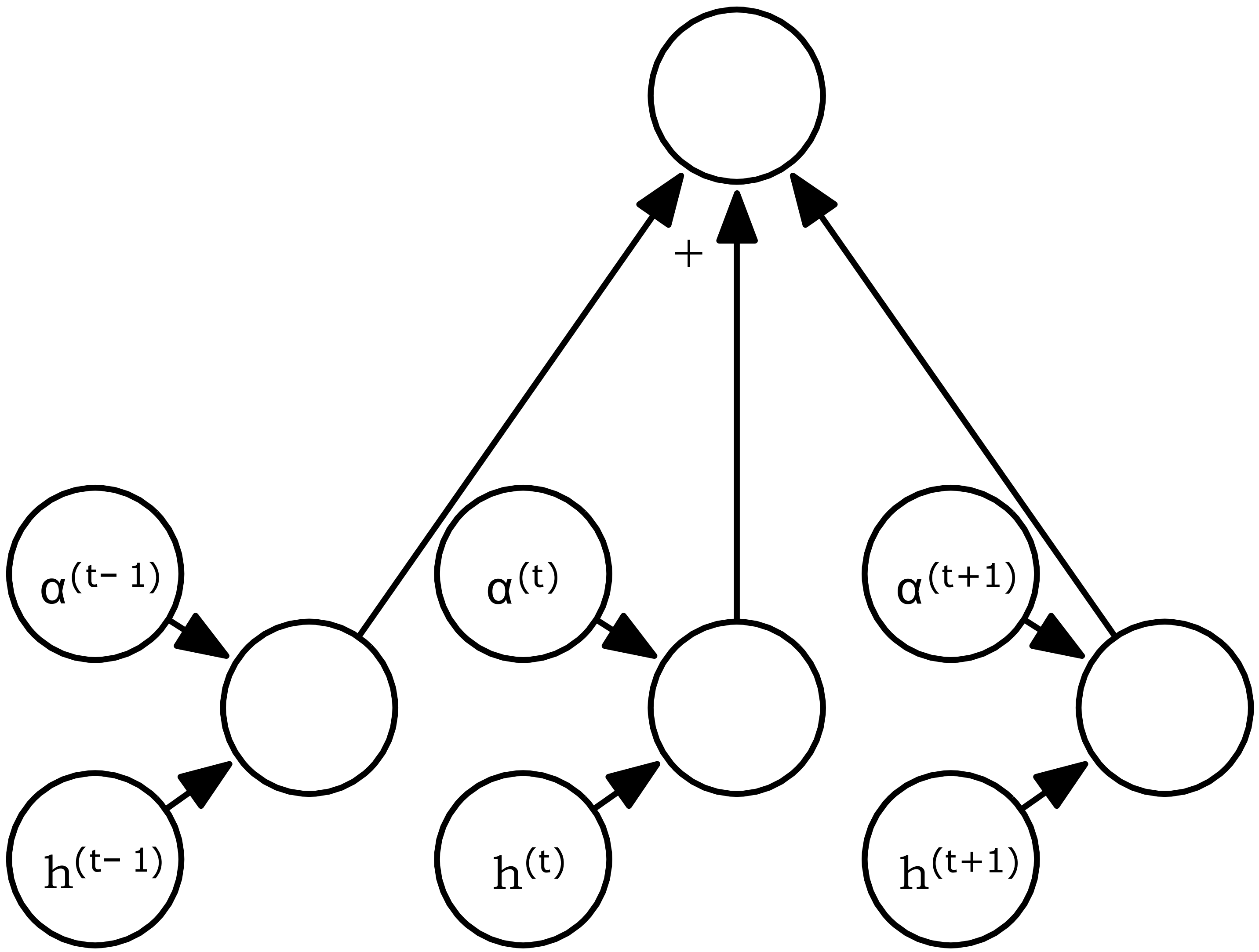


Figure 12.6

Important in many vision, speech, and NLP applications

Improved rapidly after the book was written

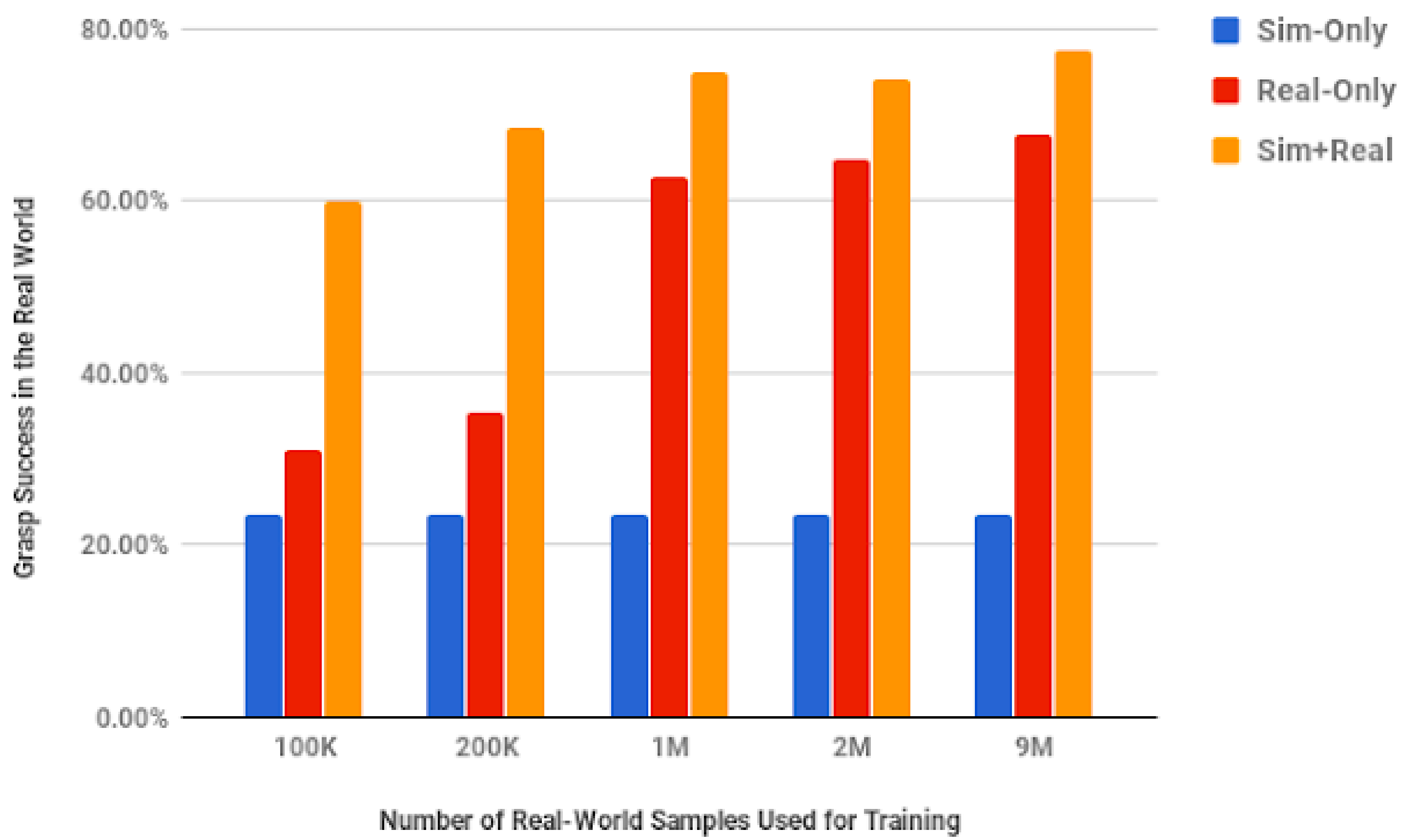
Attention for Images

Attention mechanism from Wang et al 2018

Image model from Zhang et al 2018

Generating Training Data

(Bousmalis et al, 2017)



(Bousmalis et al, 2017)

Natural Language Processing

- An important predecessor to deep NLP is the family of models based on *n*-grams:

