Making NNets faster

Problem

- You trained your neural net
- You are happy with test metrics
- But the network is awfully slow

Solution

- Buy bigger GPU
- Buy bigger CPU
- Buy more servers, CPUs, GPUs

End of presentation

Dark knowledge (https://arxiv.org/abs/1503.02531)

- Train big network (or ensemble)

- Use its predictions (probabilities) as additional labeling on top of original labels to train smaller network

Pruning and sparsification

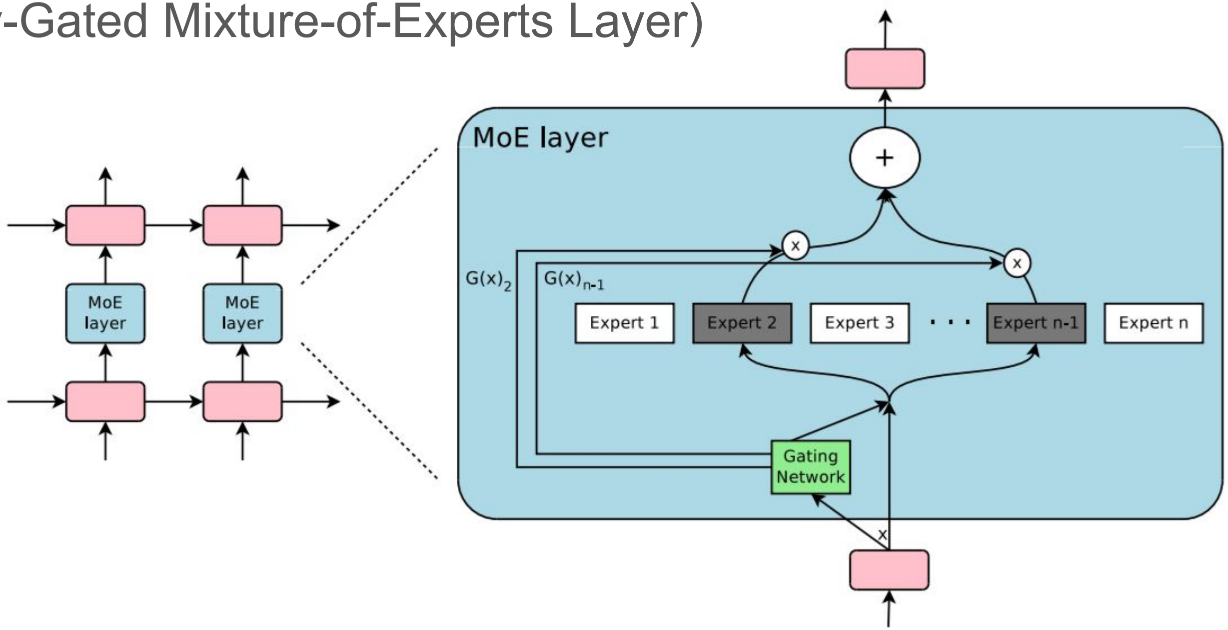
- https://arxiv.org/abs/1711.02782 (Block-Sparse Recurrent Neural Networks)
- https://arxiv.org/abs/1510.00149 (Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding)

- Make weights sparse during training

- But speedup will never be as good as sparsity factor

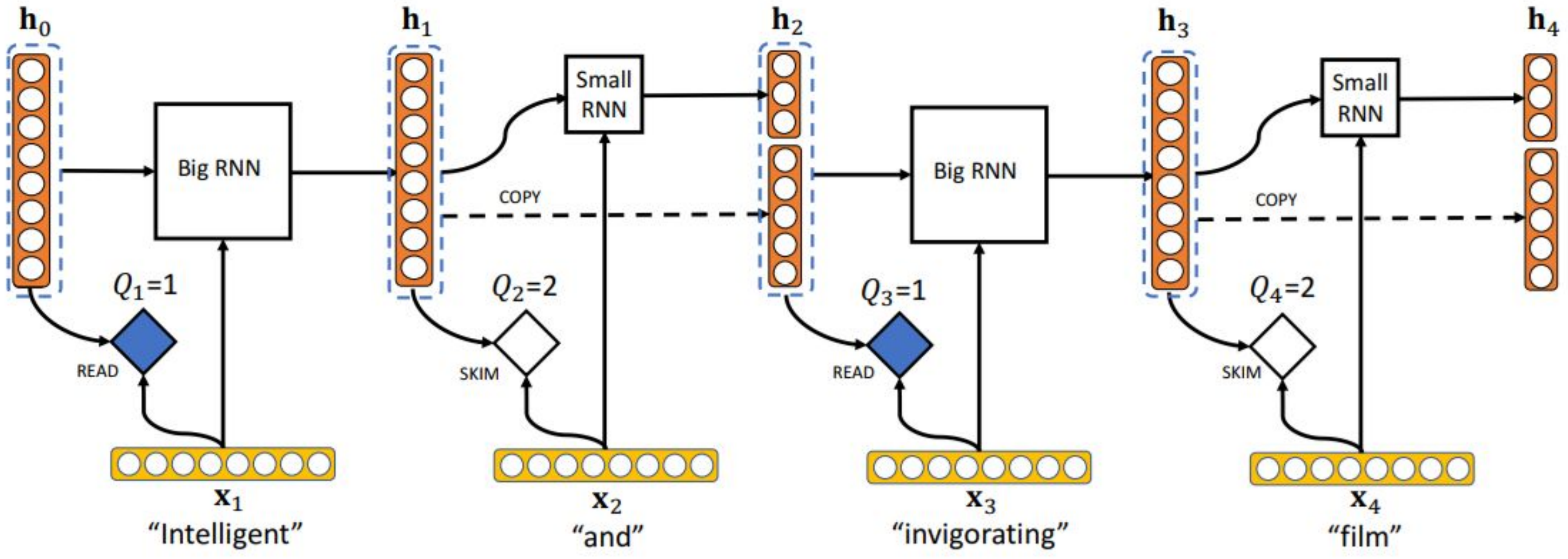
Conditional computation - mixture of experts

