Controlling the Computational Cost of Machine Learning

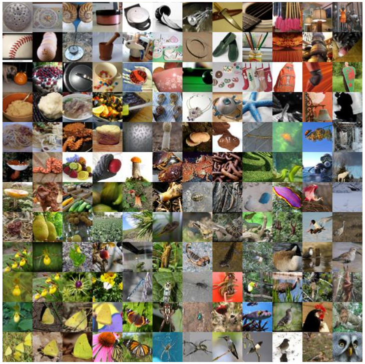
François Fleuret

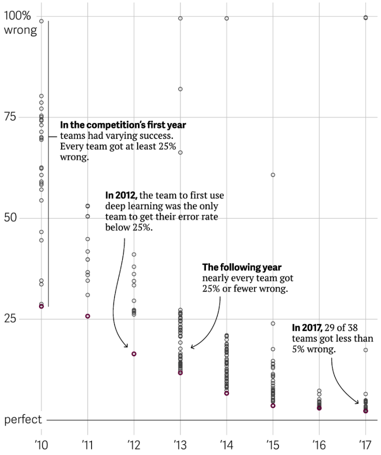
Joint work with

Angelos Katharopoulos, Olivier Canévet, and Cijo Jose

The unreasonable cost of ML

The last decade has seen artificial neural networks improving on many fundamental tasks from barely usable to close to or beyond human performance.

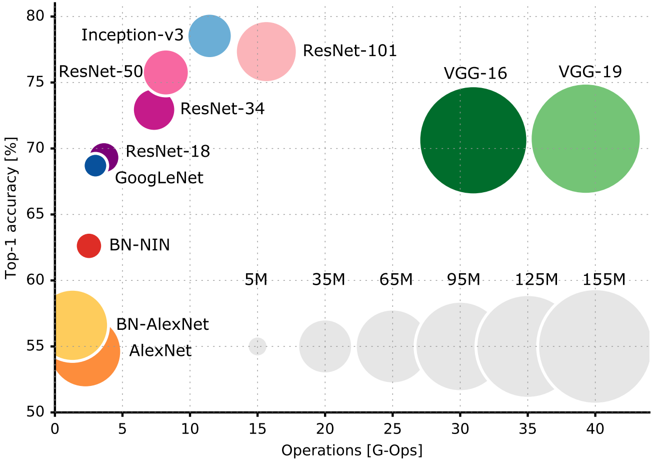

ImageNet



(Gershgorn, 2017)

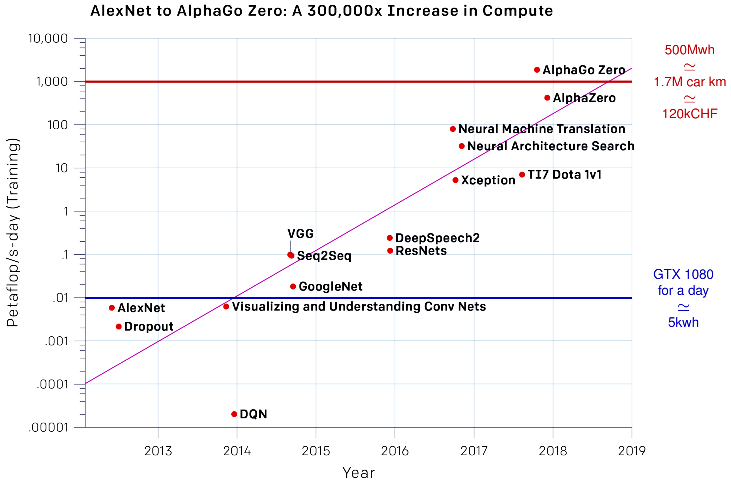
The unreasonable cost of ML

This performance is directly related to the computational cost of predictive models.



(Canziani et al., 2016)

The unreasonable cost of ML

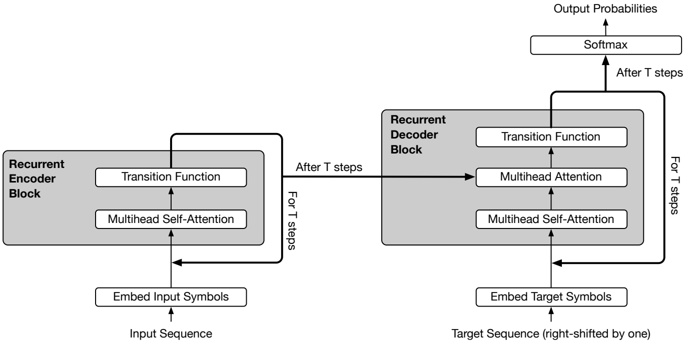


(OpenAI blog, 2018)

1 petaflop/s-day $\simeq$ 100 GPUs for a day, $\simeq$ 500kwh, $\simeq$ 100CHF of electricity

The unreasonable cost of ML

This trend has no end in sight, with techniques such as architecture optimization (Zoph and Le, 2016), and dynamical models (Dehghani et al., 2018; Dupont et al., 2019).