$$P(x_1, \ldots, x_\tau) = P(x_1, \ldots, x_{n-1}) \prod_{t=n}^{\tau} P(x_t \mid x_{t-n+1}, \ldots, x_{t-1}). \tag{12.5}$$

$$P(\text{THE DOG RAN AWAY}) = P_3(\text{THE DOG RAN})P_3(\text{DOG RAN AWAY})/P_2(\text{DOG RAN}). \tag{12.7}$$

Improve with:

-Smoothing

-Backoff

-Word categories

Word Embeddings in Neural Language Models

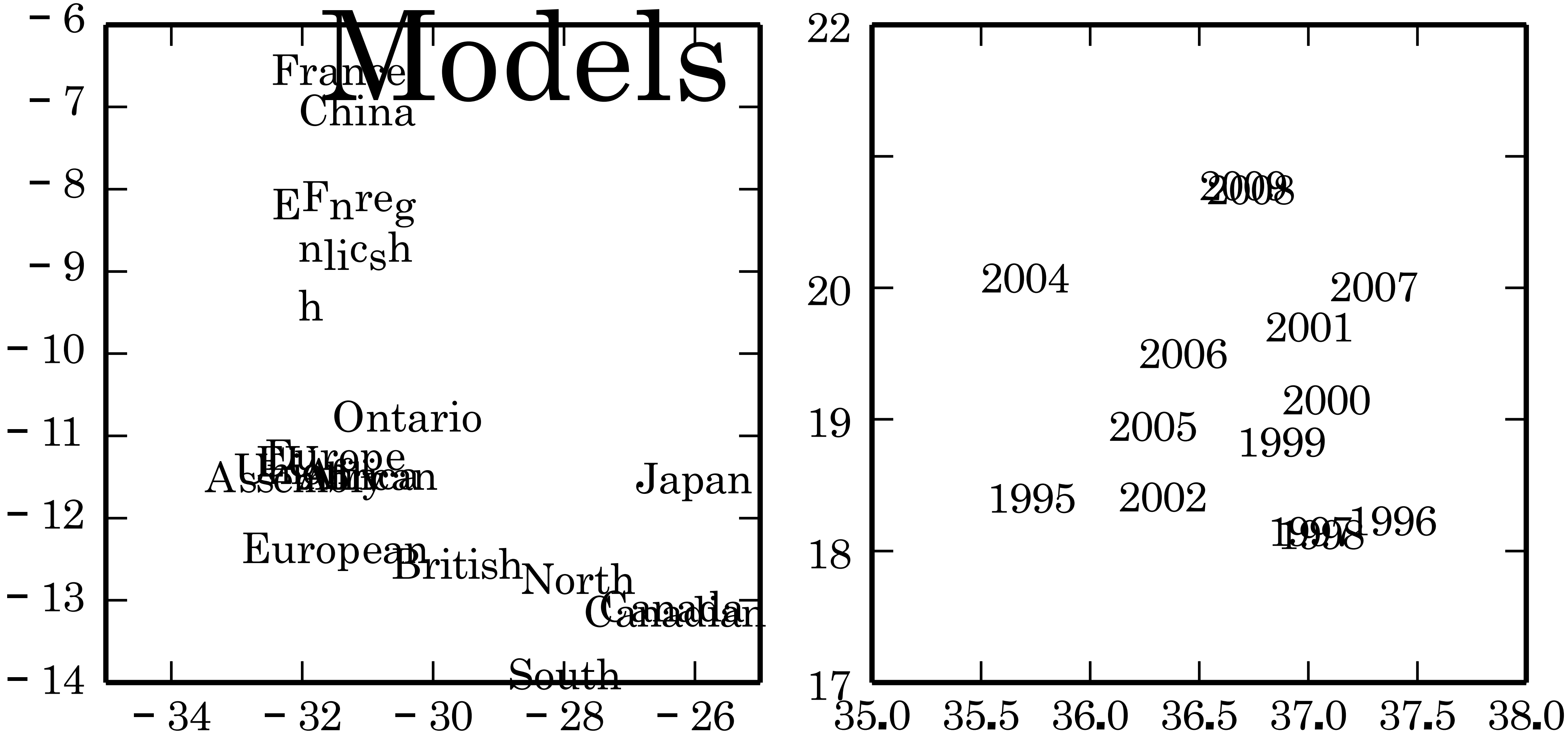


Figure 12.3

High-Dimensional Output Layers for Large Vocabularies

- Short list
- Hierarchical softmax
- Importance sampling
- Noise contrastive estimation