https://arxiv.org/abs/1701.06538 (Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer)



Conditional computation - skimRNN

https://arxiv.org/abs/1711.02085 (Neural Speed Reading via Skim-RNN)



How to train?

- Issue is propagation of gradient through gating layer
- Functions like argmax, top-k or y ~ Bernoulli(p) are nondifferentiable
- Big hammer: REINFORCE and friends
- Or make selection differentiable

Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation

(https://arxiv.org/pdf/1308.3432.pdf)

Stochastic nonlinearities

- Typically NN layer is:
  $h_i = f(W_i h_{i-1} + b)$, where f is some nonlinearity like sigmoid, tanh or max(0, x)

- We will consider nonlinearity of form:
  $h = f(a, z)$, where a is an input and z is noise

- Stochastic binary neuron:
  $h = f(a, z) = sigmoid(a) > z$ aka $f(a) \sim Bernoulli(sigmoid(a))$

- Zero derivatives almost everywhere

Noisy rectifier

$h = f(a, z) = max(0, a + z)$ z is either Gaussian or logistic noise ($p(z) = sigm(z)(1 - sigm(z))$)

For fixed *a* and logistic noise we have:

$E[h] = log(1 + exp(a))$

$P(h > 0) = sigm(a)$

Stochastic times smooth

p = sigm(a)

b ~ Bernoulli(sqrt(p))

h = b * sqrt(p)

E[h] = p

P(h > 0) = sqrt(sigm(a))

Straight-through estimator

$h = f(a, z) = sigmoid(a) > z$

During backprop pretend the gradient to be:

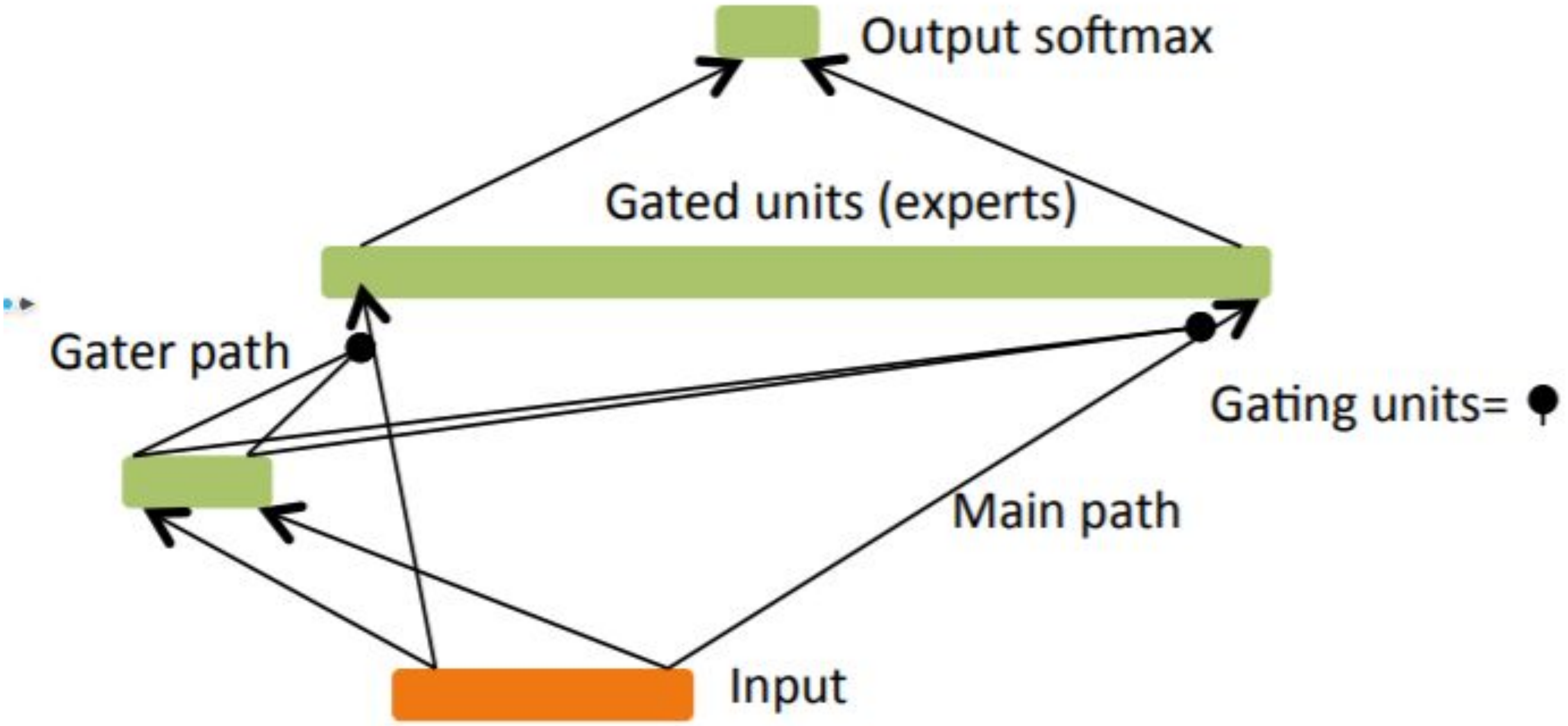
- 0 - if h = 0
- 1 - if h = 1

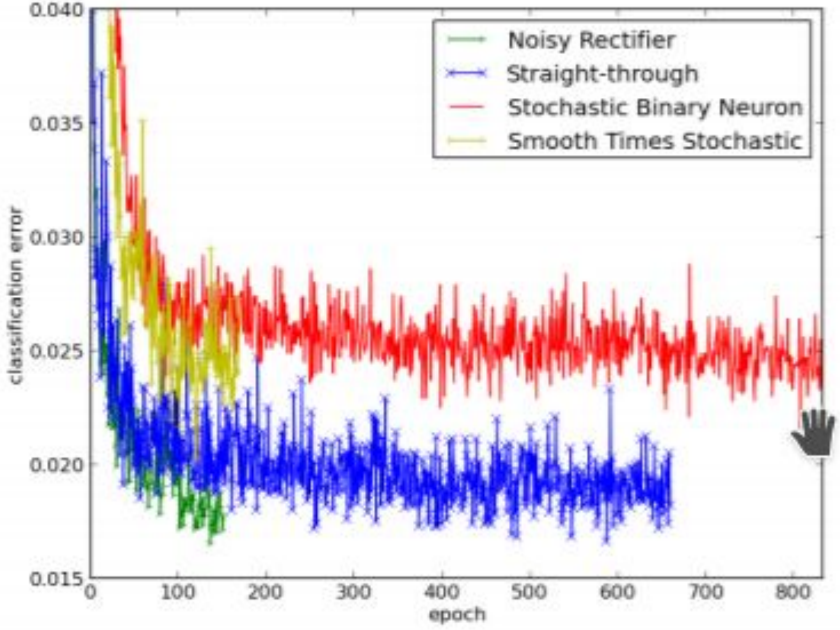
Gumbel softmax (https://arxiv.org/abs/1611.01144)

Let h be categorical variable where $P(h = i) \sim p_i$

Then h can be sampled as $h = argmax(g_i + log\ p_i)$, where $g_i \sim Gumbel(0, 1)$
g_i *can be drawn as* $-log(-log(Uniform(0, 1))$

Instead of argmax we can use softmax with temperature.


Output softmax
Gated units (experts)
Gater path
Gating units=
Main path
Input



	train	valid	test
Noisy Rectifier	6.7e-4	1.52	1.87
Straight-through	3.3e-3	1.42	1.39
Smooth Times Stoch.	4.4e-3	1.86	1.96
Stoch. Binary Neuron	9.9e-3	1.78	1.89
Baseline Rectifier	6.3e-5	1.66	1.60
Baseline Sigmoid+Noise	1.8e-3	1.88	1.87
Baseline Sigmoid	3.2e-3	1.97	1.92

The real end. Go to lunch!