Optimization

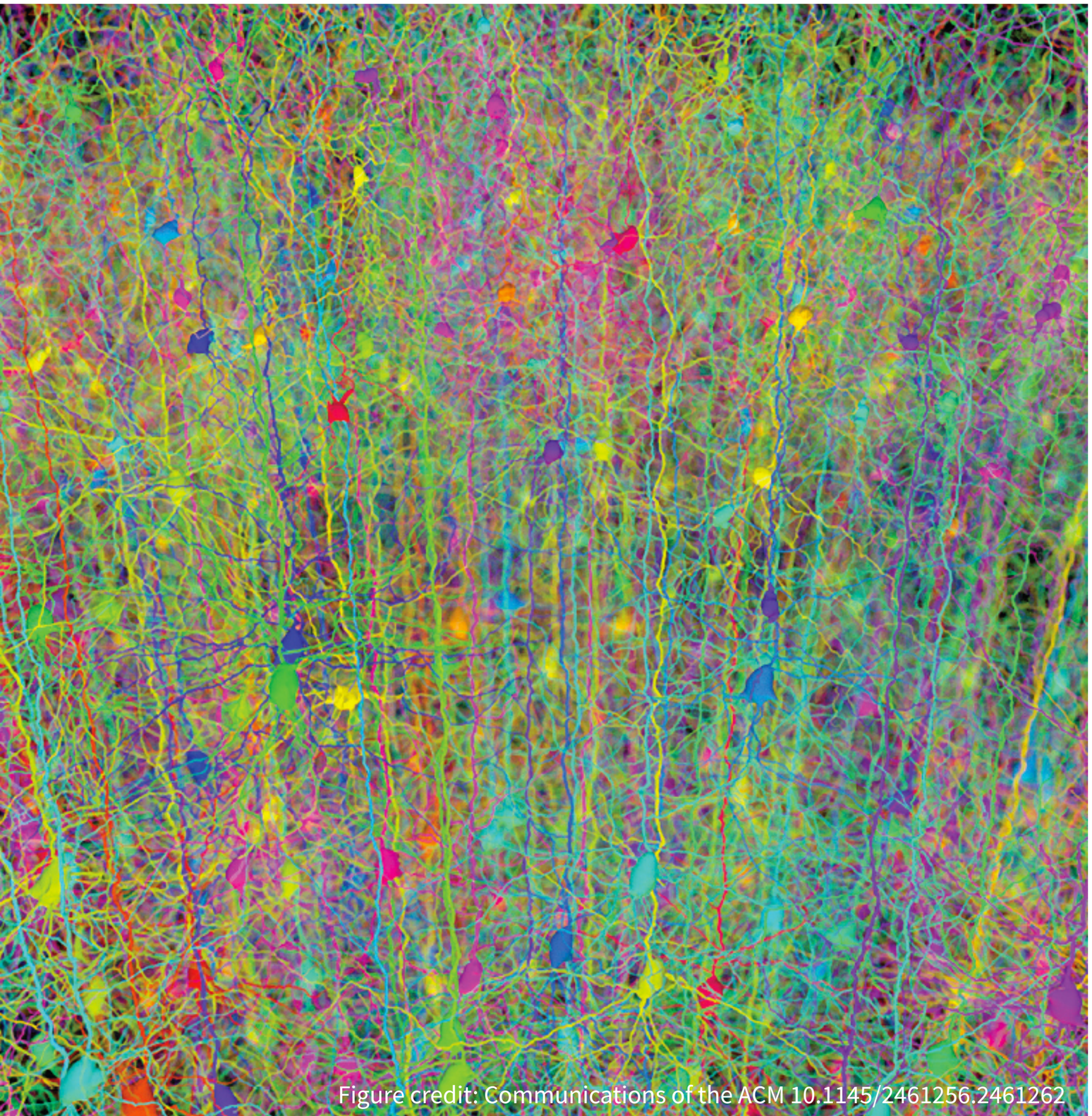


Figure credit: Communications of the ACM 10.1145/2461256.2461262

GRAHAM TAYLOR

VECTOR INSTITUTE

SCHOOL OF ENGINEERING
UNIVERSITY OF GUELPH

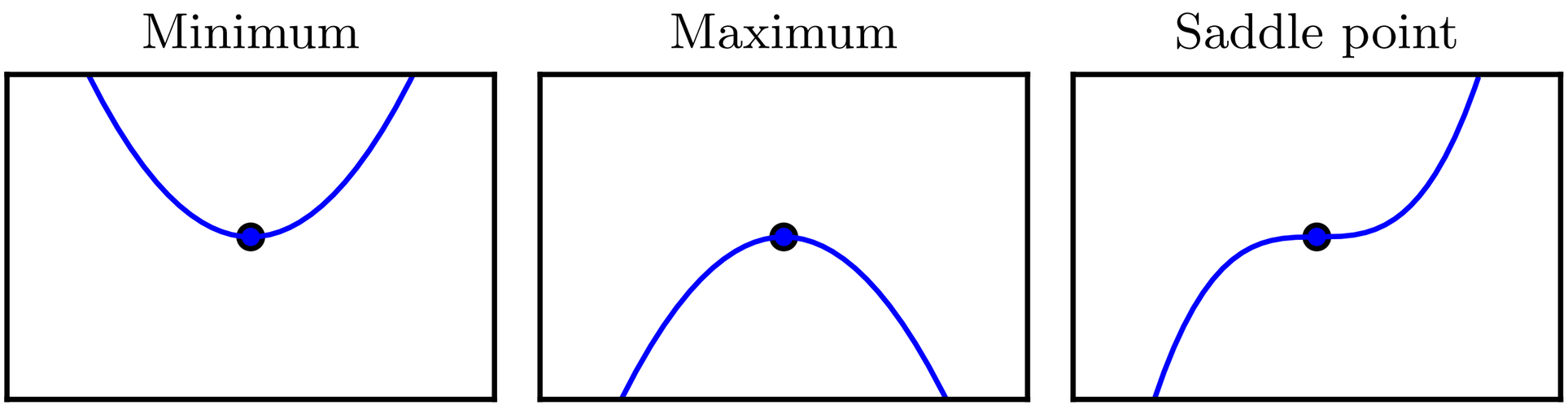
CANADIAN INSTITUTE
FOR ADVANCED RESEARCH

UNIVERSITY of GUELPH
CHANGING LIVES
IMPROVING LIFE

CIFAR
CANADIAN
INSTITUTE
FOR
ADVANCED
RESEARCH