(Dehghani et al., 2018)

The unreasonable cost of ML

Given the predicted ubiquity of AI, controlling computation is both a practical and a fundamental issue:

- reduce the economic cost,
- control the environmental impact,
- deploy on low-power environments,
- ensure privacy by allowing on-site inference,

The unreasonable cost of ML

Given the predicted ubiquity of AI, controlling computation is both a practical and a fundamental issue:

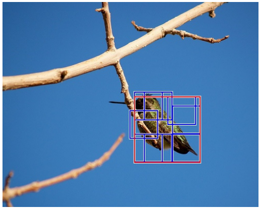
- reduce the economic cost,
- control the environmental impact,
- deploy on low-power environments,
- ensure privacy by allowing on-site inference,
- move to next-generation signal size (microscopy, high-energy physics),
- keep the growth of model size toward biological scales.

It may also shine a light on fundamental connections between computational reduction and generalization.

The unreasonable cost of ML

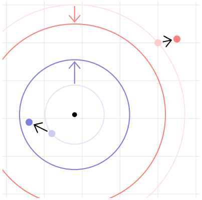
This issue pre-dates deep learning, and is amenable to exact methods in some cases:

FFT for convolutions



(Dubout and Fleuret, 2012, 2013)

Geometrical bounds for clustering



(Newling and Fleuret, 2016a,b, 2017)

Kronecker factorization of weight matrices

$$M = M_1 \otimes \cdots \otimes M_N \in \mathbb{R}^{2^N \times 2^N}$$

(Jose et al., 2018)

The unreasonable cost of ML

The complexity of deep models makes the derivation of exact accelerations impractical, beside speeding up low-level linear operations.

Approximate computational reduction has been achieved with

- smaller signal size and pre-trained models,
- smaller architectures and compression (Iandola et al., 2016; Tan and Le, 2019; Ba and Caruana, 2014; Hinton et al., 2015),
- better normalizations (Glorot and Bengio, 2010; Ioffe and Szegedy, 2015),
- aggressive optimization (Smith and Topin, 2017),
- time-dependent computation (Shelhamer et al., 2016).

Very few methods have data-driven dynamic computation modulation.

Weighting by sampling

Weighting by sampling

Costly ML processings are often sums of terms computed independently.

During training:

$$\sum_n \nabla_{\mathcal{L}(f)}(x_n)$$

Gradient

$$\sum_n \exp(-y_n\, f(x_n))\, y_n\, h(x_n)$$

Boosting edge

During inference:

$$\sum_n \alpha_n k(x, x_n) y_n$$

SVM value

$$\sum_n \text{softmax}\left(\frac{QK_n^T}{\sqrt{d}}\right) V_n$$

Attention-based value

Weighting by sampling

We can use Monte Carlo with importance sampling to estimate a sum. Given

$$v_n \in \mathbb{R}^D,\ n = 1, \dots N,$$

and

$$\mu \in \mathscr{D}(\{1, \dots, N\}),\ \mu > 0,$$

we have

$$\begin{aligned} S &= \sum_{n=1}^{N} v_n \\ &= \sum_{n=1}^{N} \mu(n) \frac{v_n}{\mu(n)} \\ &= \mathbb{E}_{\mathbf{N} \sim \mu} \left[\frac{v_{\mathbf{N}}}{\mu(\mathbf{N})} \right] \end{aligned}$$

Weighting by sampling

We can use Monte Carlo with importance sampling to estimate a sum. Given

$$v_n \in \mathbb{R}^D, \; n = 1, \dots N,$$

and

$$\mu \in \mathscr{D}(\{1, \dots, N\}), \; \mu > 0,$$

we have

$$\begin{aligned} S &= \sum_{n=1}^{N} v_n \\ &= \sum_{n=1}^{N} \mu(n) \frac{v_n}{\mu(n)} \\ &= \mathbb{E}_{\mathbf{N} \sim \mu} \left[\frac{v_{\mathbf{N}}}{\mu(\mathbf{N})} \right] \\ &\simeq \frac{1}{K} \sum_{k=1}^{K} \frac{v_{\mathbf{N}_k}}{\mu(\mathbf{N}_k)}. \end{aligned}$$

where $\mathbf{N}_1, \dots, \mathbf{N}_K$ are i.i.d. $\sim \mu$, possibly with $K \ll N$.

