“Deep” CNNs for Image Classification: Catalyst for Deep Learning Revolution

Danna Gurari
University of Colorado Boulder
Spring 2025

https://dannagurari.colorado.edu/course/neural-networks-and-deep-learning-spring-2025/

Review

- Last lecture:
  - History of Convolutional Neural Networks (CNNs)
  - CNNs – Convolutional Layers
  - CNNs – Pooling Layers
  - Pioneering CNN model: LeNet

- Assignments (Canvas)
  - Lab assignment 1 due in 1 week

- Questions?

Today’s Topics

- Key challenge: training large capacity, deep models
- AlexNet: key tricks for going 8 layers deep
- ResNet: key tricks for extending to 152 layers deep
- Programming tutorial

Today’s Topics

- Key challenge: training large capacity, deep models
- AlexNet: key tricks for going 8 layers deep
- ResNet: key tricks for extending to 152 layers deep
- Programming tutorial

Motivating Task: Predict Category from 1000 Options

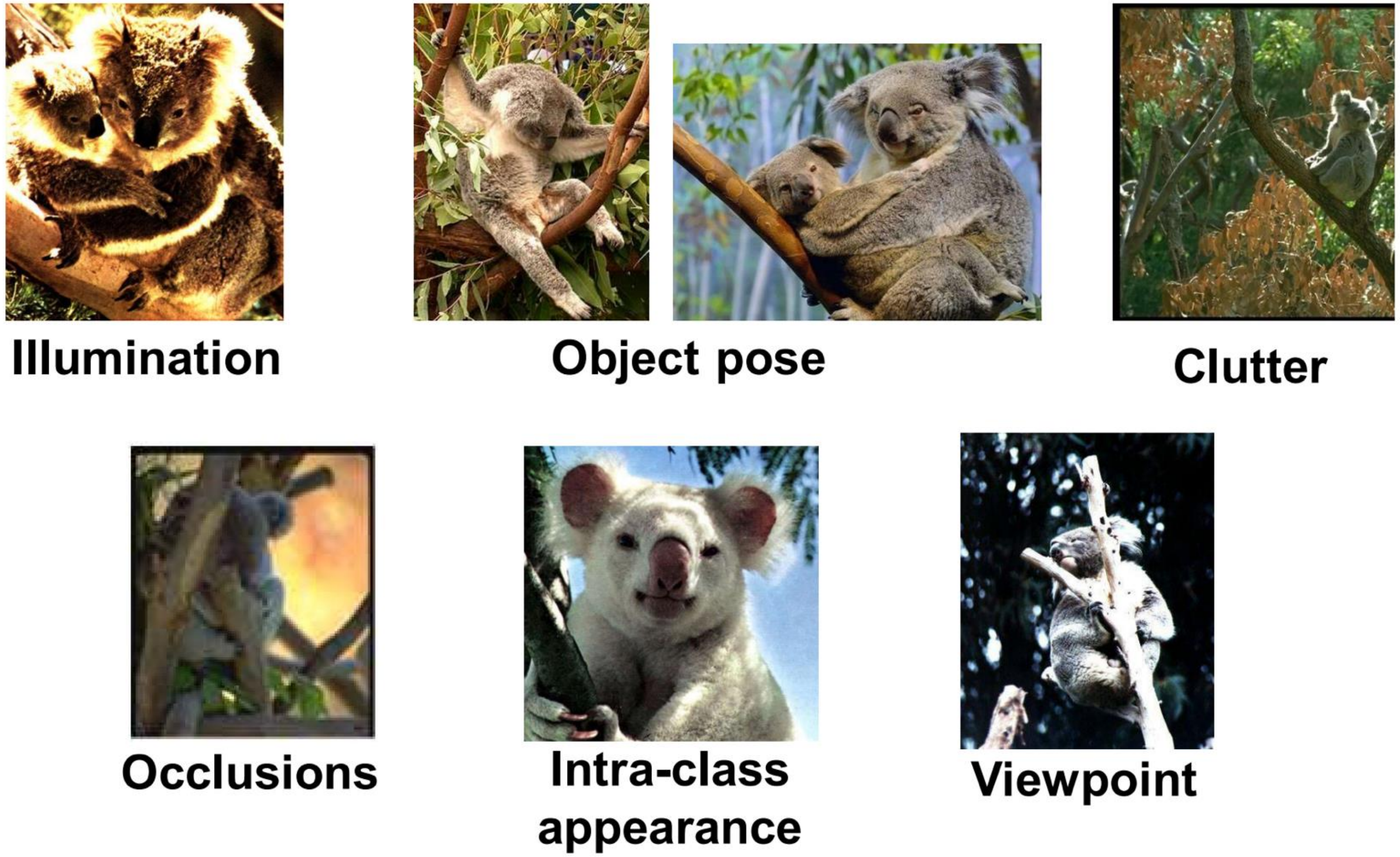
Is this a multi-label or a multi-class classification problem?