A Hierarchy of Words and Word Categories

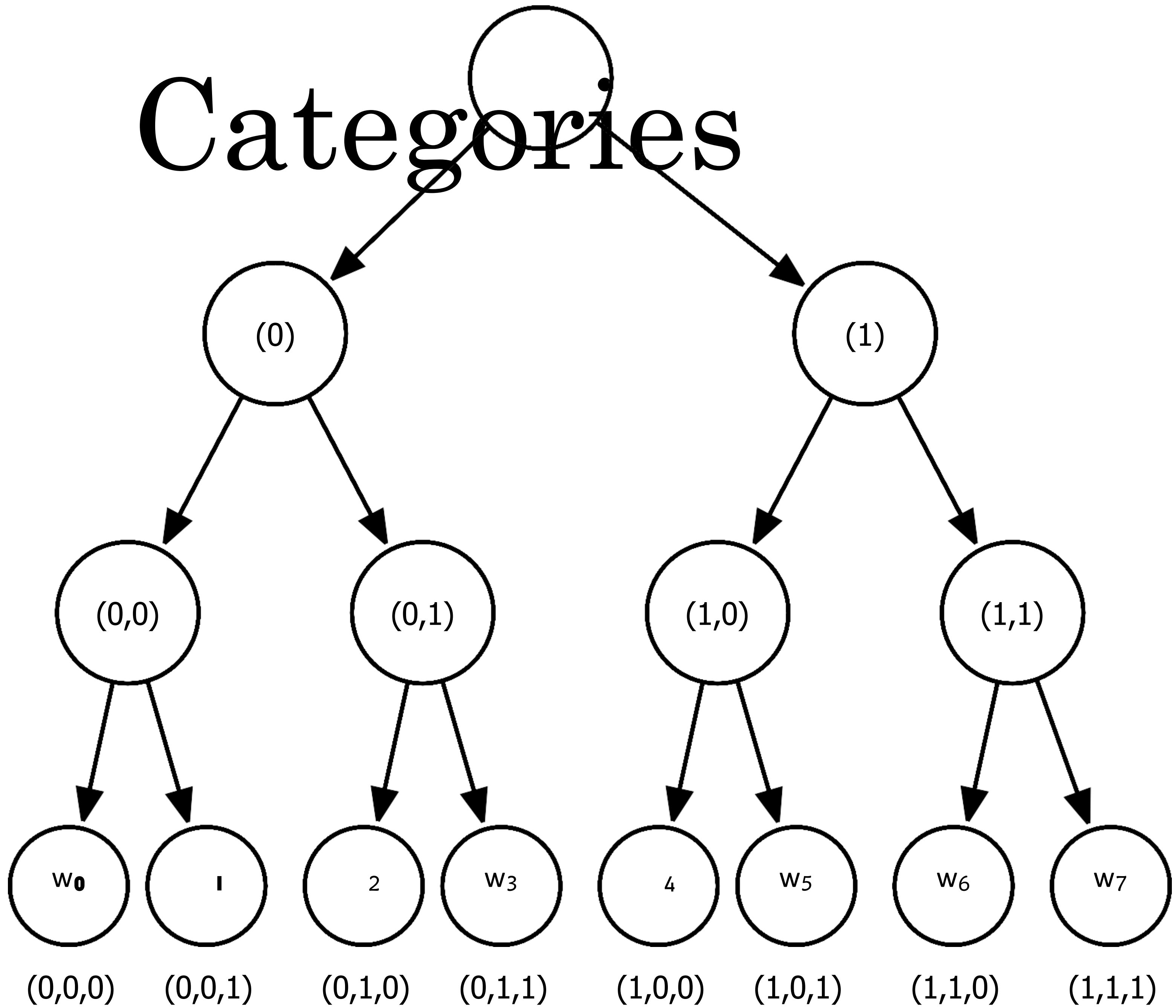


Figure 12.4