Weighting by sampling

While

$$\frac{v_{\mathbf{N}}}{\mu(\mathbf{N})}$$

is an unbiased estimator of S for any μ, the sum of its components' variance

$$\mathbb{E}_{\mathbf{N}\sim\mu}\left[\left\|S - \frac{v_{\mathbf{N}}}{\mu(\mathbf{N})}\right\|^2\right]$$

depends on it.

Weighting by sampling

While

$$\frac{v_{\mathbf{N}}}{\mu(\mathbf{N})}$$

is an unbiased estimator of S for any μ, the sum of its components' variance

$$\mathbb{E}_{\mathbf{N} \sim \mu} \left[\left\| S - \frac{v_{\mathbf{N}}}{\mu(\mathbf{N})} \right\|^2 \right]$$

depends on it.

And the μ minimizing it is

$$\forall n, \;\; \mu(n) = \frac{\|v_n\|}{\sum_m \|v_m\|}.$$

Weighting by sampling

So, to apply this idea of importance sampling we need a technical solution to sample according to [an approximation of]

$$\frac{\|v_n\|}{\sum_m \|v_m\|}.$$

The key issue is that computing $\|v_n\|$ is often as expensive as computing v_n.

Weighting by sampling

So, to apply this idea of importance sampling we need a technical solution to sample according to [an approximation of]

$$\frac{\|v_n\|}{\sum_m \|v_m\|}.$$

The key issue is that computing $\|v_n\|$ is often as expensive as computing v_n.

We have developed three algorithms to address this challenge:

- Batch re-sampling for deep learning.
- Importance Sampling Tree.
- Attention sampling networks.

Batch re-sampling

Batch re-sampling

Given a training set $(x_n, y_n), n = 1, \dots, N$, a model f and a loss $\mathcal{L}$, the standard deep learning training procedure is the mini-batch stochastic gradient descent

$$w_{k+1} = w_k - \eta \sum_{n \in B_k} \underbrace{\nabla_{|w_k} \mathcal{L}(f(x_n; w_k), y_n)}_{\nabla_{|w_k}(n)}.$$

Batch re-sampling

Given a training set $(x_n, y_n), n = 1, \dots, N$, a model f and a loss $\mathcal{L}$, the standard deep learning training procedure is the mini-batch stochastic gradient descent

$$w_{k+1} = w_k - \eta \sum_{n \in B_k} \underbrace{\nabla_{|w_k} \mathcal{L}(f(x_n; w_k), y_n)}_{\nabla_{|w_k}(n)} .$$

Computation goes into the evaluation of $f(x_n; w_k)$ (the "forward pass") and the derivative of the loss $\nabla_{|w_k}(n)$ (the "backward pass").

Batch re-sampling

To apply weighting-by-sampling to this mini-batch approach, we propose to:

1. Sample uniformly B examples, compute their importance,
2. re-sample $b < B$ of them to use for the gradient step.

Sampling according to the ideal weight $\|\nabla_{|w_k}(n)\|$ would require the full computation.

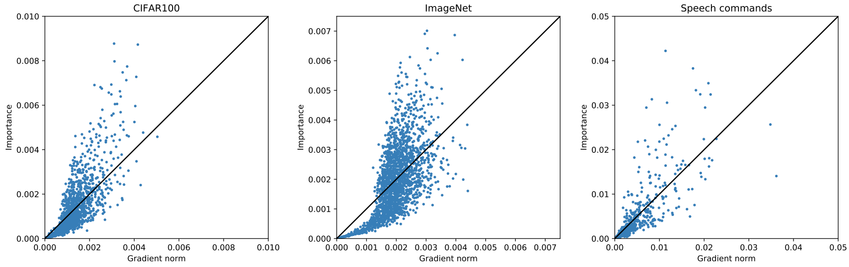
Batch re-sampling

To apply weighting-by-sampling to this mini-batch approach, we propose to:

1. Sample uniformly B examples, compute their importance,
2. re-sample $b < B$ of them to use for the gradient step.

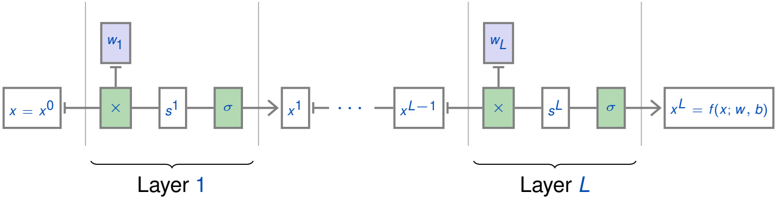
Sampling according to the ideal weight $\|\nabla_{|w_k}(n)\|$ would require the full computation.

Since the backward pass costs twice the forward, we could use the loss (Schaul et al., 2015; Loshchilov and Hutter, 2015), but it happens to be a poor approximation.