- Dataset: ~1.5 million images of objects in natural backgrounds
- Source: images scraped from search engines, such as Flickr, and labeled by crowdworkers
- Evaluation metric: % correct (top-1 and top-5 predictions)

J. Deng, W. Dong, R. Socher, L. Li, K. Li and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. 2009

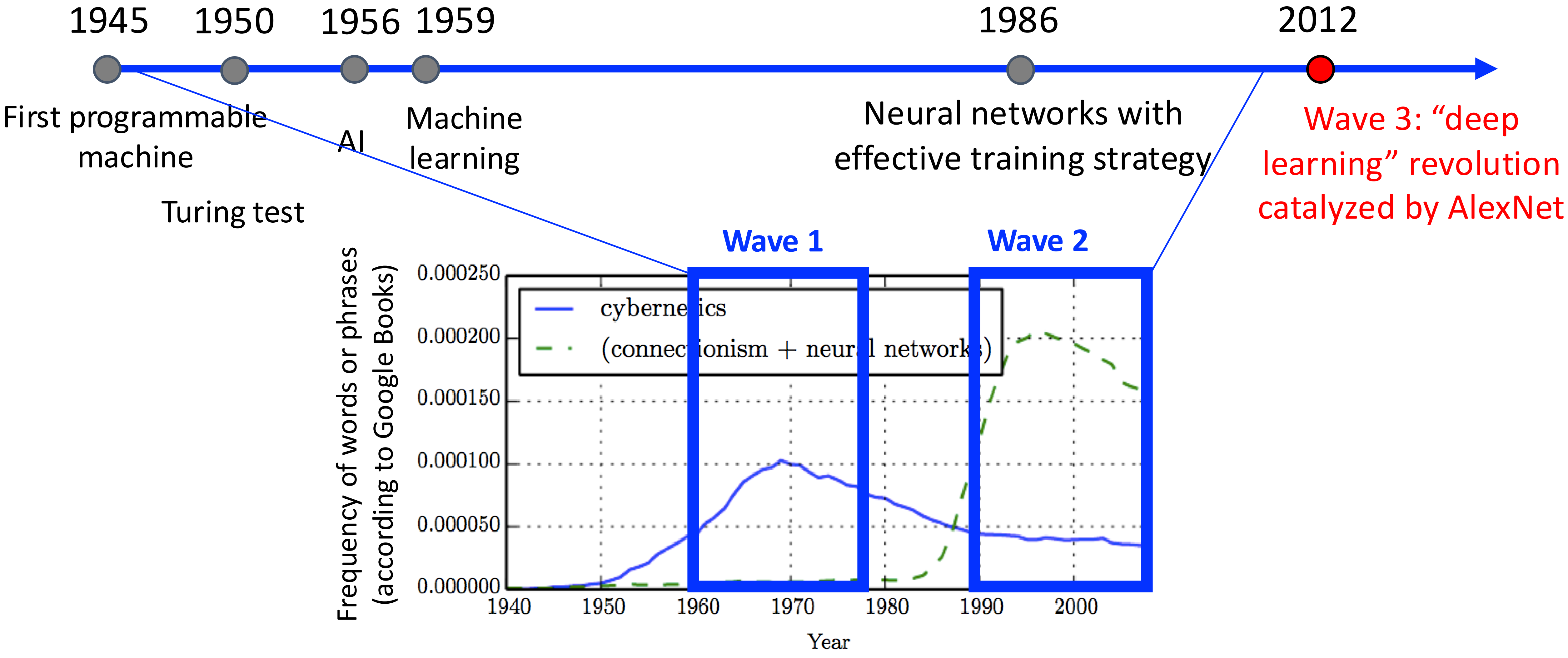
Premise: Large Capacity Model Necessary

So much complexity for even just one object category:



Source: Kristen Grauman

