Learning When to Halt With Adaptive Computation Time

Alex Graves, Oriol Vinyals, Michael Figurnov, Rafal Jozefowicz

Adaptive Computation Time With Recurrent Neural Networks
Graves, 2016

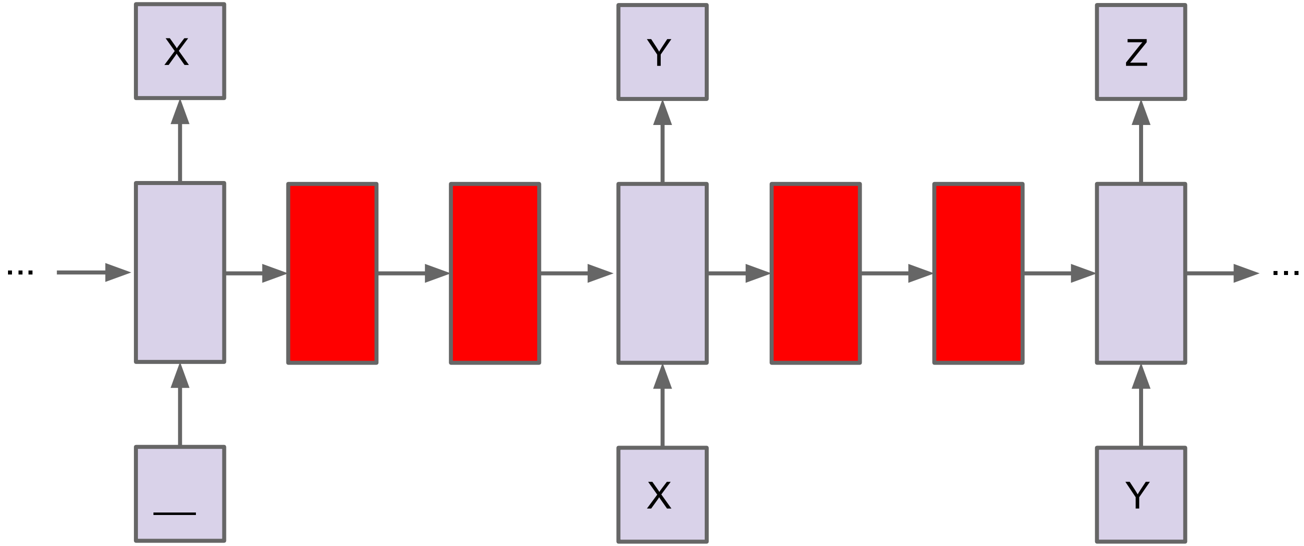
Spatially Adaptive Computation Time for Residual Networks
Figurnov et. al, 2016

Publish Me Soon!
Vinyals, Graves, Raffel, Jozefowicz, 20??

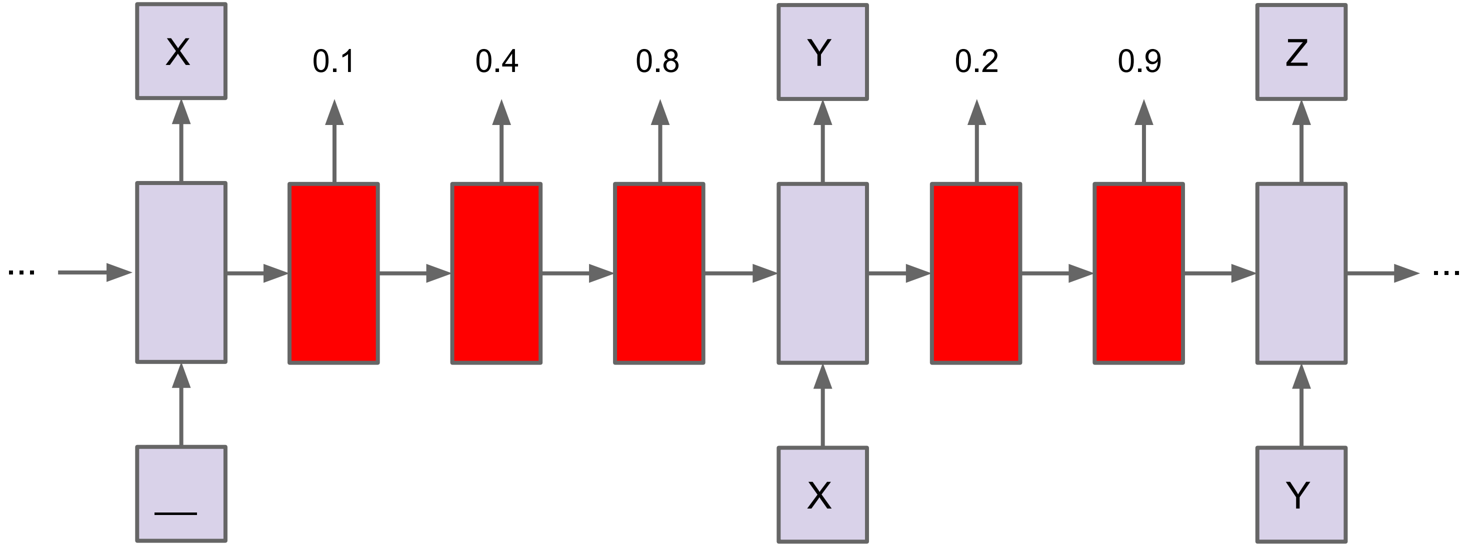
Motivation

- At the moment the number of steps of computation an RNN gets for a given problem is determined by the data (sequence length) and the experimenter (network depth, padding in sequence...)
- Would prefer the net to decide how to long to ‘ponder’ each input before it outputs an answer
- Clearly useful for algorithmic / planning type problems with a high variance in complexity (e.g. program induction, pathfinding…)
- Can also be more efficient for conventional tasks such as machine translation, language modelling and image processing
- More important for supervised learning than e.g. RL

Fixed Computation Time



Adaptive Computation Time (ACT)



Adaptive Computation Time (ACT)

Add a *halting unit h* to the output

$$h_t^n = \sigma\left(W_h s_t^n + b_h\right)$$

Use this to define the *halt probability* p_t^n at ponder step *n*

$$p_t^n = \begin{cases} R(t) \text{ if } n = N(t) \\ h_t^n \text{ otherwise} \end{cases}$$

Where *N(t)* is # updates at *t*

$$N(t) = \min\{n' : \sum_{n=1}^{n'} h_t^n >= 1 - \epsilon\}$$

And *R(t)* is the *remainder* at *t*

$$R(t) = 1 - \sum_{n=1}^{N(t)-1} h_t^n$$

The final states and outputs at *t* are weighted sums (!)

$$s_t = \sum_{n=1}^{N(t)} p_t^n s_t^n \qquad y_t = \sum_{n=1}^{N(t)} p_t^n y_t^n$$

Limiting Computation Time

We always want answers as quick as possible, but can't tell in advance how long that will be (halting problem). ACT adds a ponder cost *P(**x**)* to the loss function and uses a time penalty *τ* to trade off accuracy against speed