Neural Machine Translation

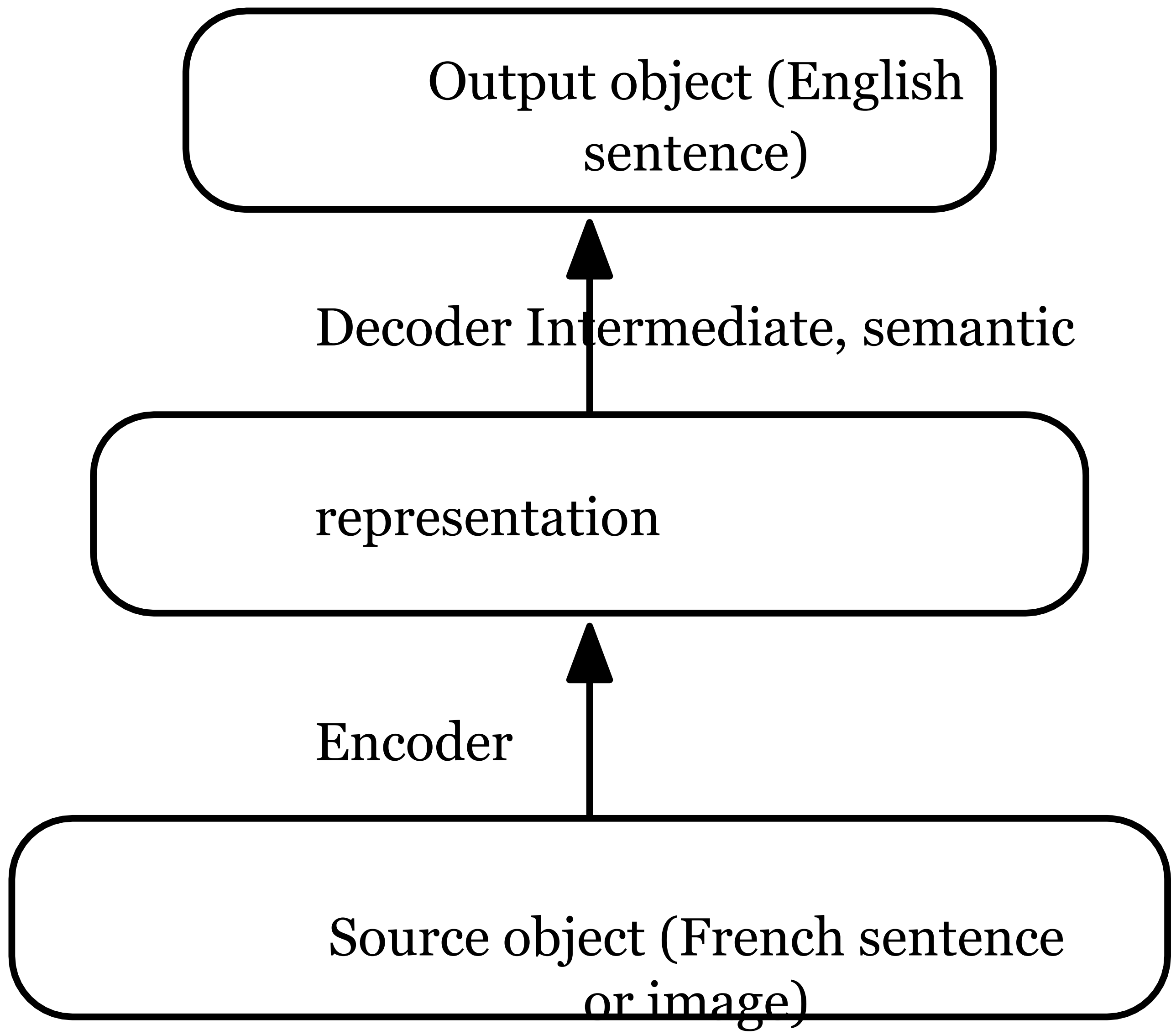
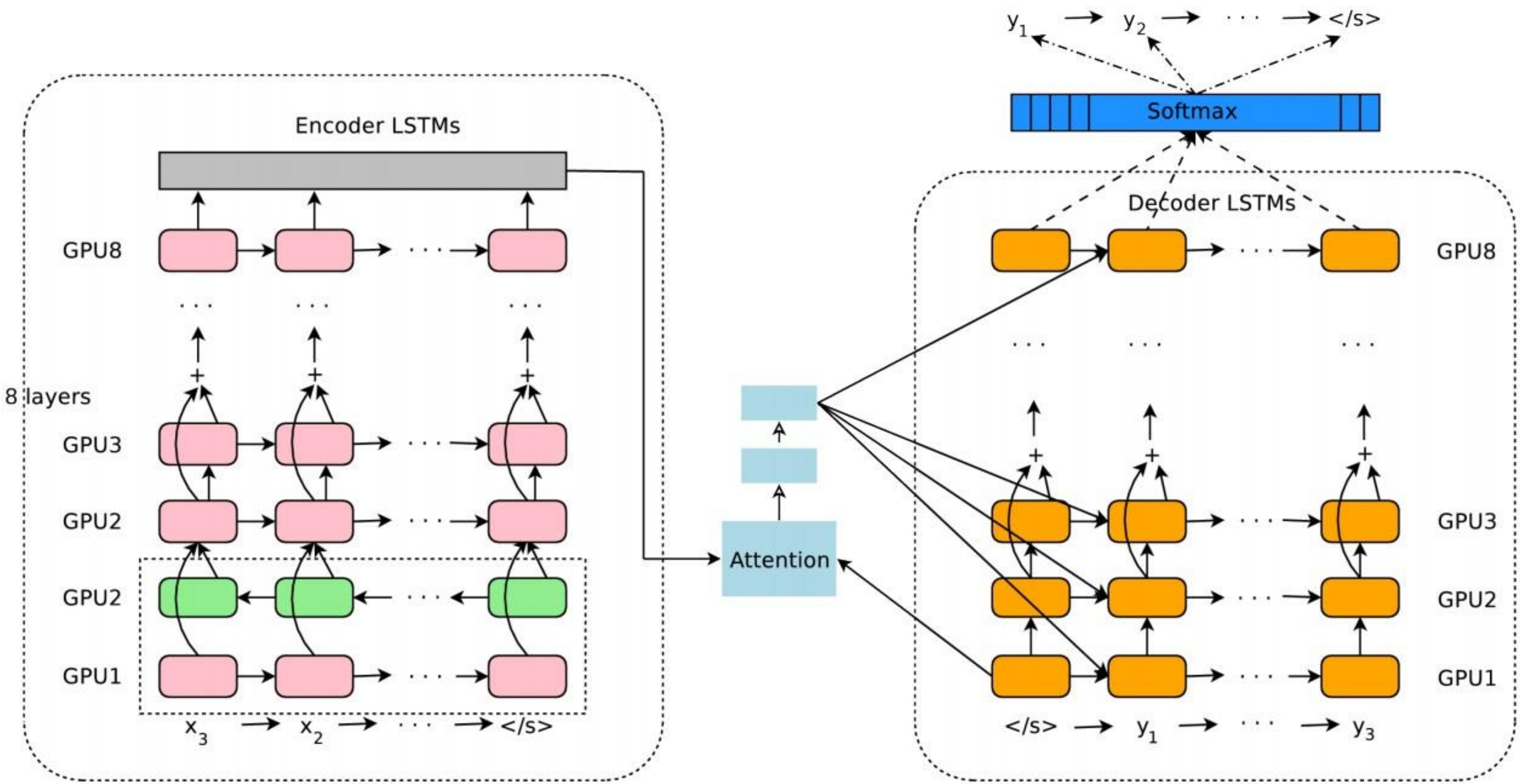