How to Successfully Train Large Capacity Model?



Ian Goodfellow, Yoshua Bengio, and Aaron Courville; Deep Learning, 2016

Today’s Topics

- Key challenge: training large capacity, deep models
- AlexNet: key tricks for going 8 layers deep
- ResNet: key tricks for extending to 152 layers deep
- Programming tutorial

(Model Named After First Author)

(2012, Neurips)

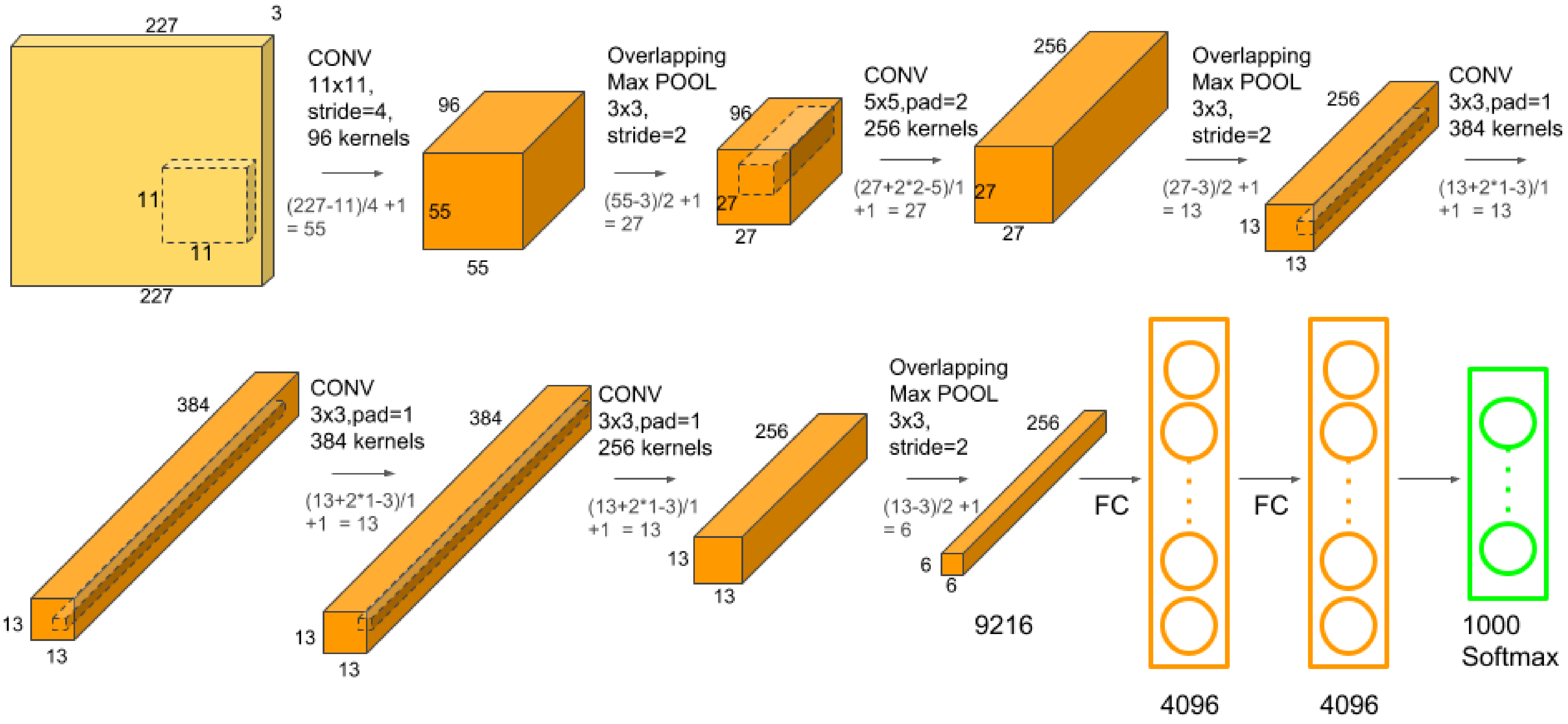
ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