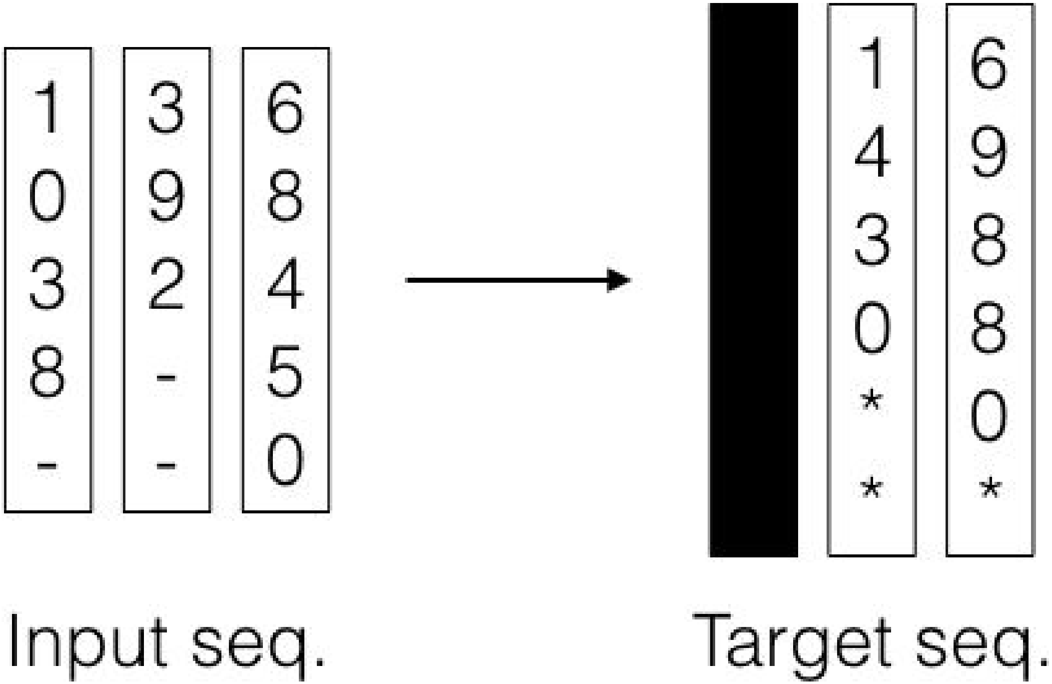
$$\hat{\mathcal{L}}(\mathbf{x}, \mathbf{y}) = \mathcal{L}(\mathbf{x}, \mathbf{y}) + \tau \mathcal{P}(\mathbf{x}) \qquad \mathcal{P}(\mathbf{x}) = \sum_{t-1}^{T} N(t) + R(t)$$

*P(**x**)* is an upper bound on the *total computation* $\sum_t N(t)$. It is discontinuous when N(t) changes, but we just ignore that and minimise *R(t)*, which maximises the amount of halt probability mass assigned to steps < *N(t)*.