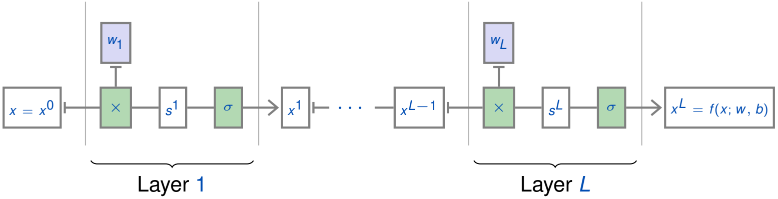
Batch re-sampling

Instead, we propose to use an upper bound of the gradient norm.



Batch re-sampling

Instead, we propose to use an upper bound of the gradient norm.



With

$$\rho = \max_{l,n} \left\| \prod_{k \geq l} \sigma'\left(s^{k-1}\right) W_k^T \right\| \left\| x_n^l \right\|,$$

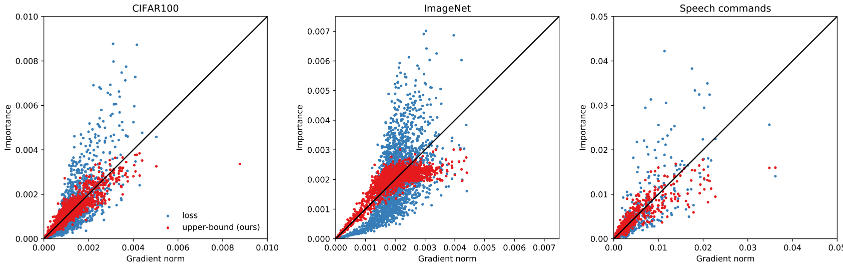
we get

$$\left\| \nabla_{|w_k}(n) \right\| \leq \rho \left\| \underbrace{\sigma'\left(s^L\right) \nabla_{|x_n^L}(n)}_{\text{Gradient wrt penultimate activations}} \right\|.$$

Recent techniques specifically for deep architectures (batchnorm, layernorm, Xavier's init) keep ρ in a reasonable range, and make this result useful in practice.

Batch re-sampling

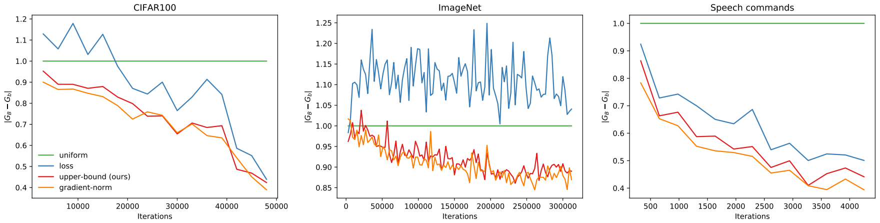
This translates into a better approximation of the gradient norm.



(Katharopoulos and Fleuret, 2018)

Batch re-sampling

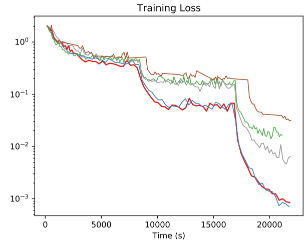
And into a better approximation of the full-batch gradient estimate.

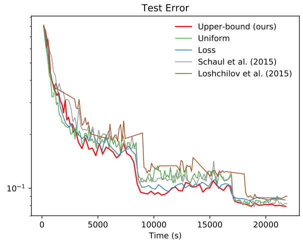


(Katharopoulos and Fleuret, 2018)

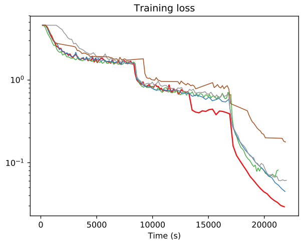
Batch re-sampling

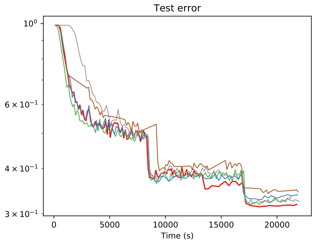
Wide Resnet 28-2 on CIFAR-10





Wide Resnet 28-2 on CIFAR-100





(Katharopoulos and Fleuret, 2018)

Importance Sampling Tree

Importance Sampling Tree

A better approach would be to have an approximation of

$$n \mapsto \frac{\|v_n\|}{\sum_m \|v_m\|}.$$

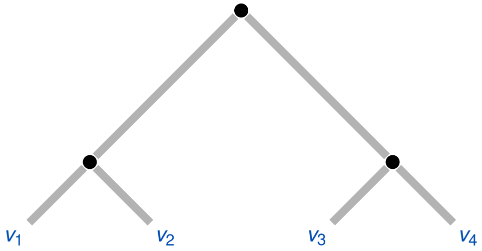
Importance Sampling Tree

A better approach would be to have an approximation of

$$n \mapsto \frac{\|v_n\|}{\sum_m \|v_m\|}.$$

We proposed a algorithm similar to the Monte-Carlo Tree Search, to sample the v_n when N is greater to even be enumerated.