AlexNet Architecture: Similar to LeNet But With More Convolutional and Pooling Layers



https://www.learnopencv.com/wp-content/uploads/2018/05/AlexNet-1.png

AlexNet Architecture

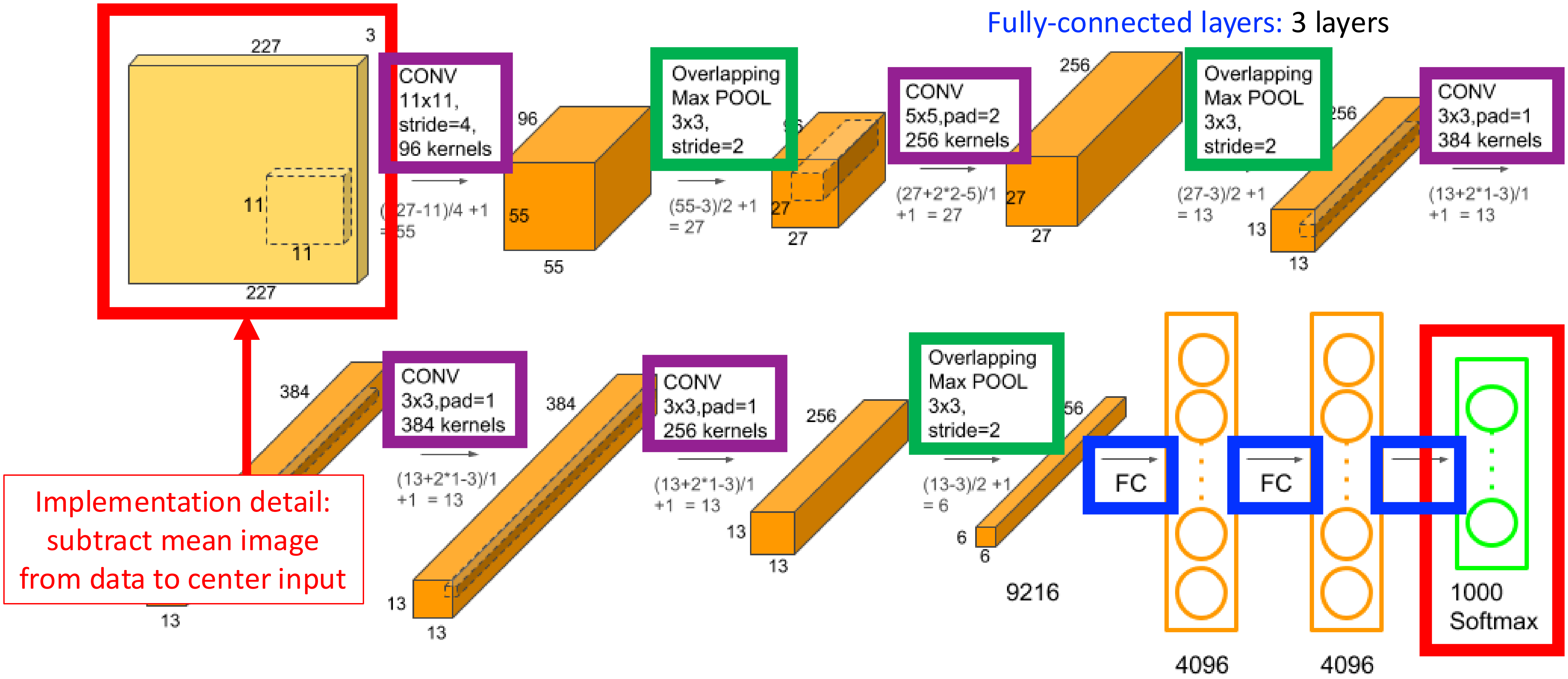
Input: RGB image resized to fixed input size