Minimising expected emission time doesn't do this

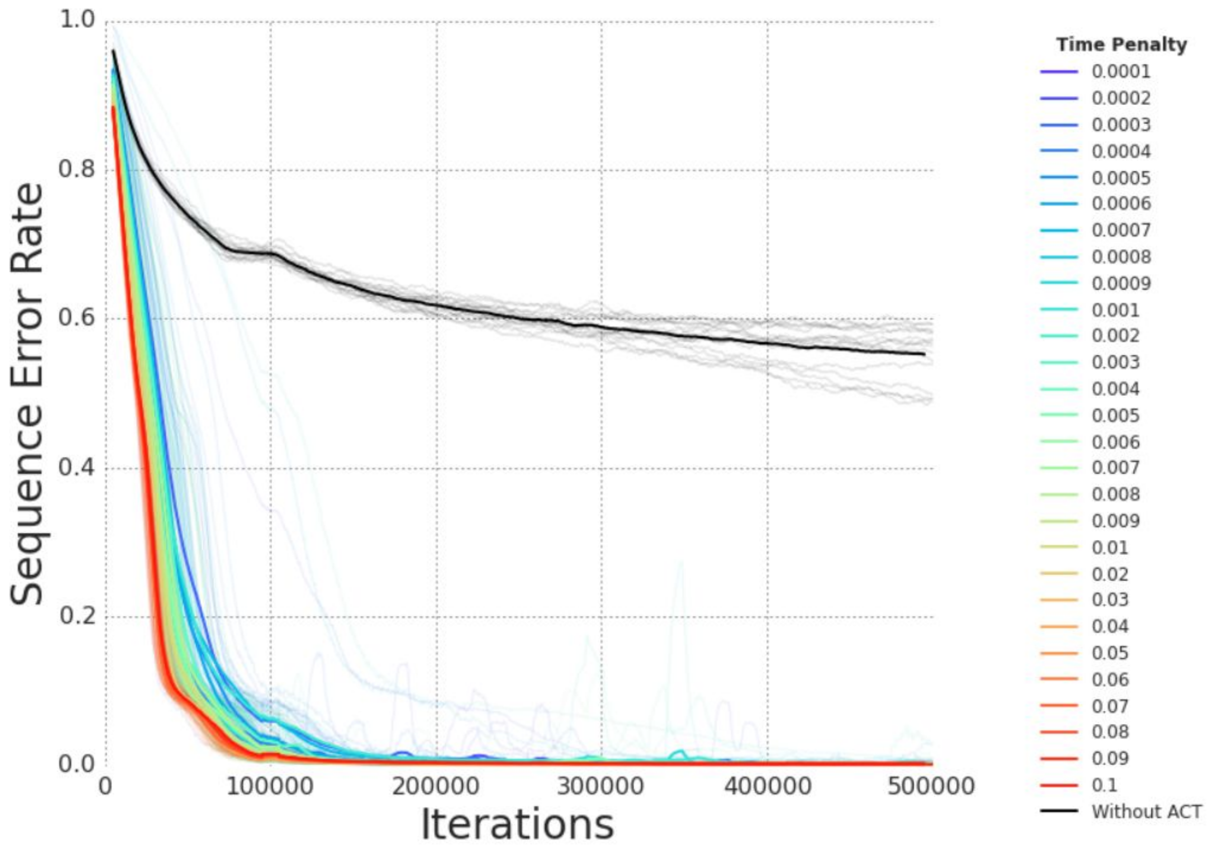
Toy Experiments: Addition

Slide Credit:
Alex Graves



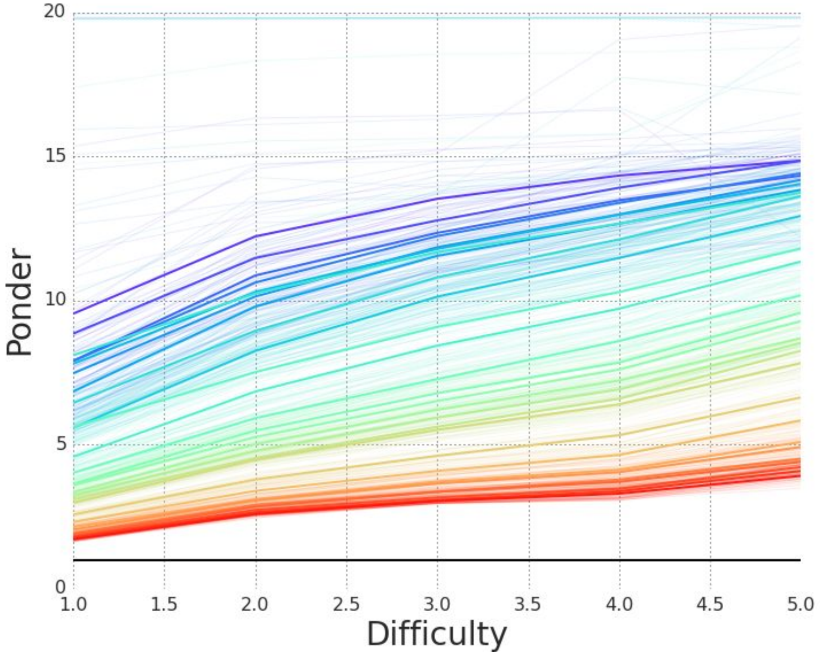
Toy Experiments: Addition

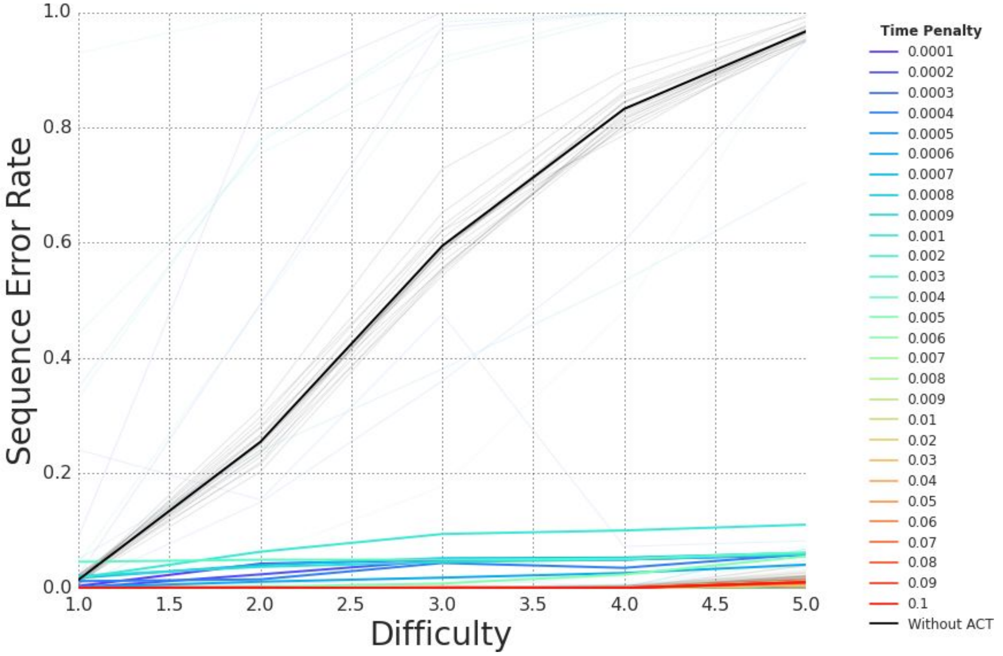
Slide Credit:
Alex Graves



Toy Experiments: Addition

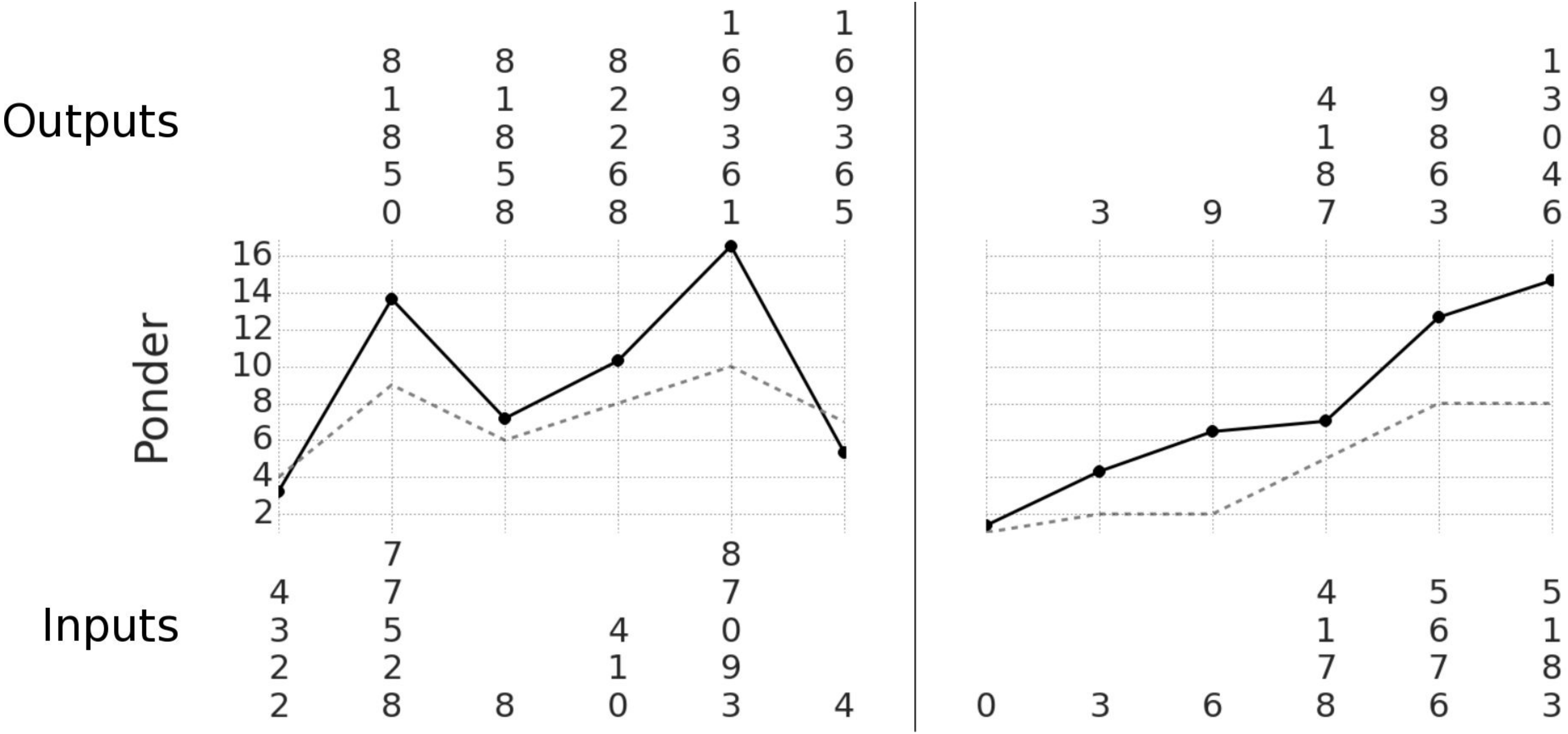
Slide Credit:
Alex Graves



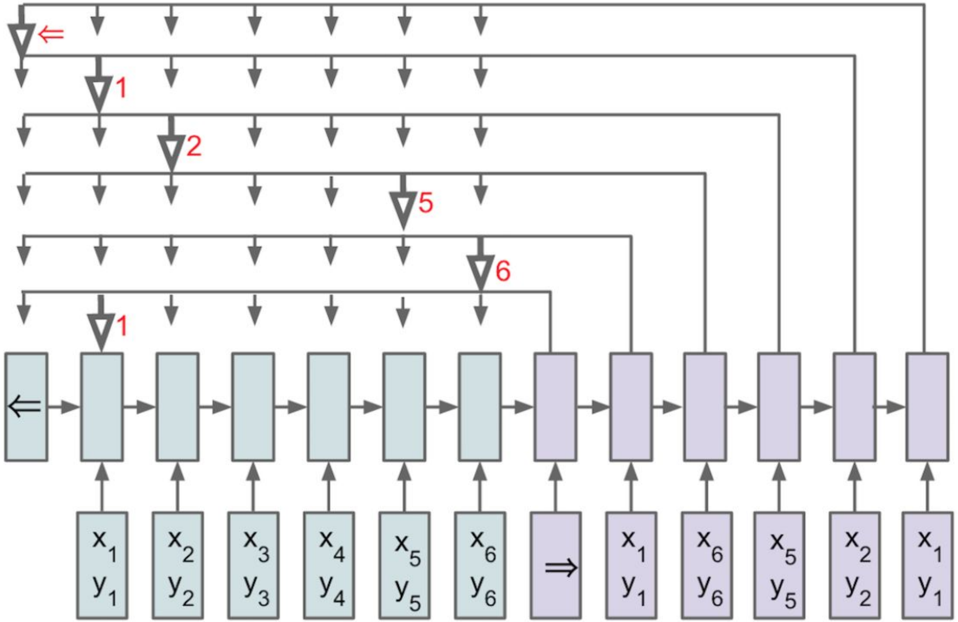


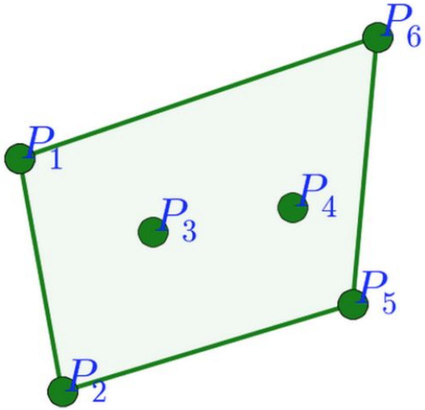
Toy Experiments: Addition

Slide Credit:
Alex Graves



Toy Experiments: PtrNets, TSP / ConvexHull





Pointer Nets + ACT

Convex Hull (50): 73% accuracy -> **85%** accuracy

TSP (50):