E.g. with $\|v_1\| = 5, \|v_2\| = 5, \|v_3\| = 10, \|v_4\| = 30$:



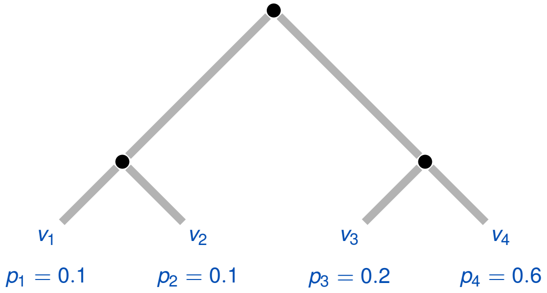
Importance Sampling Tree

A better approach would be to have an approximation of

$$n \mapsto \frac{\|v_n\|}{\sum_m \|v_m\|}.$$

We proposed a algorithm similar to the Monte-Carlo Tree Search, to sample the v_n when N is greater to even be enumerated.

E.g. with $\|v_1\| = 5, \|v_2\| = 5, \|v_3\| = 10, \|v_4\| = 30$:



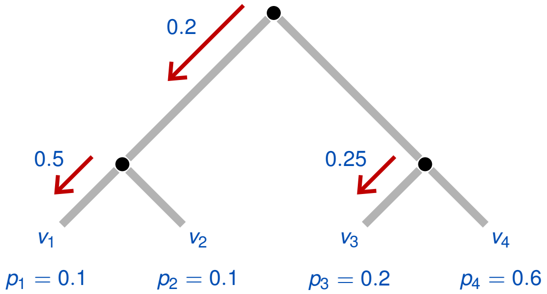
Importance Sampling Tree

A better approach would be to have an approximation of

$$n \mapsto \frac{\|v_n\|}{\sum_m \|v_m\|}.$$

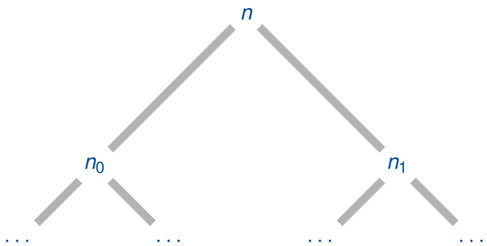
We proposed a algorithm similar to the Monte-Carlo Tree Search, to sample the v_n when N is greater to even be enumerated.

E.g. with $\|v_1\| = 5, \|v_2\| = 5, \|v_3\| = 10, \|v_4\| = 30$:



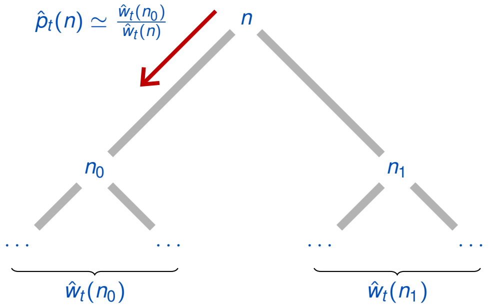
Importance Sampling Tree

Our “Importance Sampling Tree” (Canévet et al., 2016) samples and dynamically estimates the leaf distribution.



Importance Sampling Tree

Our “Importance Sampling Tree” (Canévet et al., 2016) samples and dynamically estimates the leaf distribution.

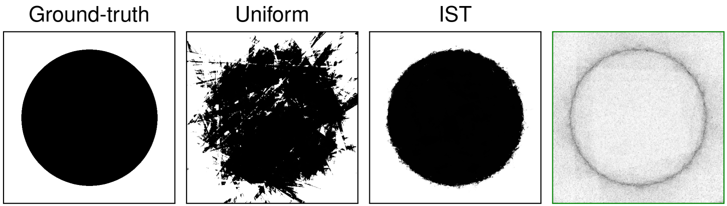


It keeps a running estimate $\hat{w}_t(n)$ of the weights under each node, and an estimate $\hat{p}_t(n)$ of the probability to “go left”.

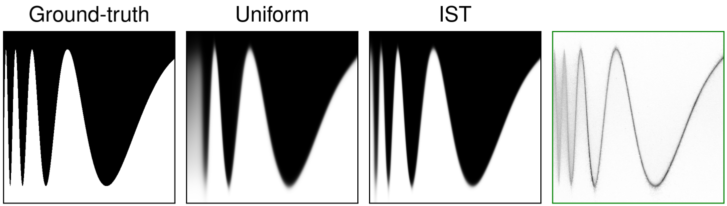
Importance Sampling Tree

We apply it to re-sampling samples according to their gradient norms.