Output: 1000 class probabilities (sums to 1)

Convolutional layers: 5 layers

Pooling Layers: 3 layers

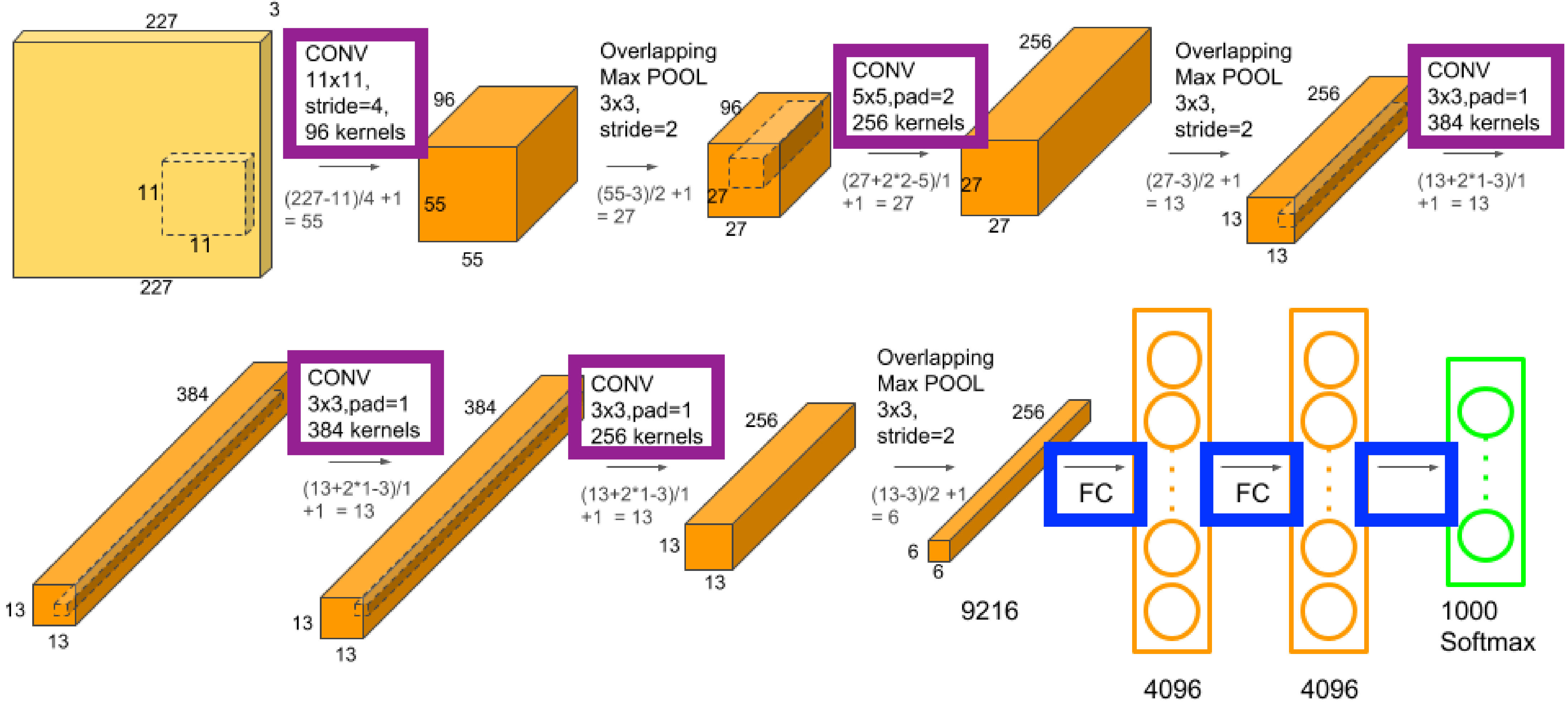
Fully-connected layers: 3 layers