Figure 12.5

Google Neural Machine Translation



Wu et al 2016

Speech Recognition

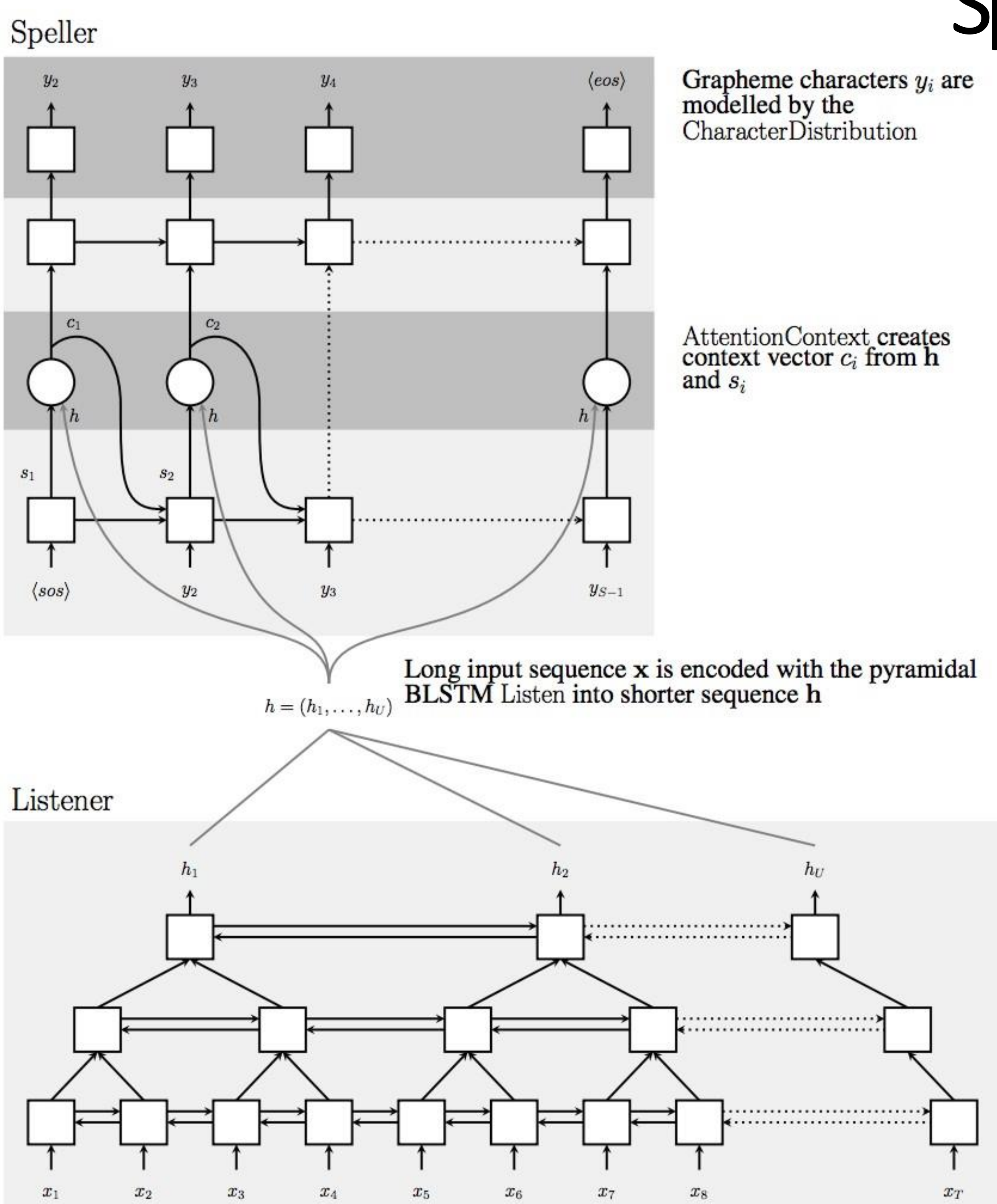
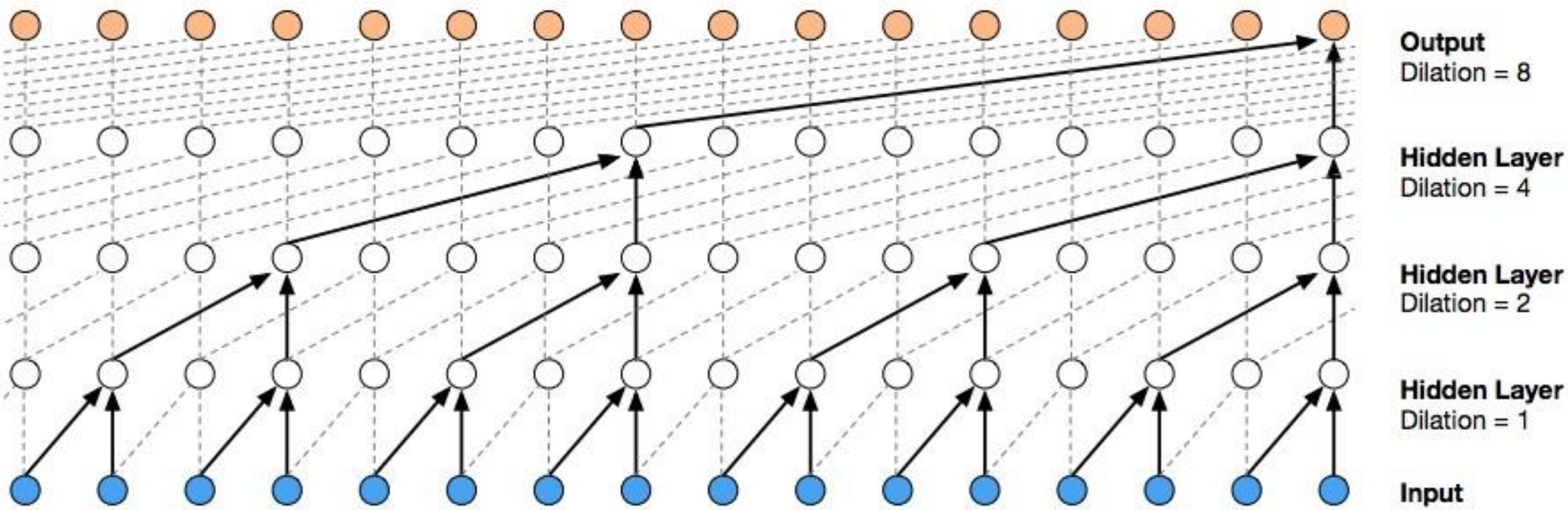