Optimization in Deep Learning

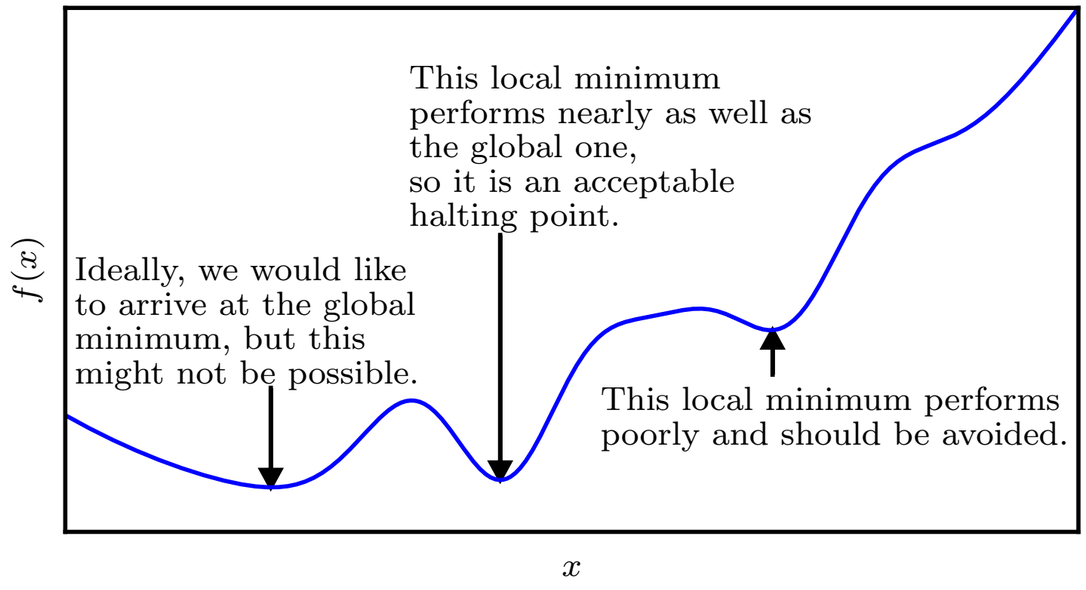
- There isn't a single global optimum (non-convex optimization)
- Non-identifiability:
  - We can permute the hidden units (with their connections) and get the same function
  - We can re-scale the inputs to a ReLU unit as long as we appropriately re-scale the outputs
- We have to deal with local minima, saddle points, and plateaus