https://www.learnopencv.com/wp-content/uploads/2018/05/AlexNet-1.png

AlexNet Architecture

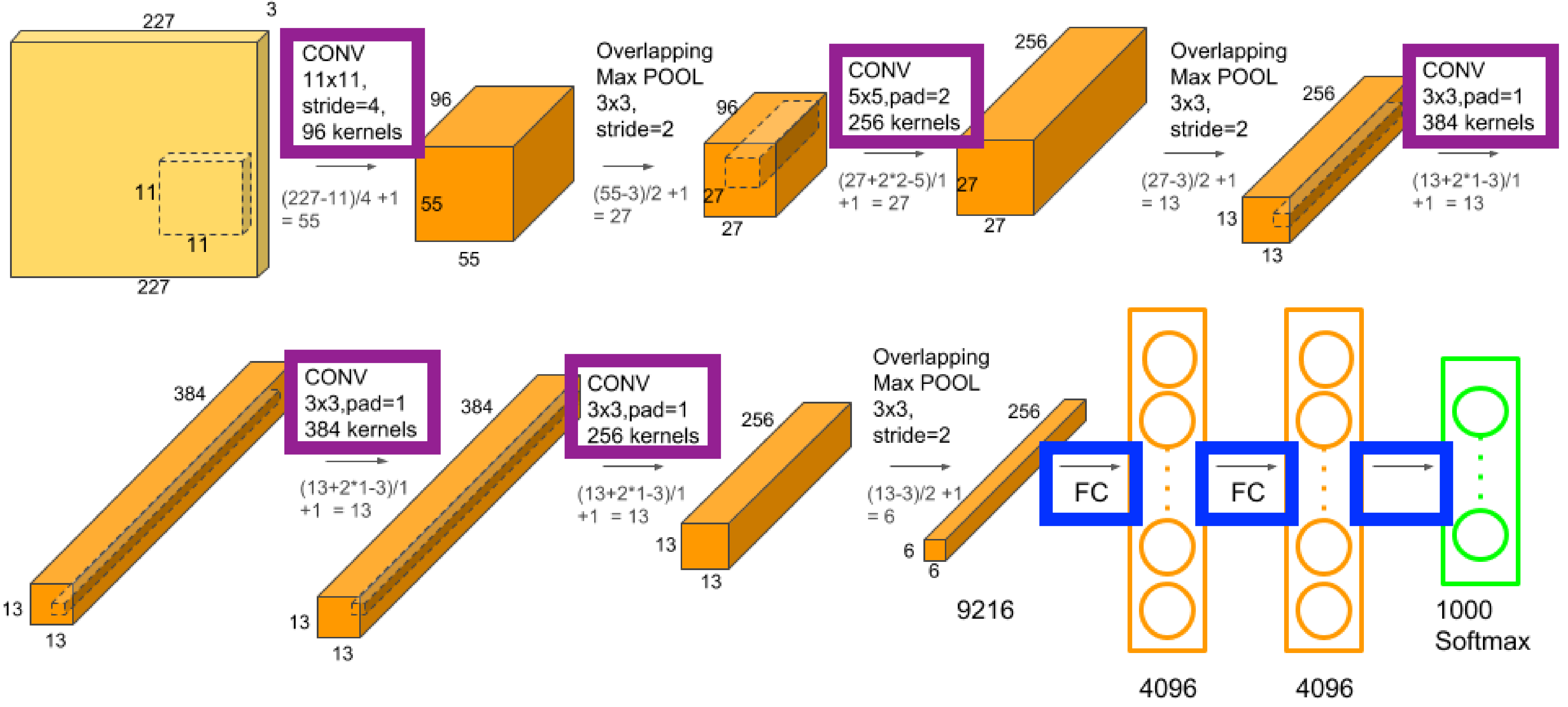
How many layers have model parameters that need to be learned?