Figure 1: Listen, Attend and Spell (LAS) model: the listener is a pyramidal BLSTM encoding our input sequence $\mathbf{x}$ into high level features $\mathbf{h}$, the speller is an attention-based decoder generating the $\mathbf{y}$ characters from $\mathbf{h}$.

Current speech recognition is based on seq2seq with attention

Graphic from "Listen, Attend, and Spell"
Chan et al 2015

Speech Synthesis



WaveNet
(van den Oord et al, 2016)

Deep RL for Atari game playing

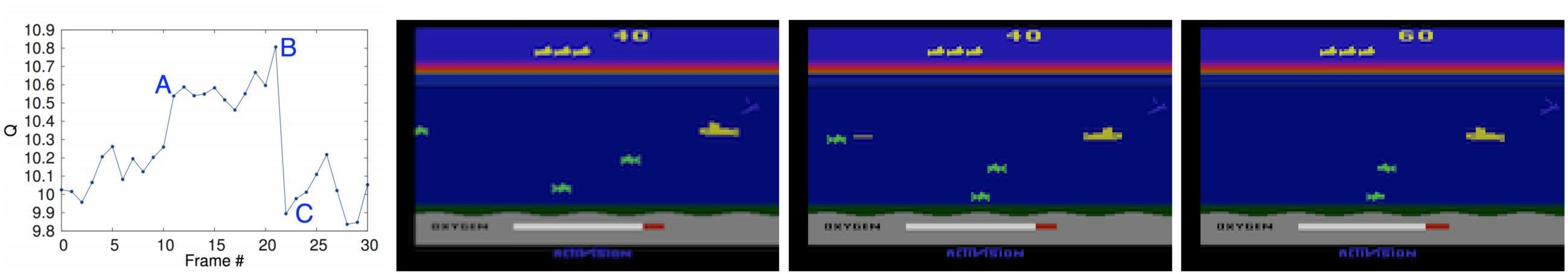


Figure 3: The leftmost plot shows the predicted value function for a 30 frame segment of the game Seaquest. The three screenshots correspond to the frames labeled by A, B, and C respectively.

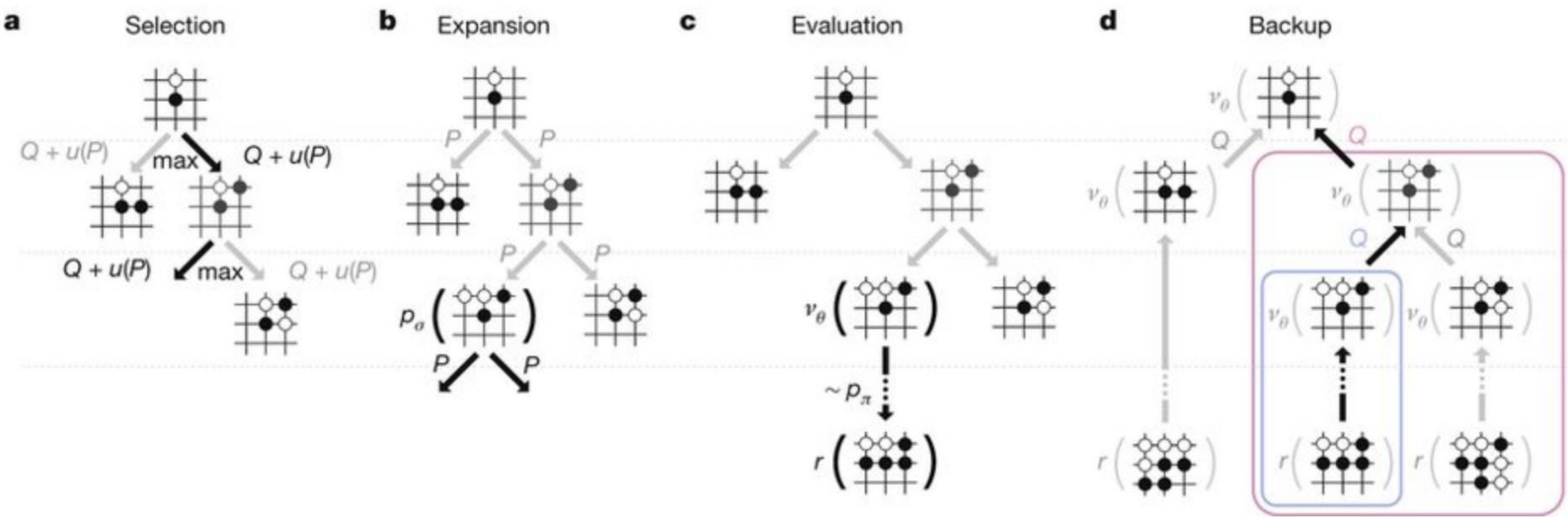
(Mnih et al 2013)

Convolutional network estimates the value function (future rewards) used to guide the game-playing agent.