Optimal: 5.7

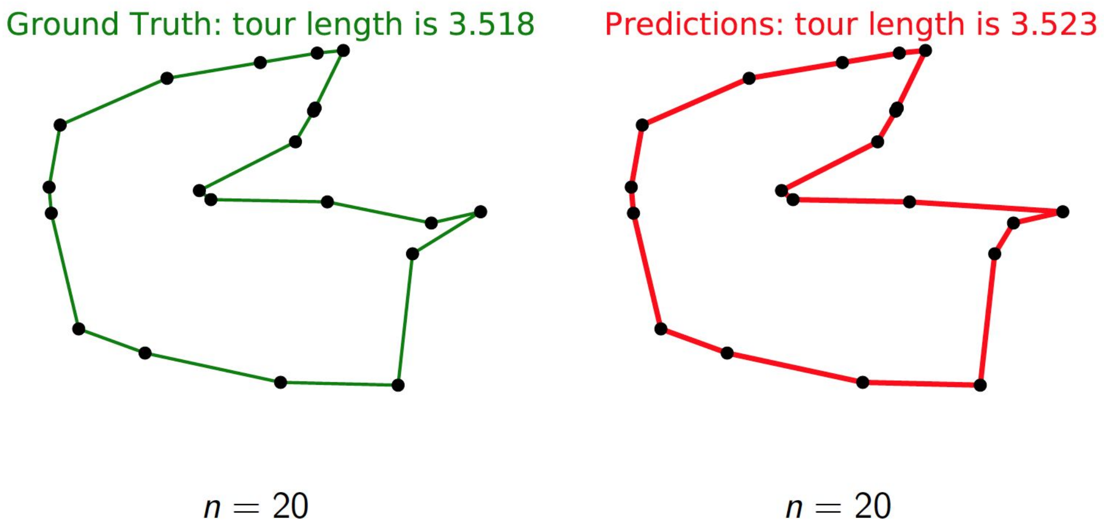
Heuristic algorithm: 5.8

PTR-NETs (NIPS version): 6.1

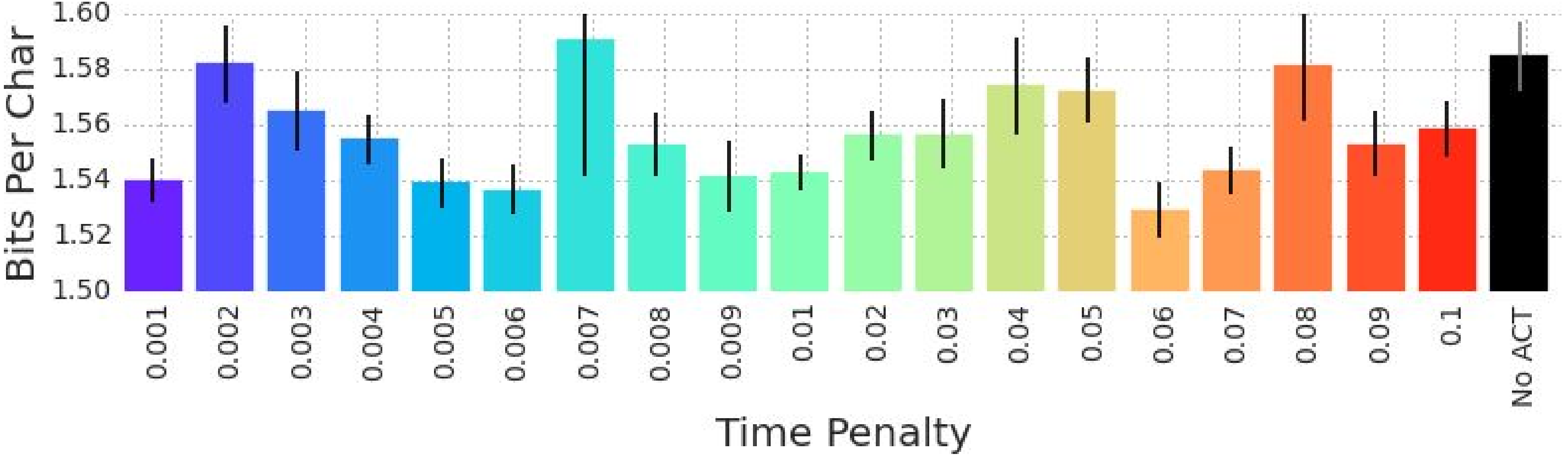
PTR-NETs + ACT: **5.9**

(new ICLR17 submission)