Reproduced from Goodfellow et al.

Historical vs. Modern Thinking

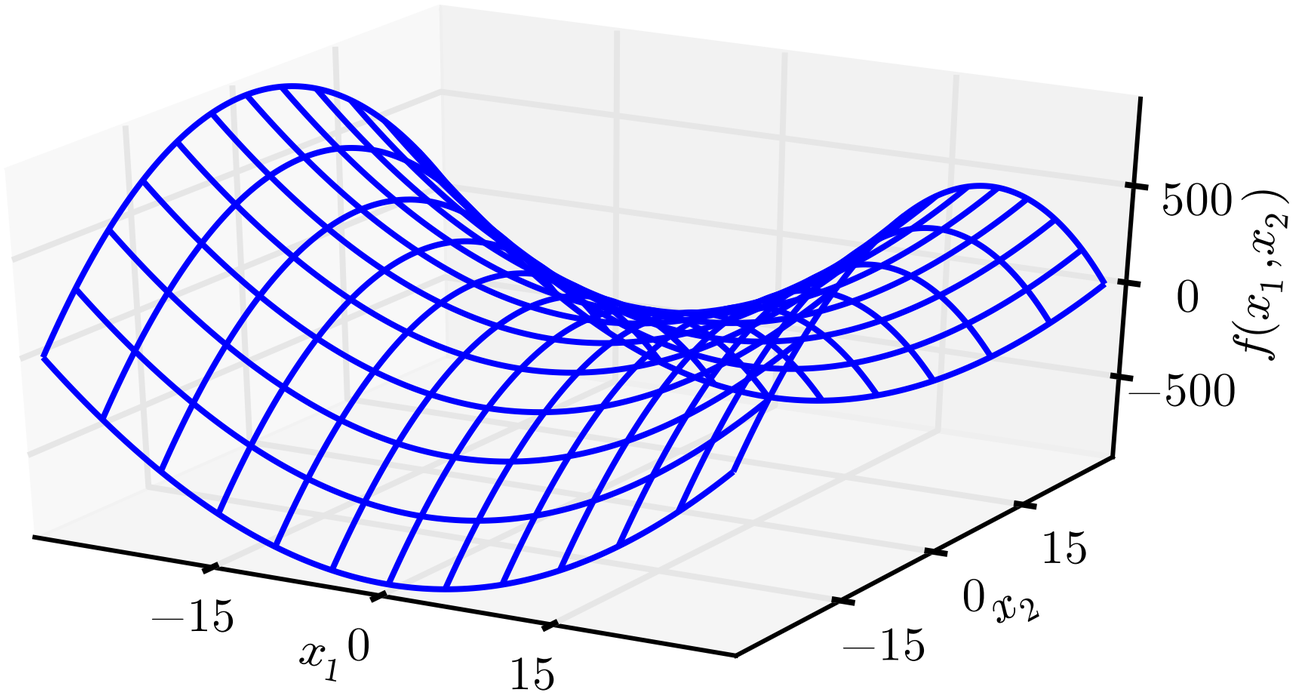
- Pre-2012: “The problem with neural nets is that they get stuck in local minima”
- Now: the local minima arising from nonidentifiability are equivalent to each other in cost function value
  - As a result, they’re not a problematic form of nonconvexity
- Open issue: Do cost surfaces have many local minima of high cost, and do optimization algorithms encounter them?



Reproduced from Goodfellow et al.

