Machine Learning Systems Design

AdamW Optimizer

First introduced in 2014, a simple and intuitive idea:

“Why use the same learning rate for every parameter, when we know that some surely need to be moved further and faster than others?”

Since the square of recent gradients tells us how much signal we’re getting for each weight, we can just divide by that to ensure even the most sluggish weights get their chance to shine.

Adam takes that idea, adds on the standard approach to momentum, and (with a little tweak to keep early batches from being biased) that’s it!

https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html

AdamW Optimizer

When Adam was first released, the deep learning community was full of excitement and it became mainstream.

But it started to become clear that all was not as we hoped. Few research articles used it to train their models, new studies began to clearly discourage to apply it and showed on several experiments that plain SGD with momentum was performing better.

https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html

AdamW Optimizer

But at the end of 2017, Adam seemed to get a new lease of life.

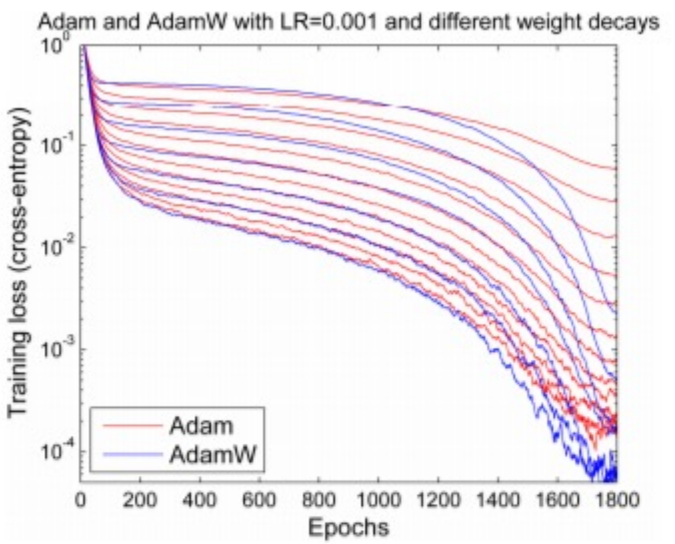
It was shown that the way weight decay is implemented in Adam in every library seems to be wrong*, and proposed a simple way (which they call AdamW) to fix it.

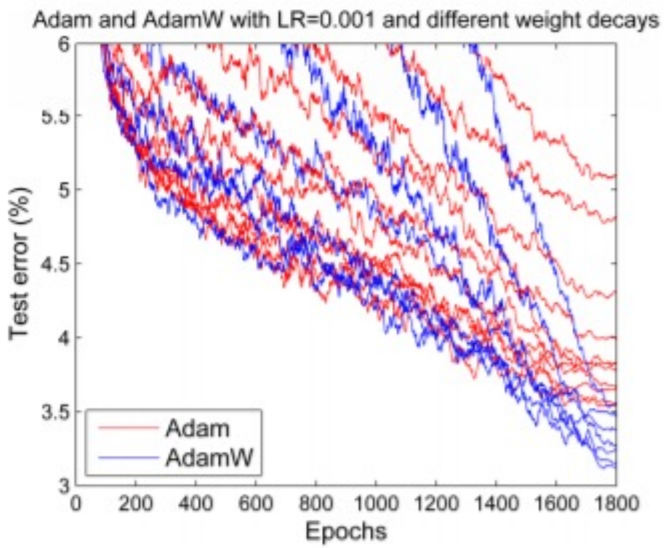
*Loshchilov, I. and Hutter, F., 2017. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html

AdamW Optimizer

Although their results were slightly mixed, they did show some encouraging charts, such as this one:





However, without broad framework availability, day-to-day practitioners were stuck with the old, “broken” Adam

*Loshchilov, I. and Hutter, F., 2017. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html

AdamW Optimizer

Two separate papers pointed out apparent problems with the convergence proof of poor Adam, although one of them claimed a fix (and won a “best paper” award at the prestigious ICLR conference), which they called amsgrad.

But if we’ve learned anything from this potted history of this most dramatic life (at least, dramatic by optimizer standards), it’s that nothing is as it seems.