Adaboost

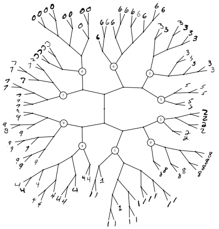


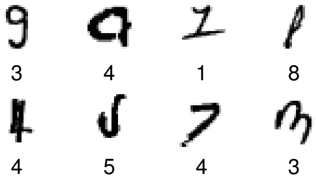
Two layer neural network

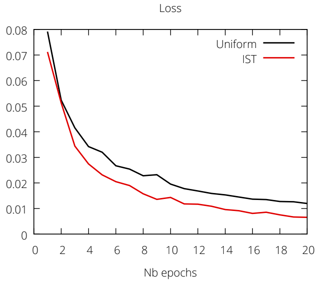


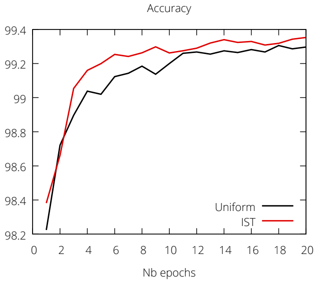
Importance Sampling Tree

Deep neural network on MNIST.









(Canévet et al., 2016)

Attention sampling

Attention Sampling

During inference, there is a similar discrepancy between information content and computational use, and state-of-the-art models are intractable on large signals.

For instance for images, using a ResNeXt101 (Xie et al., 2016):

Input size	**Nb. floating point operations** (10^9)
224×224	22.0
1600×1400	904.5

Certain application domains require to process images in the giga-pixel range. *E.g.* CAMELYON17 images of lymph node sections are $200k \times 100k$.

Attention Sampling

Recent models for NLP and image processing utilize attention mechanisms that modulate the importance of features as a function of the location in the signal, *e.g.*

$$\Psi(x; w) = g\left(\sum_{q=1}^{Q} a(x; w_a)_q \, f(x; w_f)_q\right)$$

where f are the features, $a \in \mathbb{R}_+$ and

$$\sum_{q=1}^{Q} a(x; w_a)_q = 1.$$

Attention Sampling

And as for our previous algorithms, if we sample

$$\mathbf{Q}_1, \dots, \mathbf{Q}_K \text{ i.i.d} \sim a(x; w),$$

we have

$$\sum_{q=1}^{Q} a(x; w)_q \, f(x; w)_q \simeq \frac{1}{K} \sum_{k=1}^{K} f(x; w)_{\mathbf{Q}_k}.$$

And if f is convolutional, it can be computed at sparse locations

$$f(x; w)_q = f(x_{|q}; w).$$

where $x_{|q}$ is a patch extracted at location q.

Attention Sampling

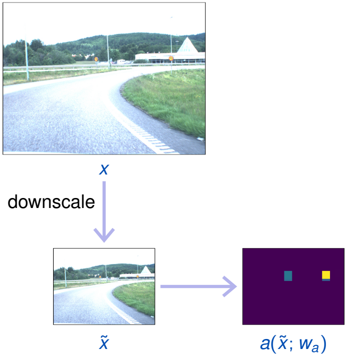
Given an input image x, our “attention sampling” algorithm

1. computes the attention map on a downscaled image $a(\tilde{x}; w_a)$,
2. samples K high-resolution patches $x_{|\mathbf{Q}_1}, \dots, x_{|\mathbf{Q}_K}$,
3. computes the final response $g\left(\sum_{k=1}^{K} f(x_{|\mathbf{Q}_k}; w_f)\right)$.

Attention Sampling

Given an input image x, our "attention sampling" algorithm

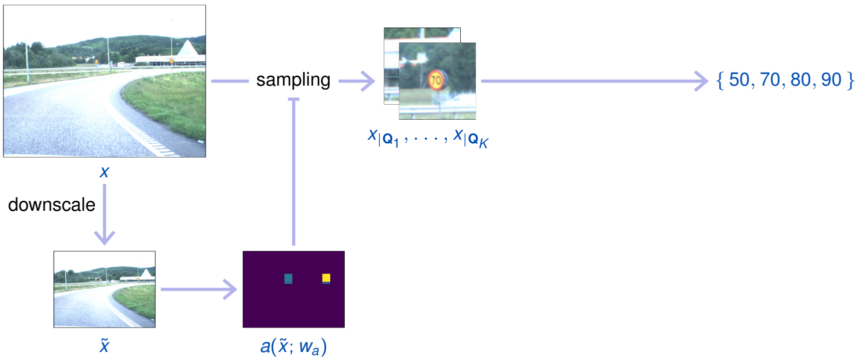
1. computes the attention map on a downscaled image $a(\tilde{x}; w_a)$,
2. samples K high-resolution patches $x_{|\mathbf{Q}_1}, \dots, x_{|\mathbf{Q}_K}$,
3. computes the final response $g\left(\sum_{k=1}^{K} f(x_{|\mathbf{Q}_k}; w_f)\right)$.