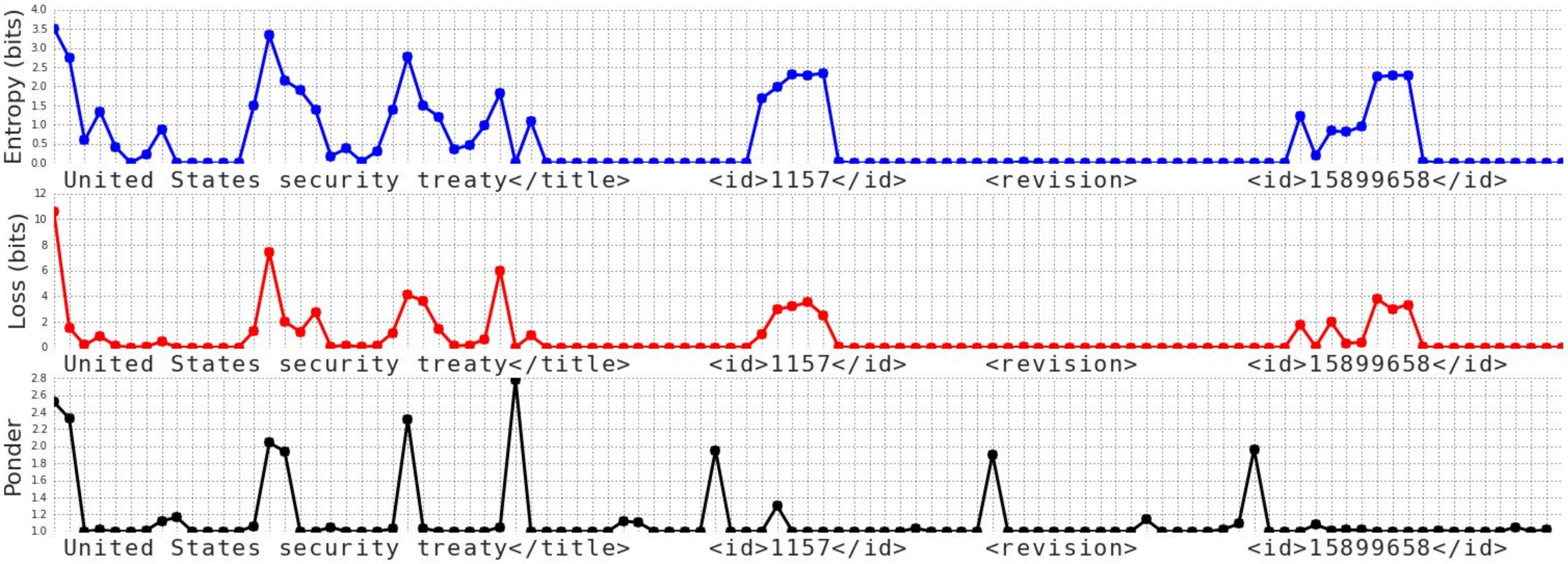
RL-PTR-NETs: 5.7



Wikipedia Character Prediction Results



Pondering Wikipedia



Word level LM

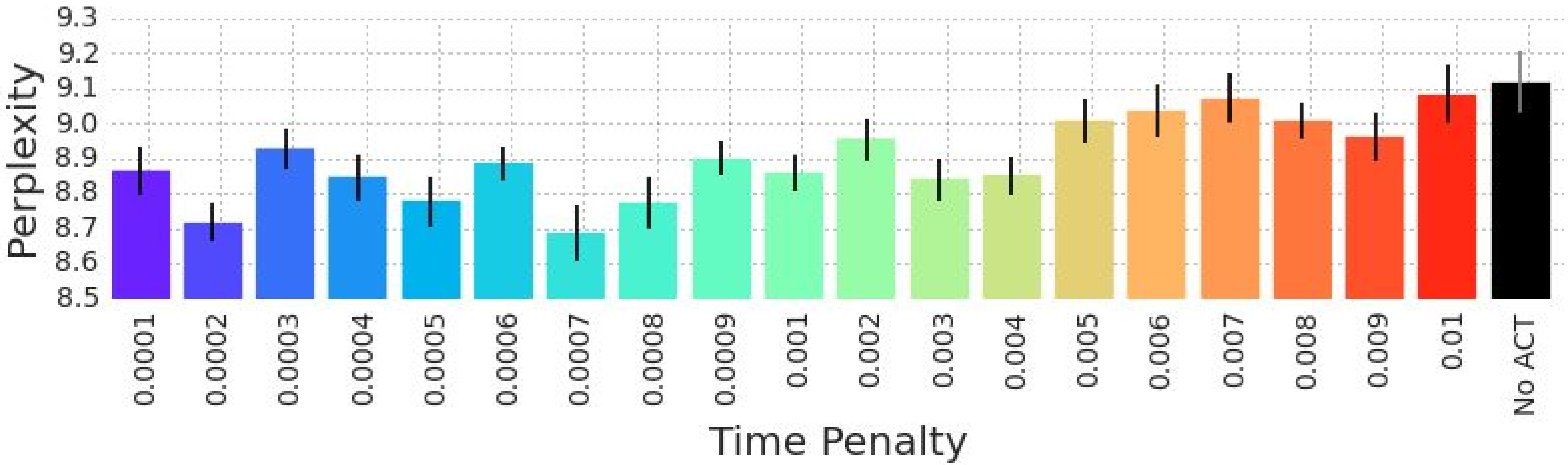
Baseline: 44 PPL

With fix pondering of 5: **39**

With pondering up to 10 (average **4.3**): **39**

<S>	2
Elsewhere	3
,	4
Nymex	4
WTI	4
crude	5
<UNK>	6
under	5
the	4
$	3
70	3
a	4
barrel	5
mark	6
,	5
losing	6
0.3	3
per	2
cent	4
to	4
$	3
69.57	4
.	2

Character Level Machine Translation (BTEC)



Machine Translation + ACT

Dataset: WMT14 test set, English to French

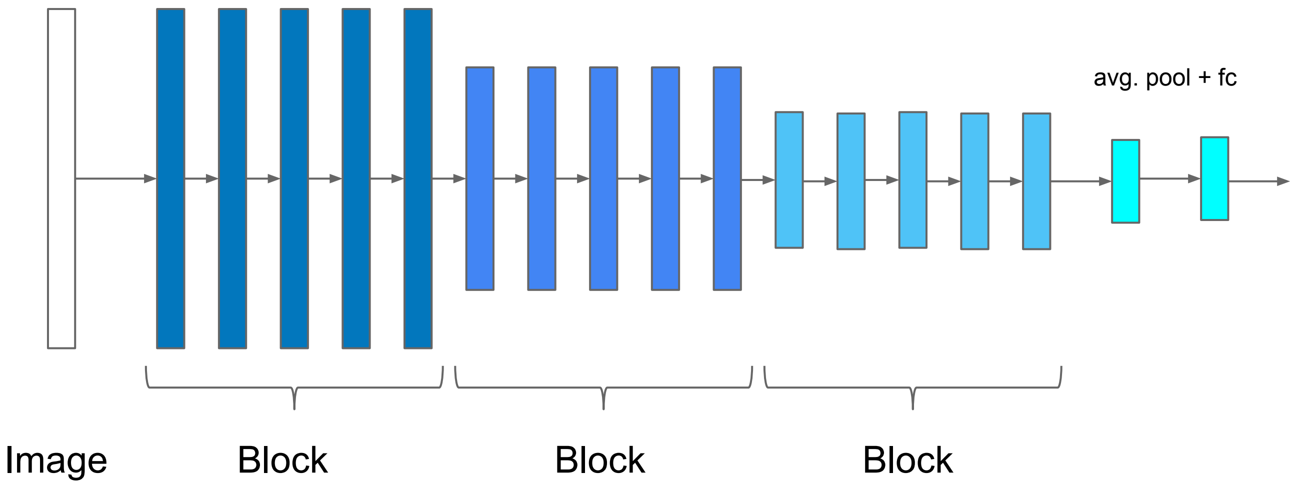
(SMT): 37.0 BLEU

Baseline AttLSTM: 3.4 PPL, 37.5 BLEU

AttLSTM + ACT (between input and output): 3.3 PPL, 37.6 BLEU

AttLSTM + ACT: **3.1** PPL, **38.3** BLEU

Residual Network (ResNet)