Saddle Points

- Many classes of random functions exhibit the following behaviour:
  - local minima are rare
  - saddle points are more common
- It has been shown experimentally that real neural networks also have loss functions that contain many high-cost saddle points
- What are the implications?
  - for first order methods, still unclear
  - but, second order methods seem to be attracted to saddle points



Reproduced from Goodfellow et al.

Convergence Conditions

- Stochastic gradient descent will converge if:

$$\sum_{k=1}^{\infty} \epsilon_k = \infty, \text{and}$$

$$\sum_{k=1}^{\infty} \epsilon_k^2 < \infty$$

- Several recipes for decreasing the learning rate:

$$\epsilon_k = \frac{\epsilon_0}{1 + \delta k}$$

$$\epsilon_k = \frac{\epsilon_0}{k^\delta} \quad 0.5 < \delta \leq 1$$

$$\epsilon_k = (1 - \alpha)\epsilon_0 + \alpha\epsilon_\tau \quad \alpha = \frac{k}{\tau}$$

Batch and Minibatch Algorithms

- Optimization algorithms that use the entire training set are called batch or deterministic gradient methods
  - process examples simultaneously in a large “batch”
  - a lot of computation to get one gradient update
- Optimization algorithms that use only a single example are called stochastic and sometimes online methods
  - noisy estimates of the true gradient
- Most algorithms used for deep learning fall in-between, using more than one, but fewer than all training examples
  - traditionally called minibatch, but commonly called stochastic methods

Reproduced from Goodfellow et al.

Choosing Minibatch Sizes

- Larger batches provide a more accurate estimate of the gradient, but require more computation
- Multicore architectures are underutilized by extremely small batches
- When processing all examples in a batch in parallel, amount of memory scales with batch size
- Some kinds of hardware (e.g. GPUs) benefit with specific sizes of arrays; powers of 2 are common (typically 32 to 256)
- Small batches can offer some regularizing effect

Reproduced from Goodfellow et al.

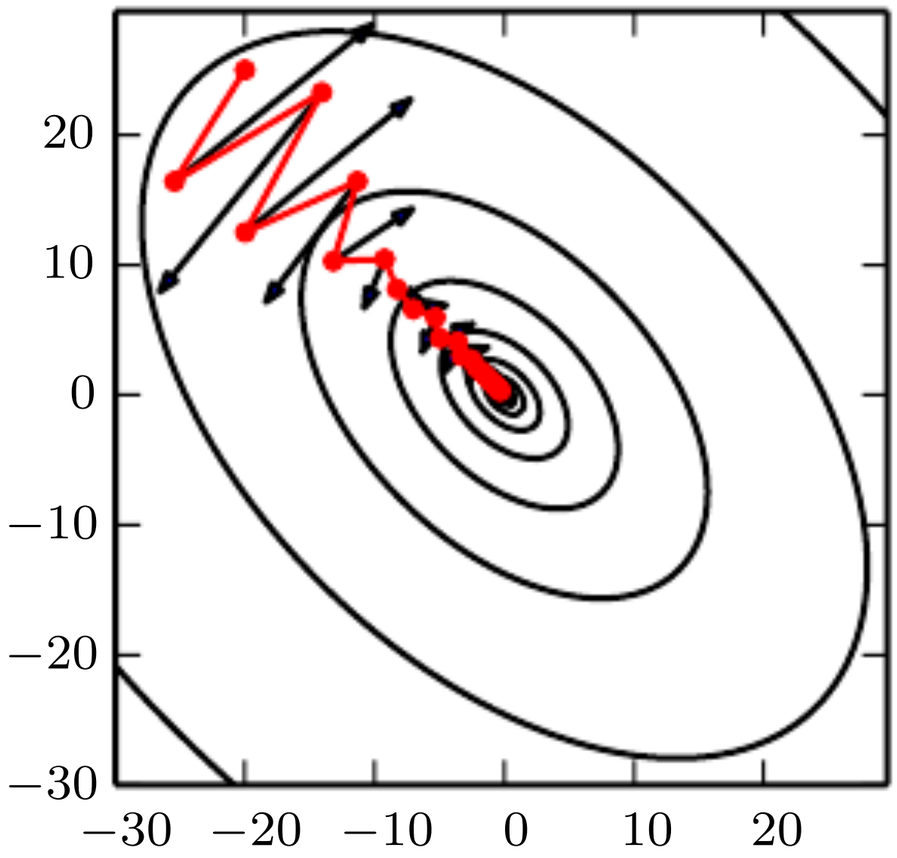
Momentum

- Use an exponential average of previous gradients:

$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) \right)$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$$

- Aims primarily to solve two problems:
  - poor conditioning of the Hessian
  - variance of the stochastic gradients



Reproduced from Goodfellow et al.