Attention Sampling

Given an input image x, our “attention sampling” algorithm

1. computes the attention map on a downscaled image $a(\tilde{x}; w_a)$,
2. samples K high-resolution patches $x_{|\mathbf{Q}_1}, \dots, x_{|\mathbf{Q}_K}$,
3. computes the final response $g\left(\sum_{k=1}^{K} f(x_{|\mathbf{Q}_k}; w_f)\right)$.



Attention Sampling

The two networks $a(.; w_a)$ and $f(.; w_f)$ are trained end-to-end jointly by propagating the gradient w.r.t w_a through the sampling.

With $\mathbf{Q} \sim a(\tilde{x}; w_a)$:

$$\frac{\partial}{\partial w_f} \mathbb{E}\Big[f\big(x_{|\mathbf{Q}}; w_f\big) \Big] = \mathbb{E}\left[\frac{\partial}{\partial w_f} f\big(x_{|\mathbf{Q}}; w_f\big) \right]$$

Attention Sampling

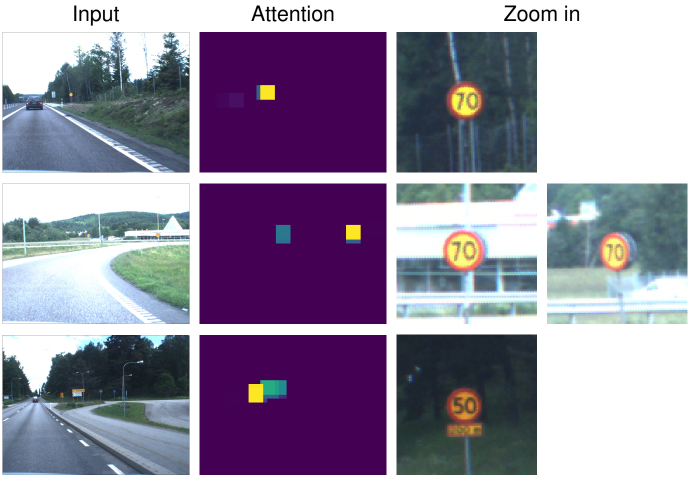
The two networks $a(.; w_a)$ and $f(.; w_f)$ are trained end-to-end jointly by propagating the gradient w.r.t w_a through the sampling.

With $\mathbf{Q} \sim a(\tilde{x}; w_a)$:

$$\frac{\partial}{\partial w_f} \mathbb{E}\Big[f\big(x_{|\mathbf{Q}}; w_f\big) \Big] = \mathbb{E}\left[\frac{\partial}{\partial w_f} f\big(x_{|\mathbf{Q}}; w_f\big) \right],$$

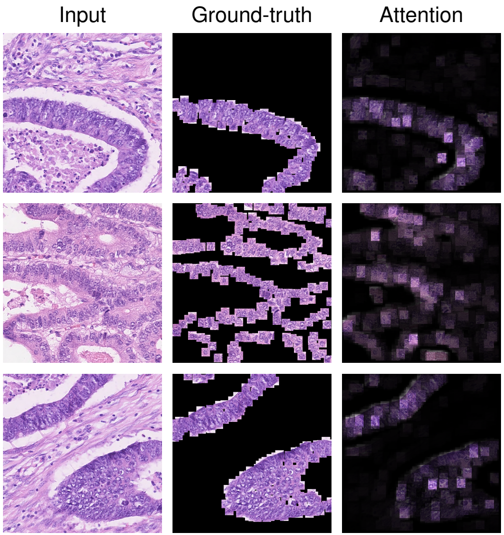
$$\begin{aligned}
\frac{\partial}{\partial w_a} \mathbb{E}\Big[f\big(x_{|\mathbf{Q}}; w_f\big) \Big] &= \frac{\partial}{\partial w_a} \sum_{q=1}^{Q} a(\tilde{x}; w_a)_q \, f(x_{|q}; w_f) \\
&= \sum_{q=1}^{Q} a(\tilde{x}; w_a)_q \frac{\frac{\partial}{\partial w_a} a(\tilde{x}; w_a)_q}{a(\tilde{x}; w_a)_q} f(x_{|q}; w_f) \\
&= \mathbb{E}\left[\frac{\partial}{\partial w_a} \log(a(\tilde{x}; w_a)_{\mathbf{Q}}) \, f\big(x_{|\mathbf{Q}}; w_f\big) \right].
\end{aligned}$$

Attention Sampling



(Katharopoulos and Fleuret, 2019)