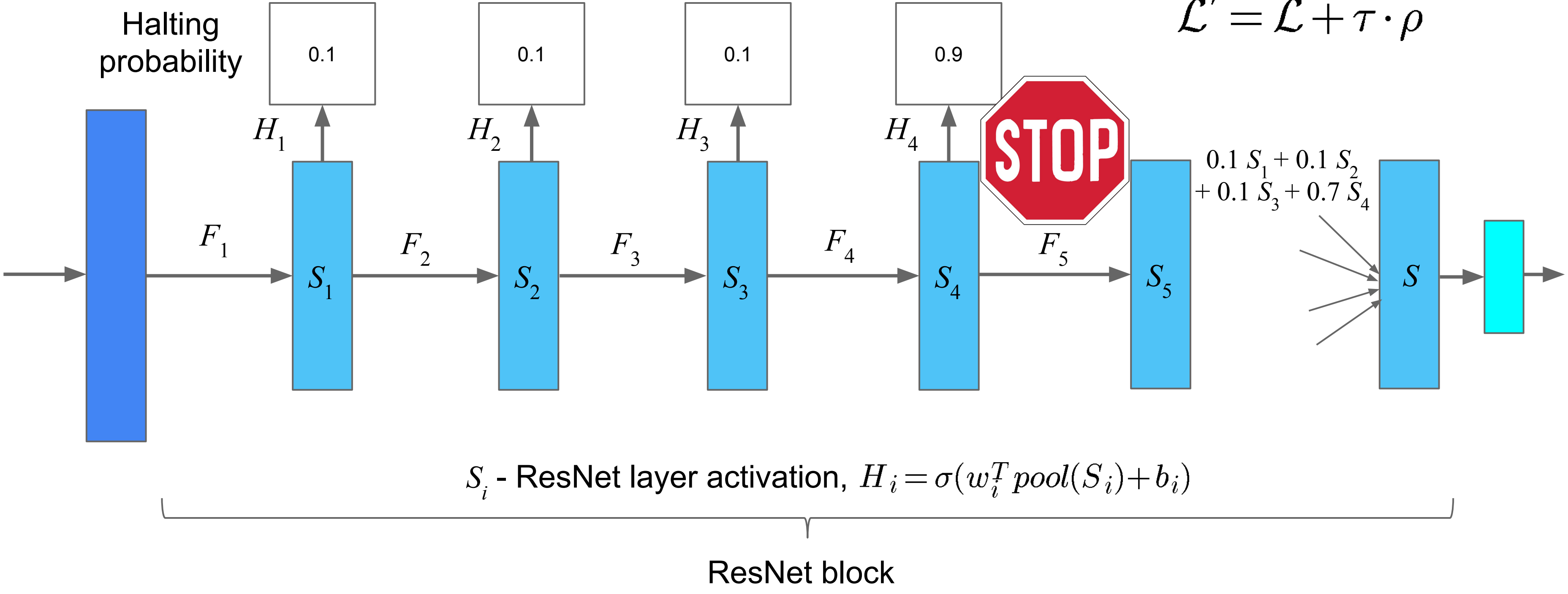


Residual layer: $y = \mathcal{F}(x) + x$

ResNet with Adaptive Computation Time

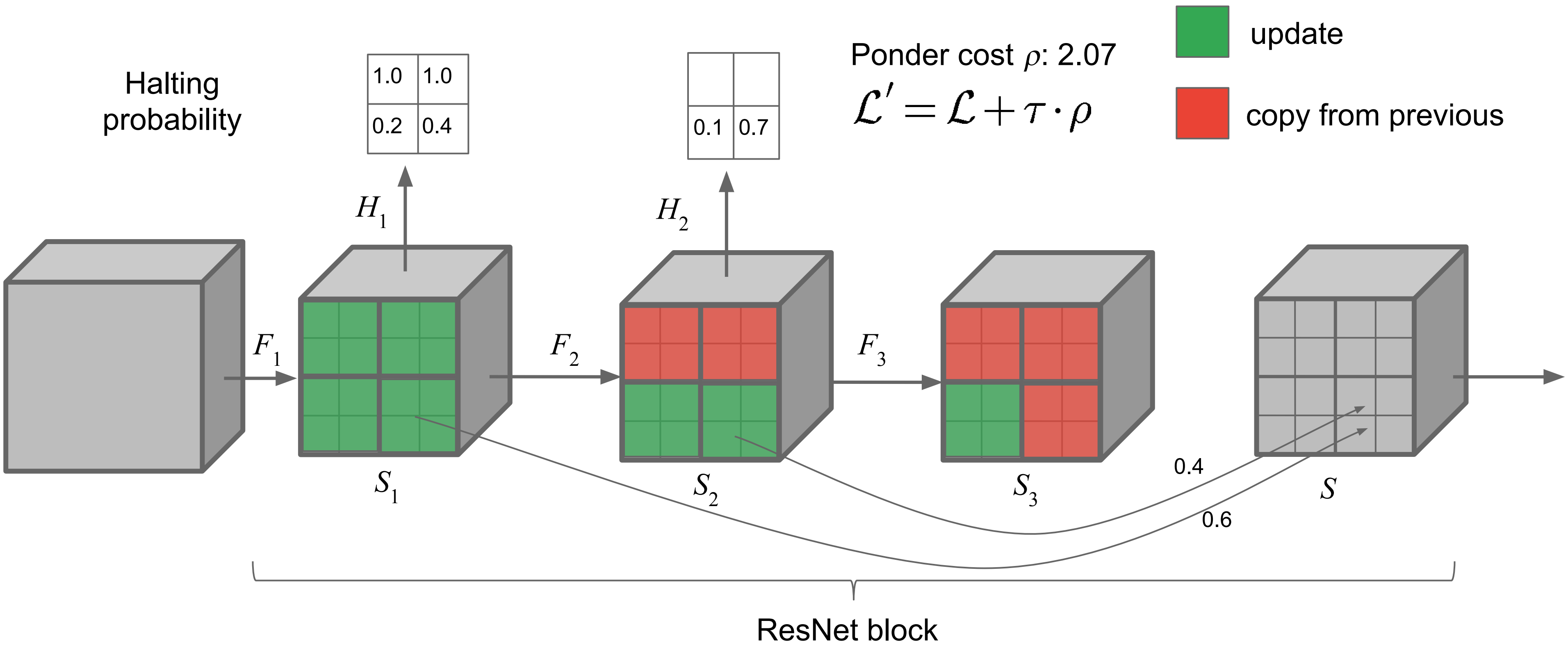
Ponder cost ρ: 4.7

$$\mathcal{L}' = \mathcal{L} + \tau \cdot \rho$$



Spatially Adaptive Computation Time for Residual Networks, Figurnov et. al, 2016

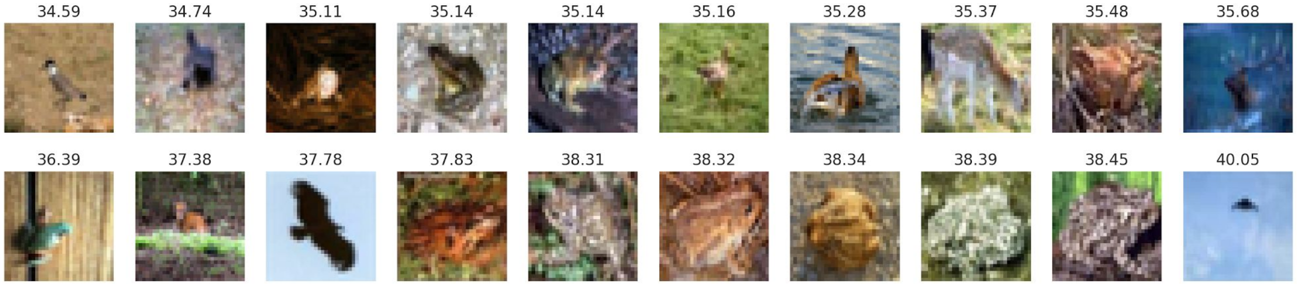
ResNet with Spatially Adaptive Computation Time