(Note: deep RL didn't really exist when we started the book, became a success while we were writing it, extremely hot topic by the time the book was printed)

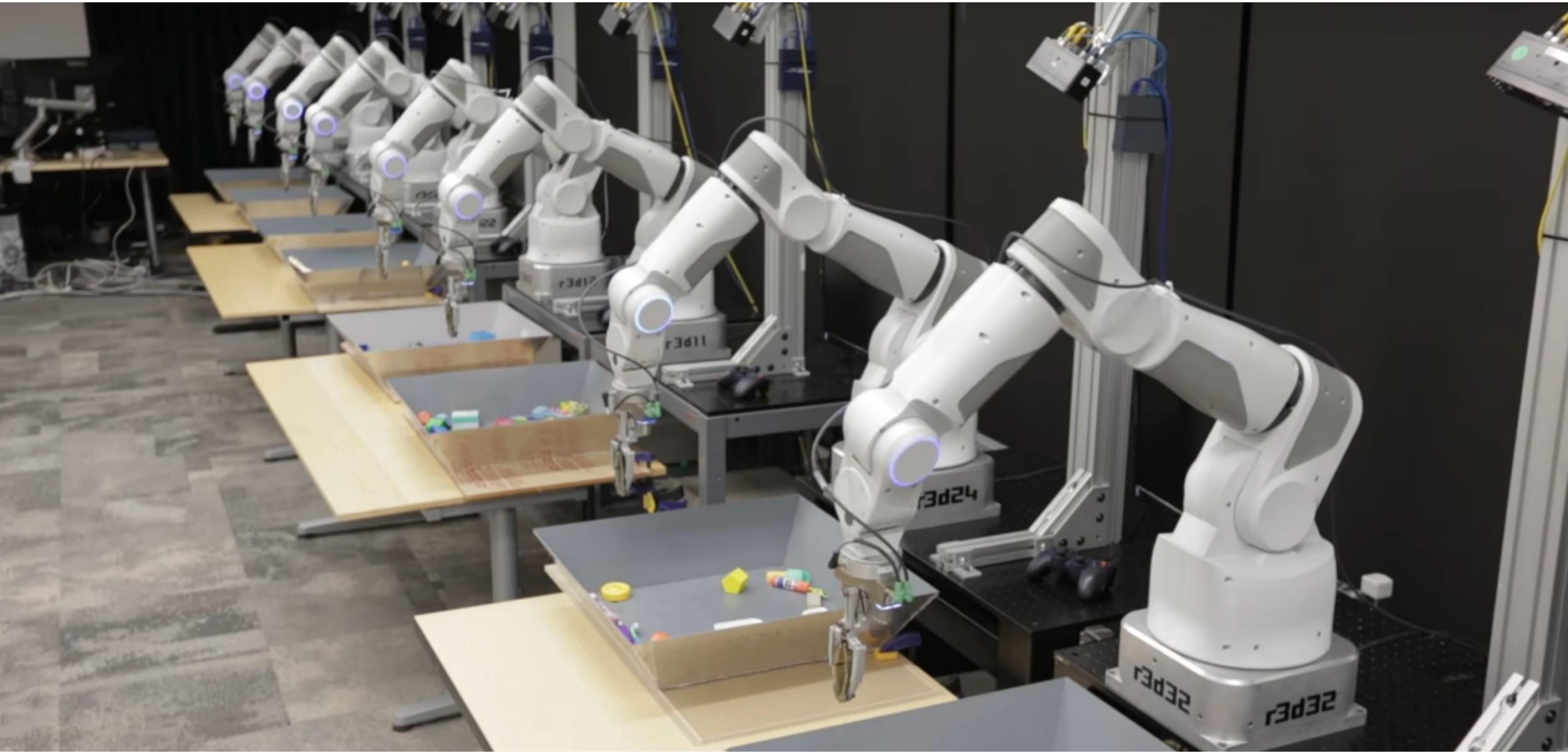
Superhuman Go Performance

Monte Carlo tree search, with convolutional networks for value function and policy