Parameter Initialization

- Modern strategies are simple and heuristic
- Only property known with complete certainty is to "break symmetry"
- The goal of having each unit compute a different function motivates initialization with small random values

- Biases are typically set to heuristic chosen constants and only the weights are chosen randomly (almost always Gaussian or uniform)
- Common recipe (Glorot and Bengio 2010):

$$W_{i,j} \sim U\left(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}}\right)$$

m - # of inputs ("fan-in")

n - # of outputs ("fan-out")

Reproduced from Goodfellow et al.

Choosing the Right Optimization Algorithm

- Unfortunately, there is no consensus
- See Unit Tests for Stochastic Optimization (Schaul et al. 2014)
  - results suggest that the family of algorithms with adaptive learning rates performed fairly robustly
- Currently, the most popular optimization algorithms actively in use include:
  - SGD, SGD with momentum
  - RMSProp, RMSProp with momentum
  - AdaDelta and Adam

Reproduced from Goodfellow et al.

Batch Normalization (Ioffe and Szegedy 2015)

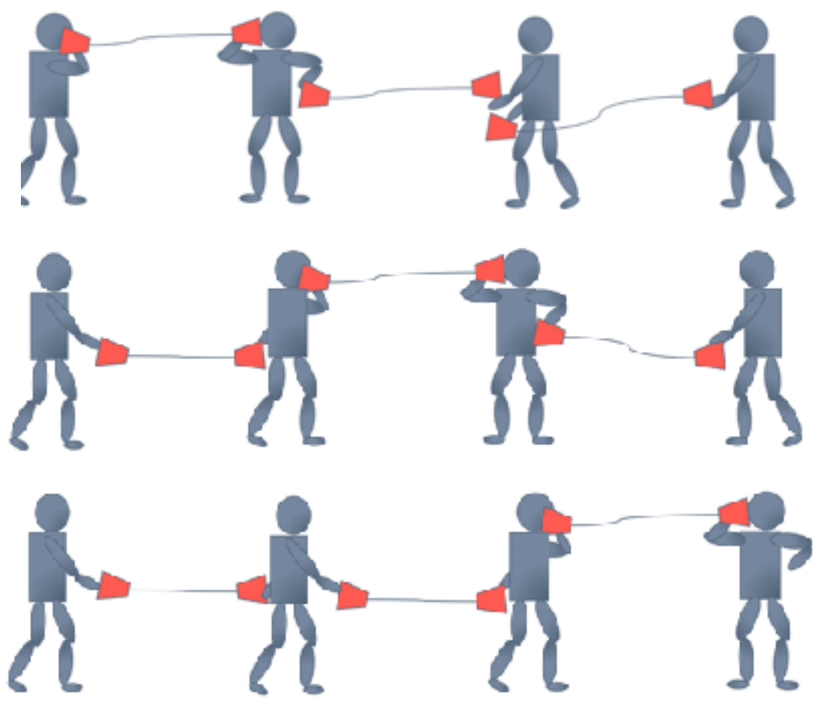
- A method of adaptive reparametrization, motivated by the difficulty of training very deep models
- Significantly reduces the problem of coordinating updates across many layers ("internal covariate shift")
- Reparametrize a minibatch of activations at any layer:

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W}$$

$$\tilde{\boldsymbol{Z}} = \boldsymbol{Z} - \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{Z}_{i,:}$$

$$\hat{\boldsymbol{Z}} = \frac{\tilde{\boldsymbol{Z}}}{\sqrt{\epsilon + \frac{1}{m}\sum_{i=1}^{m} \tilde{\boldsymbol{Z}}_{i,:}^2}}$$

$$\boldsymbol{H} = \max\{0, \boldsymbol{\gamma}\hat{\boldsymbol{Z}} + \boldsymbol{\beta}\}$$



via lab41.org