CIFAR-10 ACT qualitative results

Low ponder cost

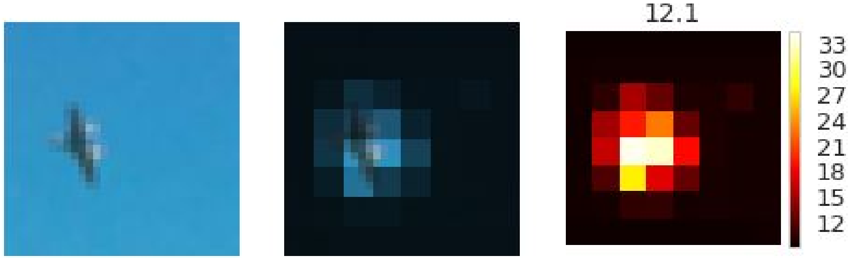
High ponder cost

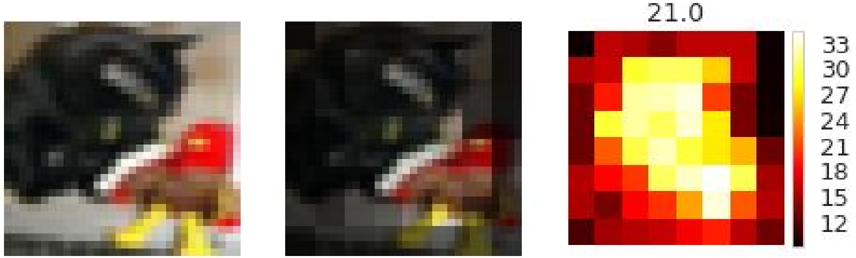


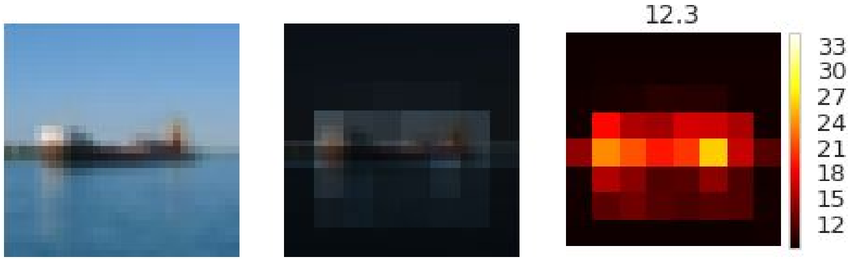
ResNet-110, τ = 0.01

Slides: go/resnet-act-midterm

CIFAR-10 SACT qualitative results

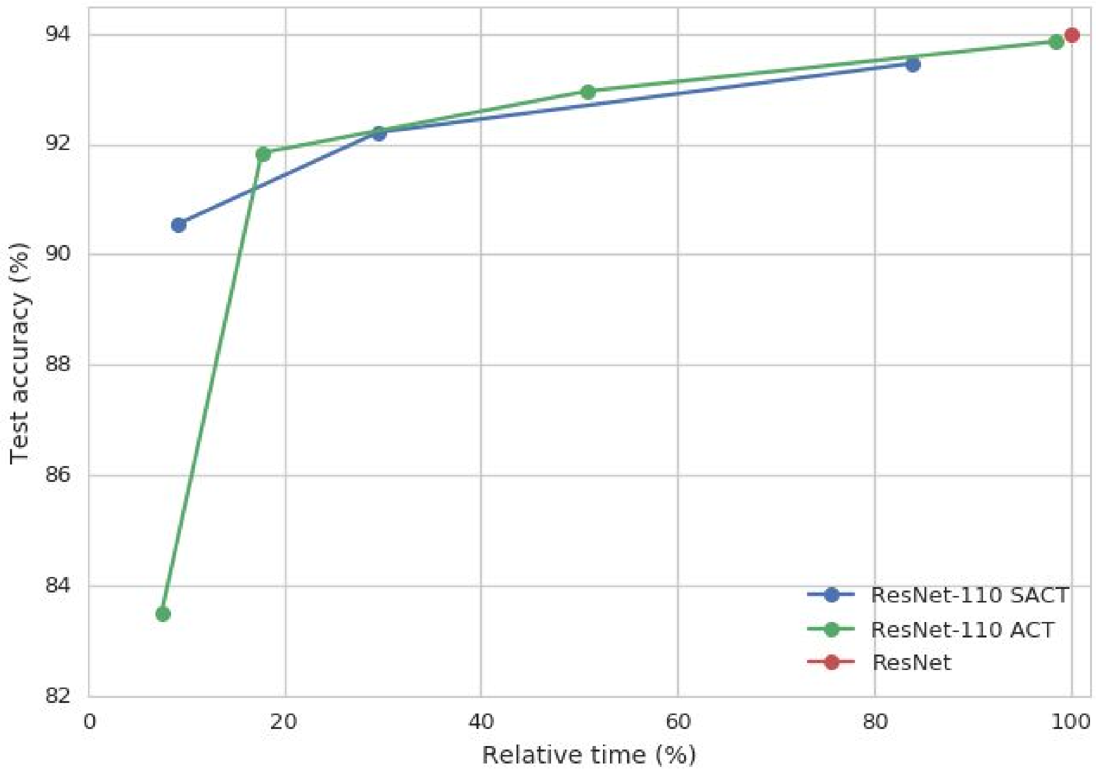




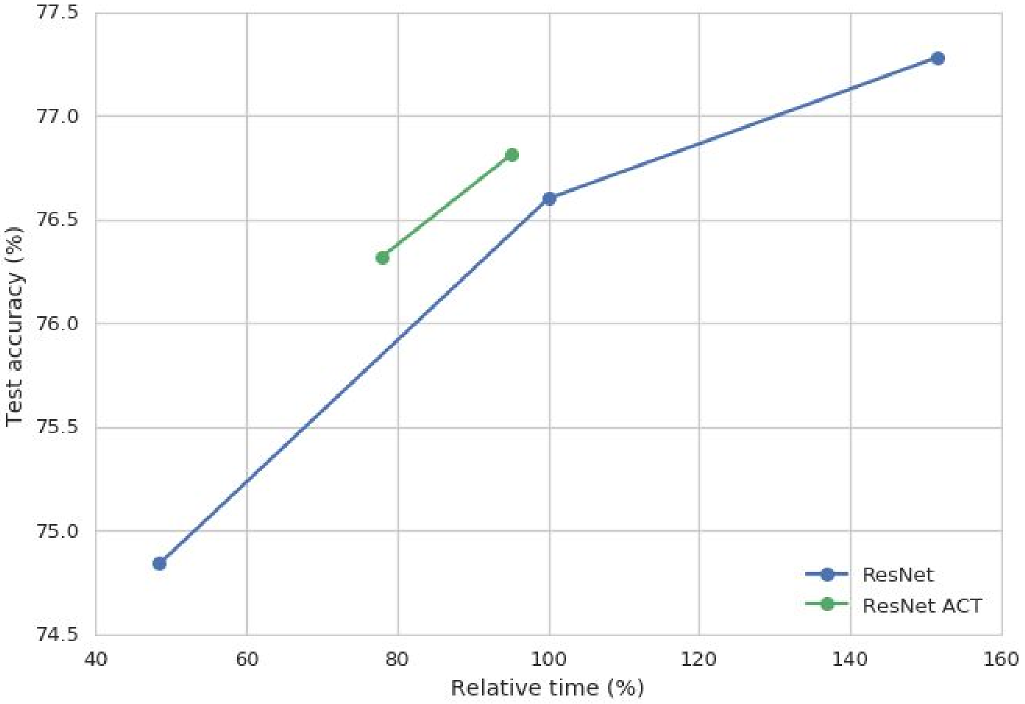


ResNet-110, τ = 0.01

CIFAR-10 ACT vs. SACT



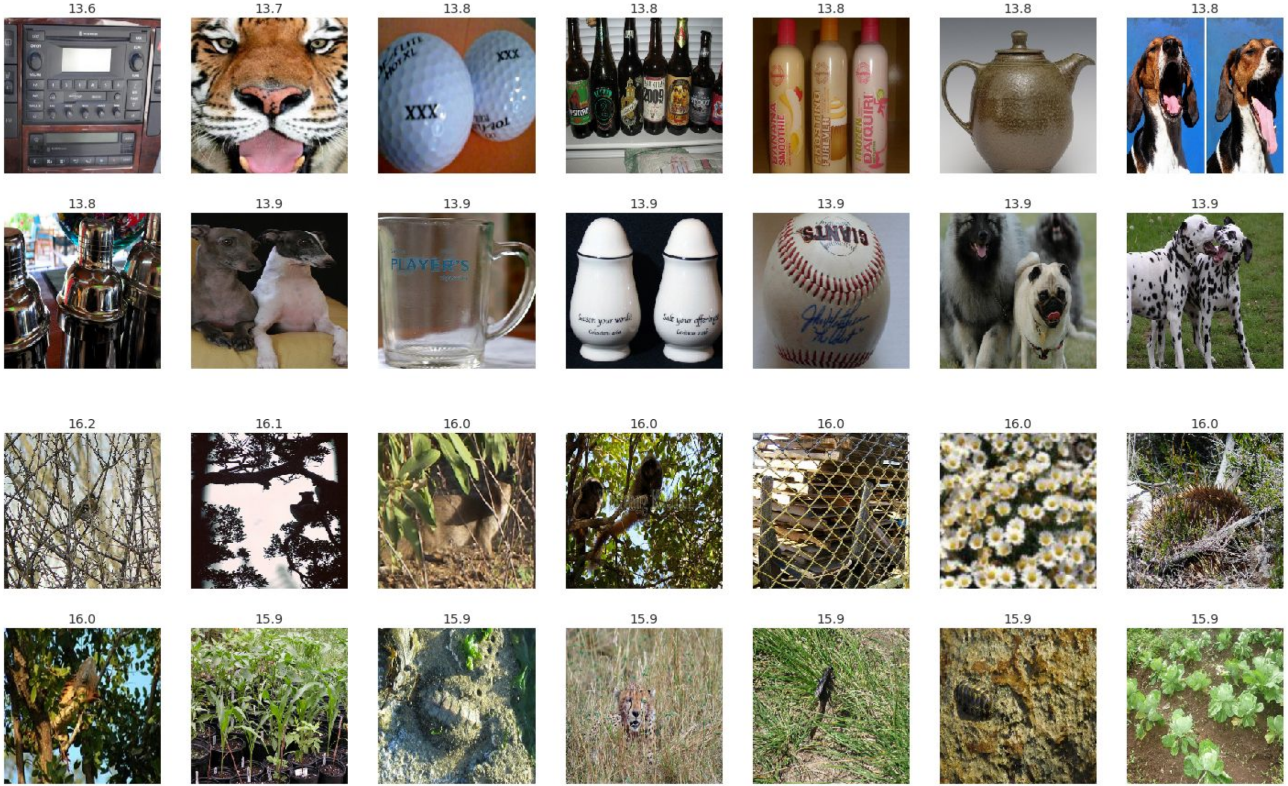
ImageNet ACT accuracy vs. time



ImageNet ACT qualitative results

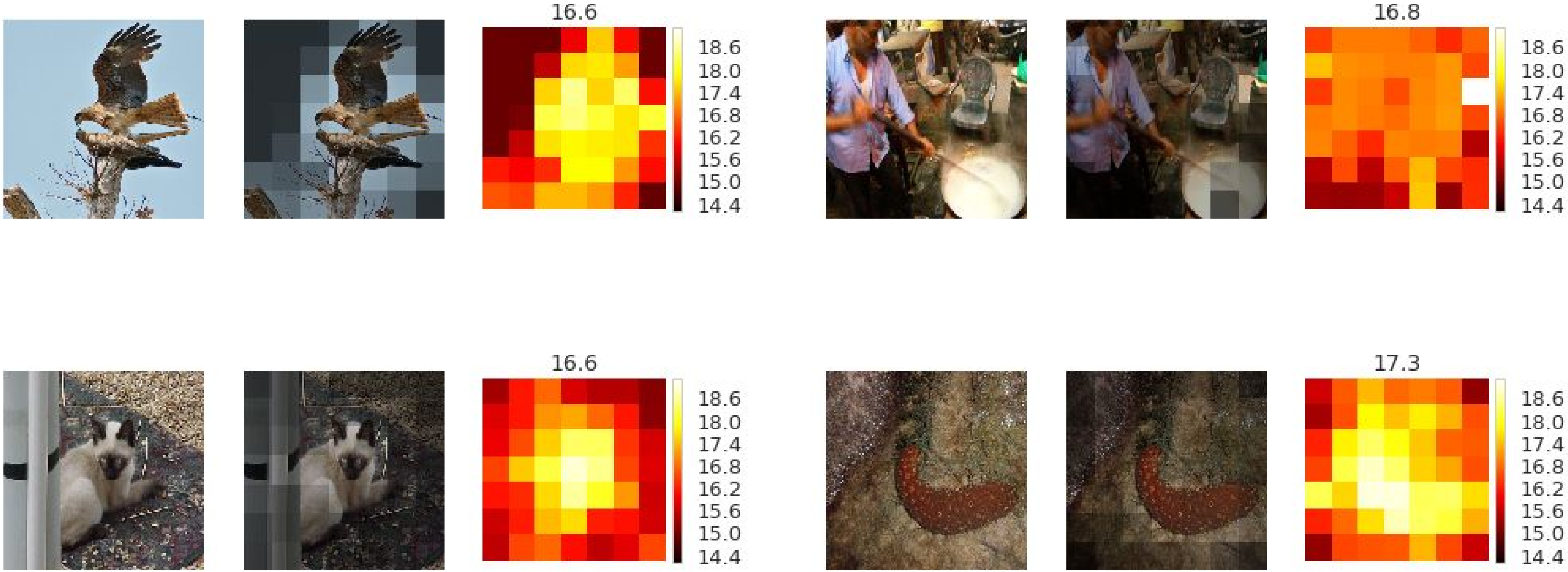
Low ponder cost