It has been pointed out that the claimed convergence problems are because of poorly chosen hyper-parameters, and amsgrad may not be a fix.

https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html

AdamW Optimizer

Properly tuned, Adam really works!
We got new state of the art results (in terms of training time) on various tasks like:

- Training CIFAR10 to >94% accuracy in as few as 18 epochs with Test Time Augmentation or with 30 epochs without, as in the DAWNBench competition.
- Fine-tuning Resnet50 to 90% accuracy on the Cars Stanford Dataset in just 60 epochs (previous reports to the same accuracy used 600);
- training from scratch an AWD LSTM or QRNN in 90 epochs (1.5 hours on a single GPU) to state-of-the-art perplexity on Wikitext-2 (previous reports used 750 for LSTMs, 500 for QRNNs).

https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html

AdamW Optimizer

- That means that we’ve seen (for the first time we’re aware of) super convergence using Adam! Super convergence is a phenomenon that occurs when training a neural net with high learning rates, growing for half the training. Before it was understood, training CIFAR10 to 94% accuracy took about 100 epochs.
- In contrast to previous work, we see Adam getting about as good accuracy as SGD+Momentum on every CNN image problem we’ve tried it on, as long as it’s properly tuned, and it’s nearly always a bit faster too.
- The suggestions that amsgrad are a poor “fix” are correct. We consistently found that amsgrad didn’t achieve any gain in accuracy (or other relevant metric) than plain Adam/AdamW.

https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html

AdamW Optimizer

- When you hear people saying that Adam doesn't generalize as well as SGD+Momentum, you'll nearly always find that they're choosing poor hyper-parameters for their model.

- Adam generally requires more regularization than SGD, so be sure to adjust your regularization hyper-parameters when switching from SGD to Adam.

https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html

Warm-up

- Warmup steps are a few updates with low learning rate before / at the beginning of training.
- After this warmup, you use the regular learning rate (schedule) to train your model to convergence.
- The main reason for warmup steps is to allow adaptive optimisers such as Adam and RMSProp to compute correct statistics of the gradients.
- Therefore, a warmup period is not useful when training with plain SGD.

Warm-up

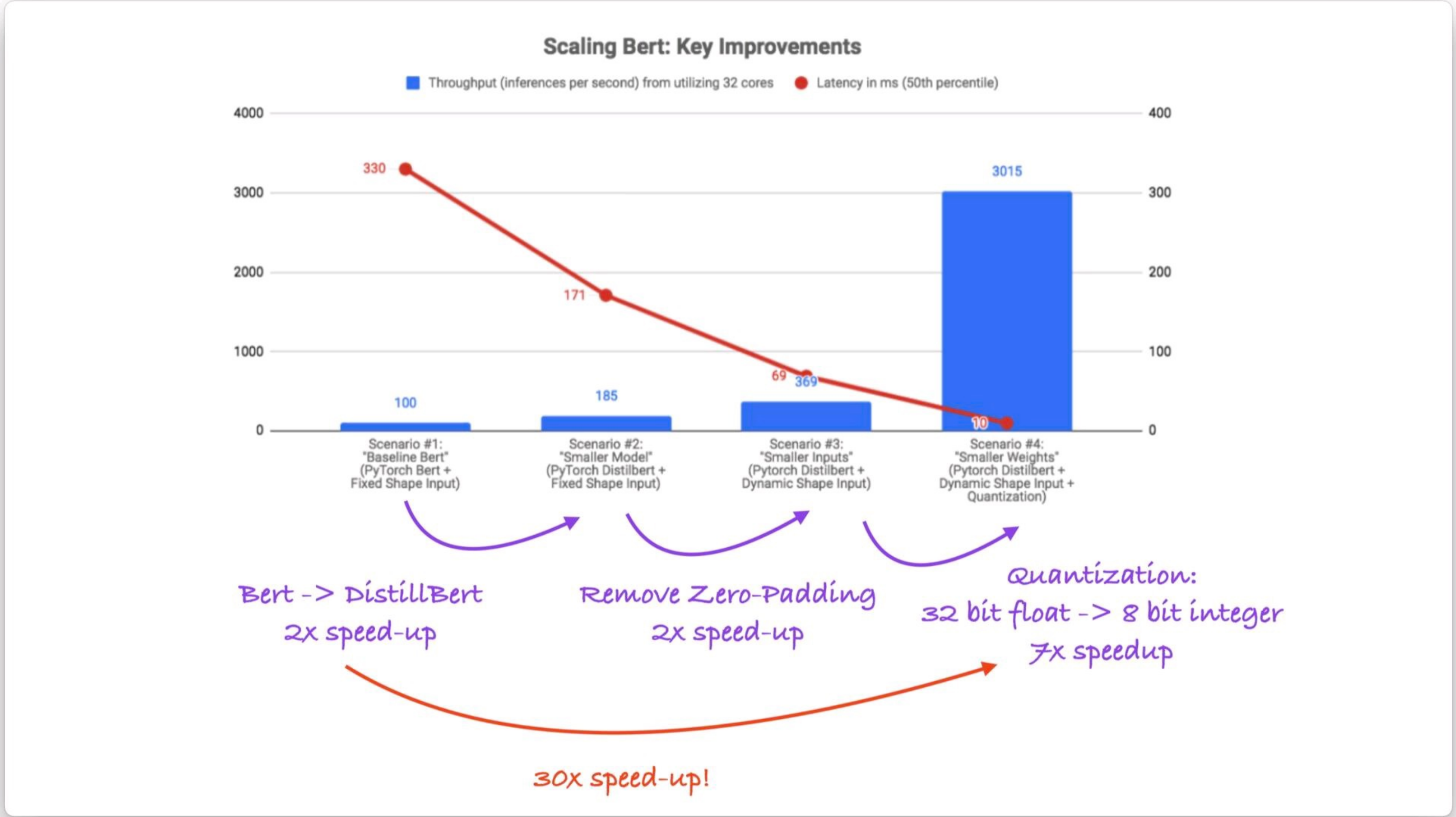
- For example: RMSProp computes a moving average of the squared gradients to get an estimate of the variance in the gradients for each parameter.