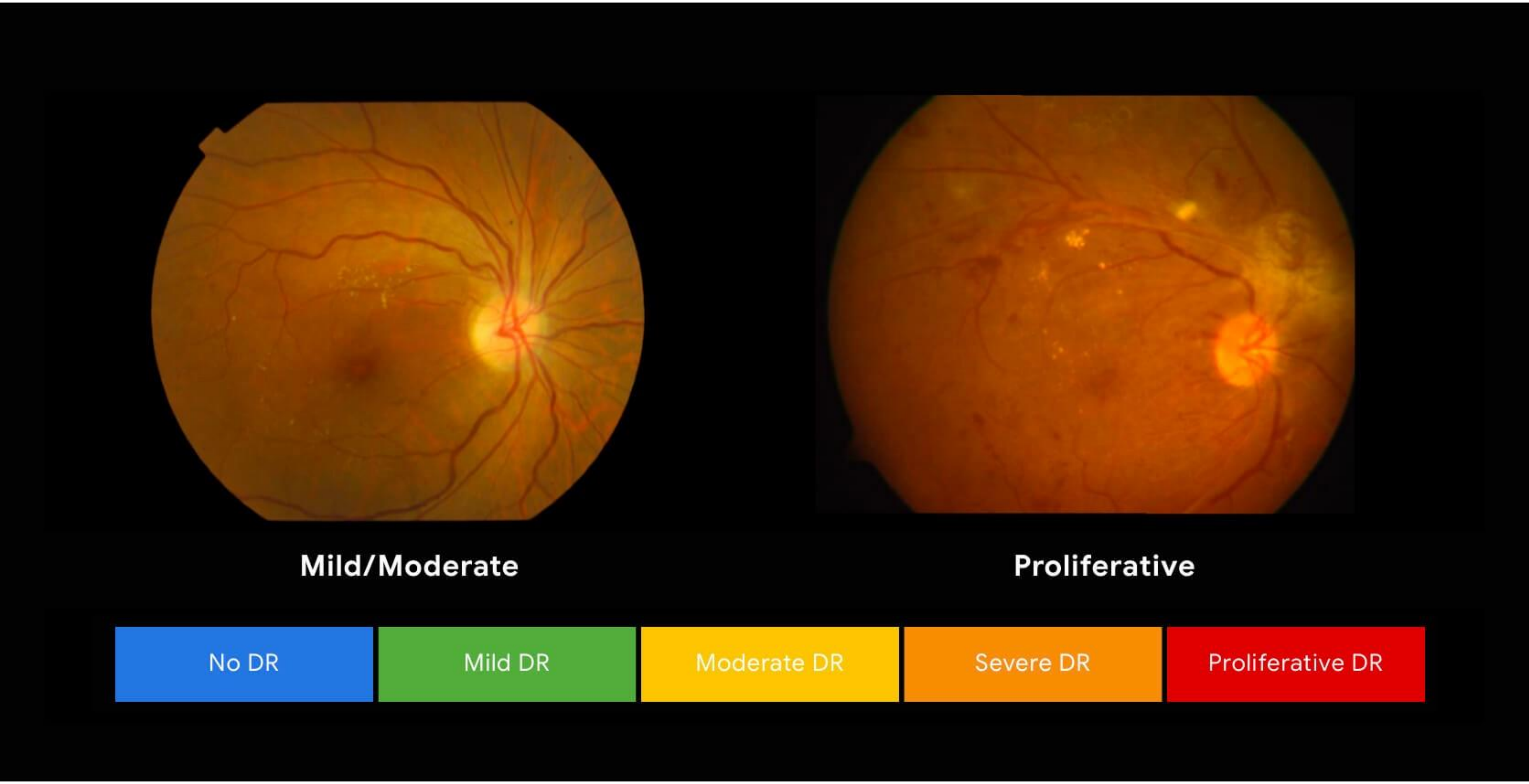


a, Each simulation traverses the tree by selecting the edge with maximum action value Q, plus a bonus $u(P)$ that depends on a stored prior probability P for that edge. **b**, The leaf node may be expanded; the new node is processed once by the policy network p_σ and the output probabilities are stored as prior probabilities P for each action. **c**, At the end of a simulation, the leaf node is evaluated in two ways: using the value network v_θ; and by running a rollout to the end of the game with the fast rollout policy p_π, then computing the winner with function r. **d**, Action values Q are updated to track the mean value of all evaluations $r(\cdot)$ and $v_\theta(\cdot)$ in the subtree below that action.

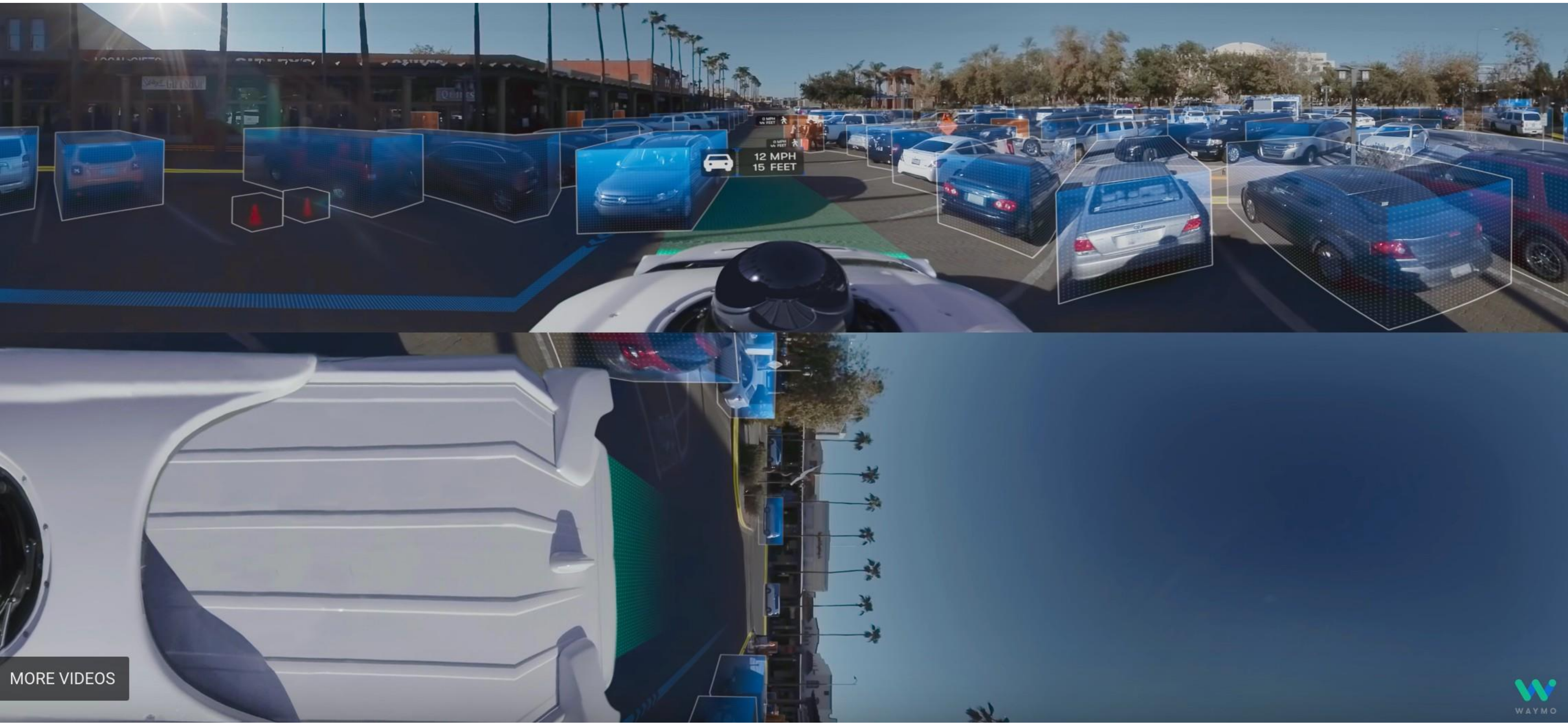
(Silver et al, 2016)



(Google Brain)



(Google Brain)



(WayMo)