High ponder cost



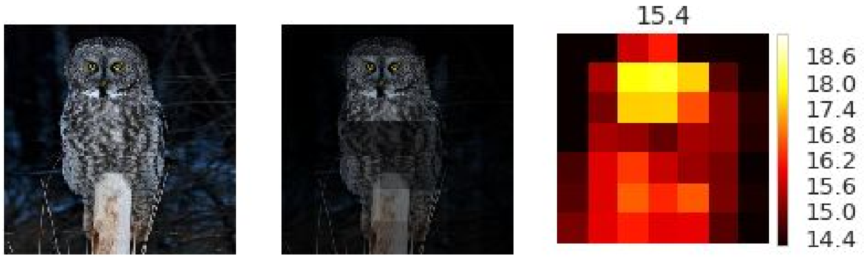
ResNet-110, τ = 0.01, 3M steps (unconverged)

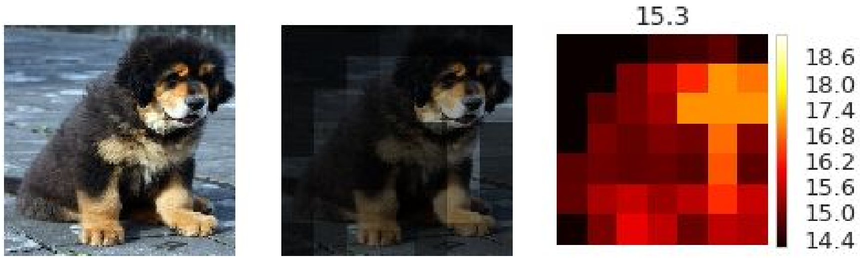
ImageNet SACT random examples



ResNet-110, τ = 0.01, 3.5M steps (unconverged)

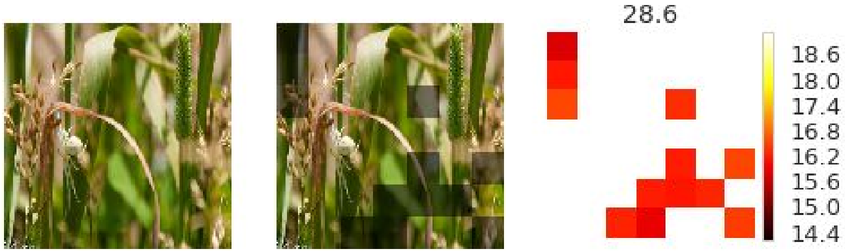
ImageNet SACT low ponder cost examples

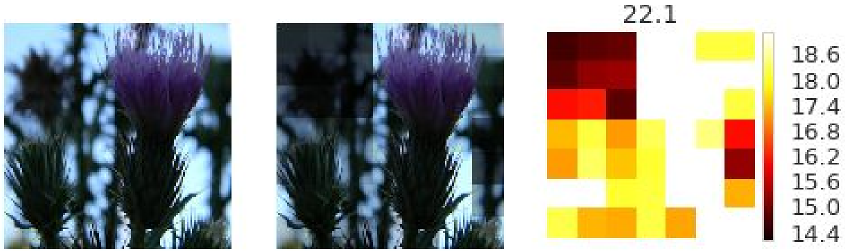


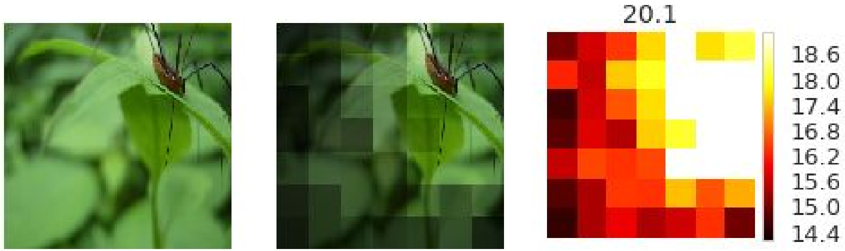


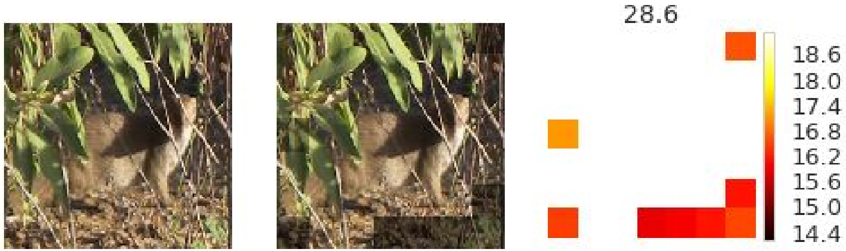
ResNet-110, τ = 0.01, 3.5M steps (unconverged)

ImageNet SACT high ponder cost examples









ResNet-110, τ = 0.01, 3.5M steps (unconverged)