- For the first update, the estimated variance is just the square root of the sum of the squared gradients for the first batch.

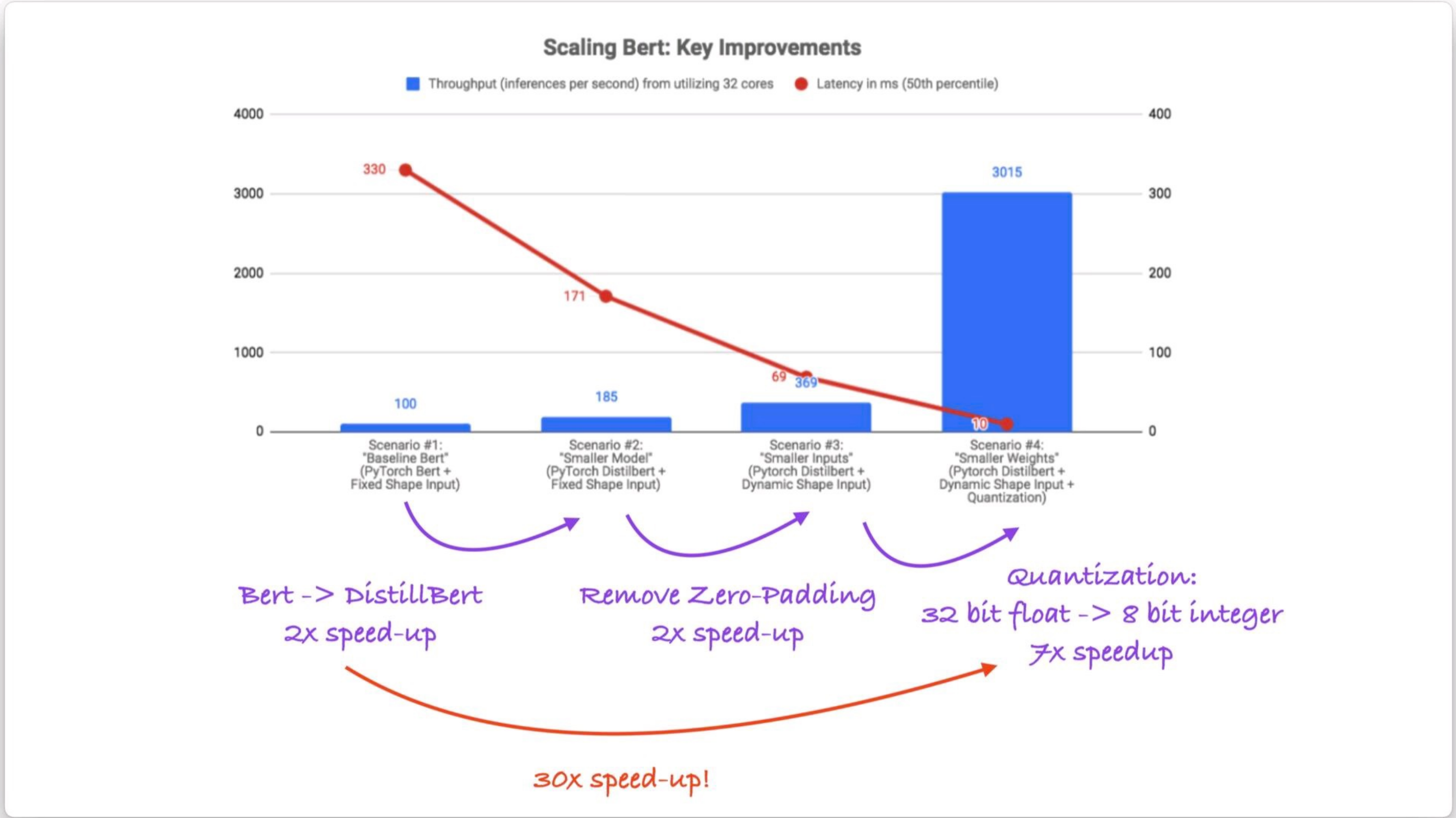
- Since, in general, this will not be a good estimate, your first update could push your network in a wrong direction.