Attention Sampling



(Katharopoulos and Fleuret, 2019)

Attention Sampling

Speed limit road sign dataset

Method	Scale	Test Error	Time/sample	Memory/sample
CNN	0.3	0.311 ± 0.049	6.6 ms	86 MB
CNN	1	0.247 ± 0.001	64.2 ms	958 MB
U-5	0.3/1	0.531 ± 0.004	7.8 ms	39 MB
U-10	0.3/1	0.472 ± 0.008	10.8 ms	78 MB
MIL*	1	0.083 ± 0.006	97.2 ms	1,497 MB
ATS-5†	0.3/1	0.089 ± 0.002	8.5 ms	86 MB
ATS-10†	0.3/1	0.095 ± 0.008	10.3 ms	118 MB

Colon cancer dataset

Method	Scale	Test Error	Time/sample	Memory/sample
CNN	0.5	0.104 ± 0.009	4.8 ms	65 MB
CNN	1	0.092 ± 0.012	18.7 ms	250 MB
U-10	0.2/1	0.156 ± 0.006	1.8 ms	19 MB
U-50	0.2/1	0.124 ± 0.010	4.6 ms	24 MB
MIL*	1	0.093 ± 0.004	48.5 ms	644 MB
ATS-10†	0.2/1	0.093 ± 0.014	1.8 ms	21 MB
ATS-50†	0.2/1	0.093 ± 0.019	4.5 ms	26 MB

*Ilse et al. (2018), †Katharopoulos and Fleuret (2019)

Conclusion

This approach has an enormous practical potential:

- Weighting-by-sampling works in practice in a clear formal framework.
- The trend toward larger models does not seem to slow down.
- Recent state-of-the-art approaches are attention-based.
- Lots of promising applications of ML involve very high dimensions signal (particle physics, astronomy, microscopy, satellite imaging).

Conclusion

This approach has an enormous practical potential:

- Weighting-by-sampling works in practice in a clear formal framework.
- The trend toward larger models does not seem to slow down.
- Recent state-of-the-art approaches are attention-based.
- Lots of promising applications of ML involve very high dimensions signal (particle physics, astronomy, microscopy, satellite imaging).

The scientific challenges are exciting:

- How to relate minimal computation and generalization?
- Are there fundamental computational bounds?
- How to deal with physical constraints and locality?
- Models have to be re-imagined for computation-by-sampling. What is the resnet or the batchnorm for it?

The end

References

J. Ba and R. Caruana. Do deep nets really need to be deep? In *Neural Information Processing Systems (NIPS)*, pages 2654–2662. 2014.

O. Canévet, C. Jose, and F. Fleuret. Importance sampling tree for large-scale empirical expectation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1454–1462, 2016.

A. Canziani, A. Paszke, and E. Culurciello. An analysis of deep neural network models for practical applications. *CoRR*, abs/1605.07678, 2016.

M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser. Universal transformers. *CoRR*, abs/1807.03819, 2018.

C. Dubout and F. Fleuret. Exact acceleration of linear object detectors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 301–311, 2012.

C. Dubout and F. Fleuret. Accelerated training of linear object detectors. In *Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 572–577, 2013.

E. Dupont, A. Doucet, and Y. Teh. Augmented neural odes. *CoRR*, abs/1904.01681, 2019.

D. Gershgorn. The data that transformed ai researchand possibly the world, July 2017.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5mb model size. *CoRR*, abs/1602.07360, 2016.

M. Ilse, J. Tomczak, and M. Welling. Attention-based deep multiple instance learning. In *International Conference on Machine Learning (ICML)*, volume 80, pages 2127–2136, 2018.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

C. Jose, M. Cissé, and F. Fleuret. Kronecker recurrent units. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2380–2389, 2018.

A. Katharopoulos and F. Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2525–2534, 2018.

A. Katharopoulos and F. Fleuret. Processing megapixel images with deep attention-sampling models. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3282–3291, 2019.

I. Loshchilov and F. Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

J. Newling and F. Fleuret. Fast k-means with accurate bounds. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 936–944, 2016a.

J. Newling and F. Fleuret. Fast mini-batch k-means by nesting. In *Proceedings of the international conference on Neural Information Processing Systems (NIPS)*, pages 1352–1360, 2016b.

J. Newling and F. Fleuret. A sub-quadratic exact medoid algorithm. In *Proceedings of the international conference on Artificial Intelligence and Statistics (AISTATS)*, pages 185–193, 2017. (Best paper award).

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell. Clockwork convnets for video semantic segmentation. *CoRR*, abs/1608.03609, 2016.

L. Smith and N. Topin. Super-convergence: Very fast training of neural networks using large learning rates. *CoRR*, abs/1708.07120, 2017.

M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431.pdf, 2016.

B. Zoph and Q. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016.