- To avoid this problem, you give the optimiser a few steps to estimate the variance while making as little changes as possible (low learning rate) and only when the estimate is reasonable, you use the actual (high) learning rate.

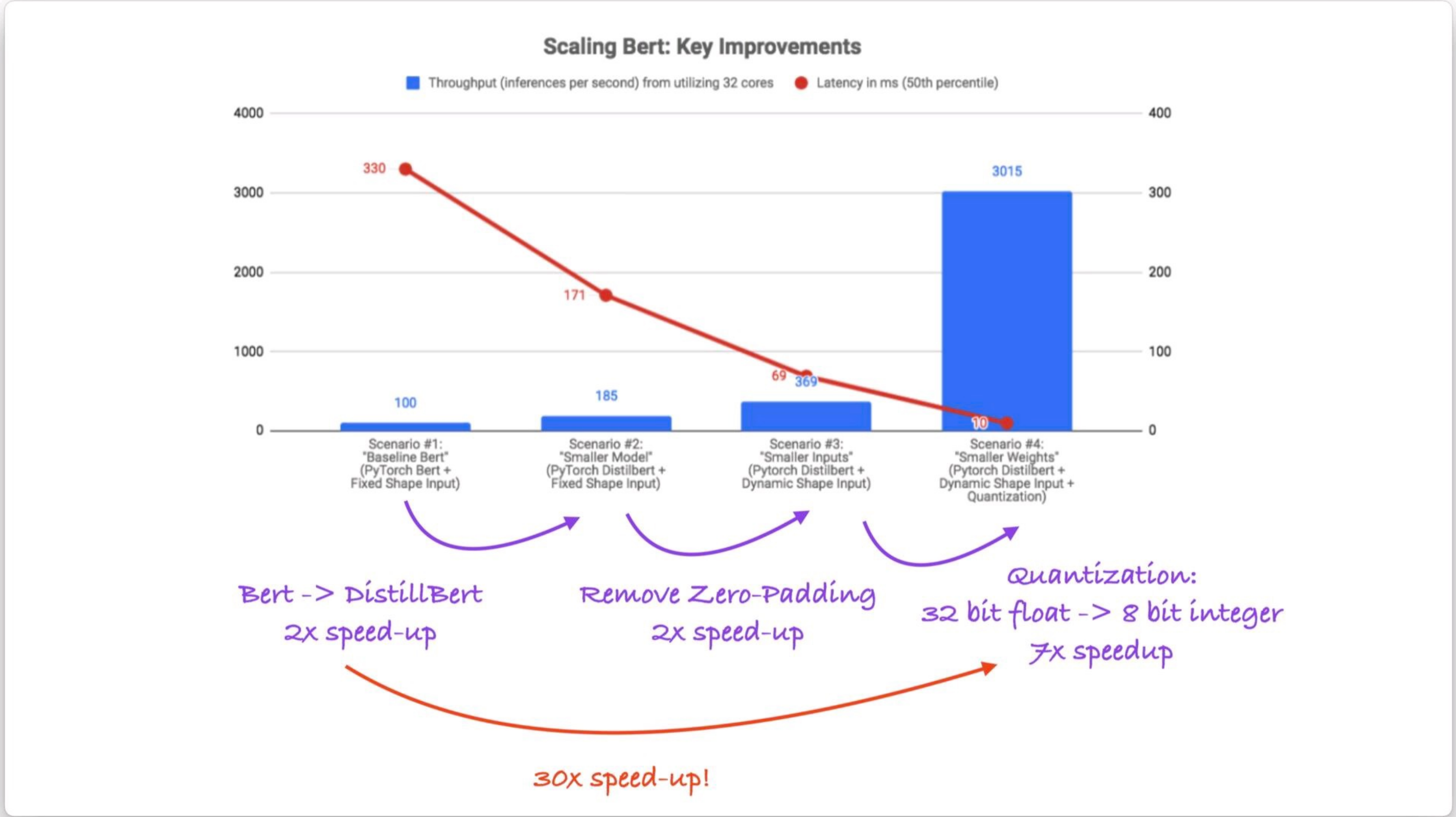
Scaling up BERT-like model Inference on modern CPU



Scaling up BERT-like model Inference on modern CPU



Scaling up BERT-like model Inference on modern CPU



Sparsity

https://neuralmagic.com/blog/how-neural-magics-deep-sparse-technology-works/

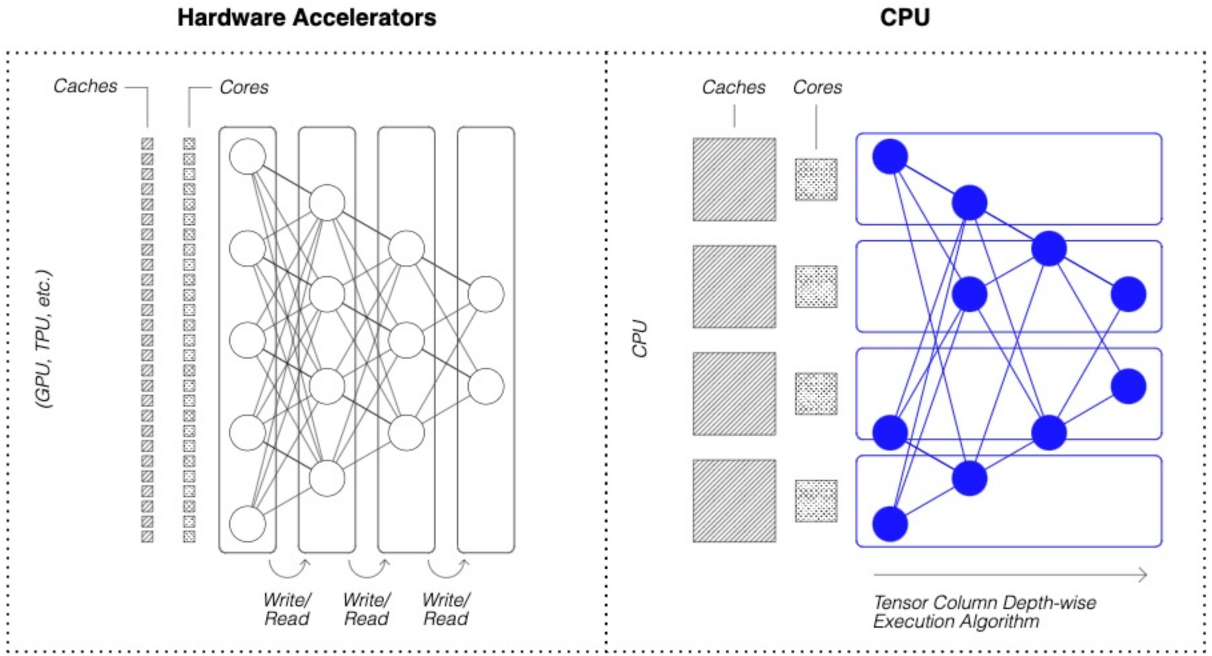
Sparsity

https://github.com/neuralmagic/sparsify

https://github.com/neuralmagic/sparseml

Sparsity

https://docs.neuralmagic.com/deepsparse/



Left: Hardware Accelerators — Standard execution, synchronously layer by layer to maximize parallelism, reading and writing to memory using high bandwidth.

Right: CPU — Deeply sparsify the network to reduce compute…and execute it depth-wise, asynchronously, and fully inside the large CPU caches.

Sparsity

https://www.youtube.com/watch?v=hk-6j6hgsVU&ab_channel=NeuralMagic