https://www.learnopencv.com/wp-content/uploads/2018/05/AlexNet-1.png

AlexNet Architecture

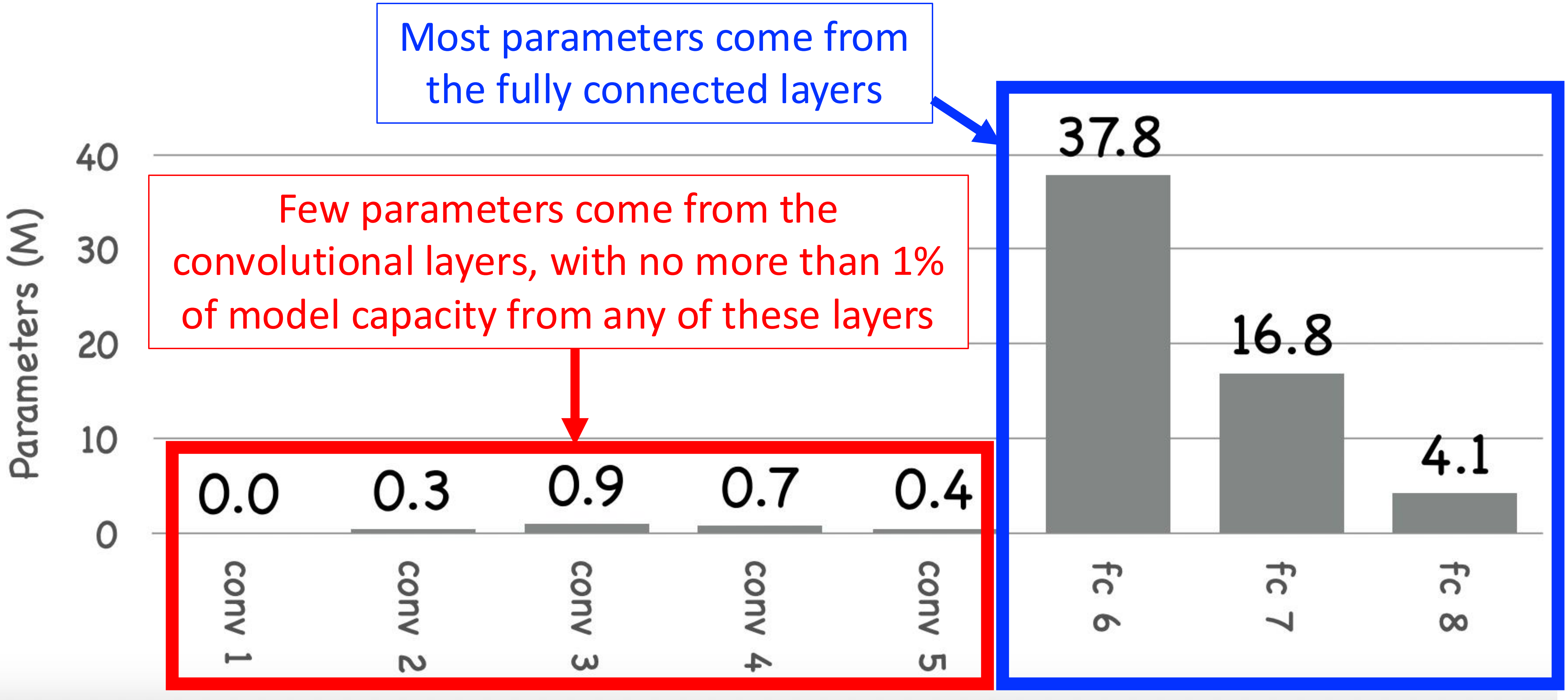
Altogether, 60 million model parameters must be learned!



https://www.learnopencv.com/wp-content/uploads/2018/05/AlexNet-1.png

AlexNet Architecture

Altogether, 60 million model parameters must be learned!



http://www.philkr.net/cs342/lectures/computer_vision/02.pdf

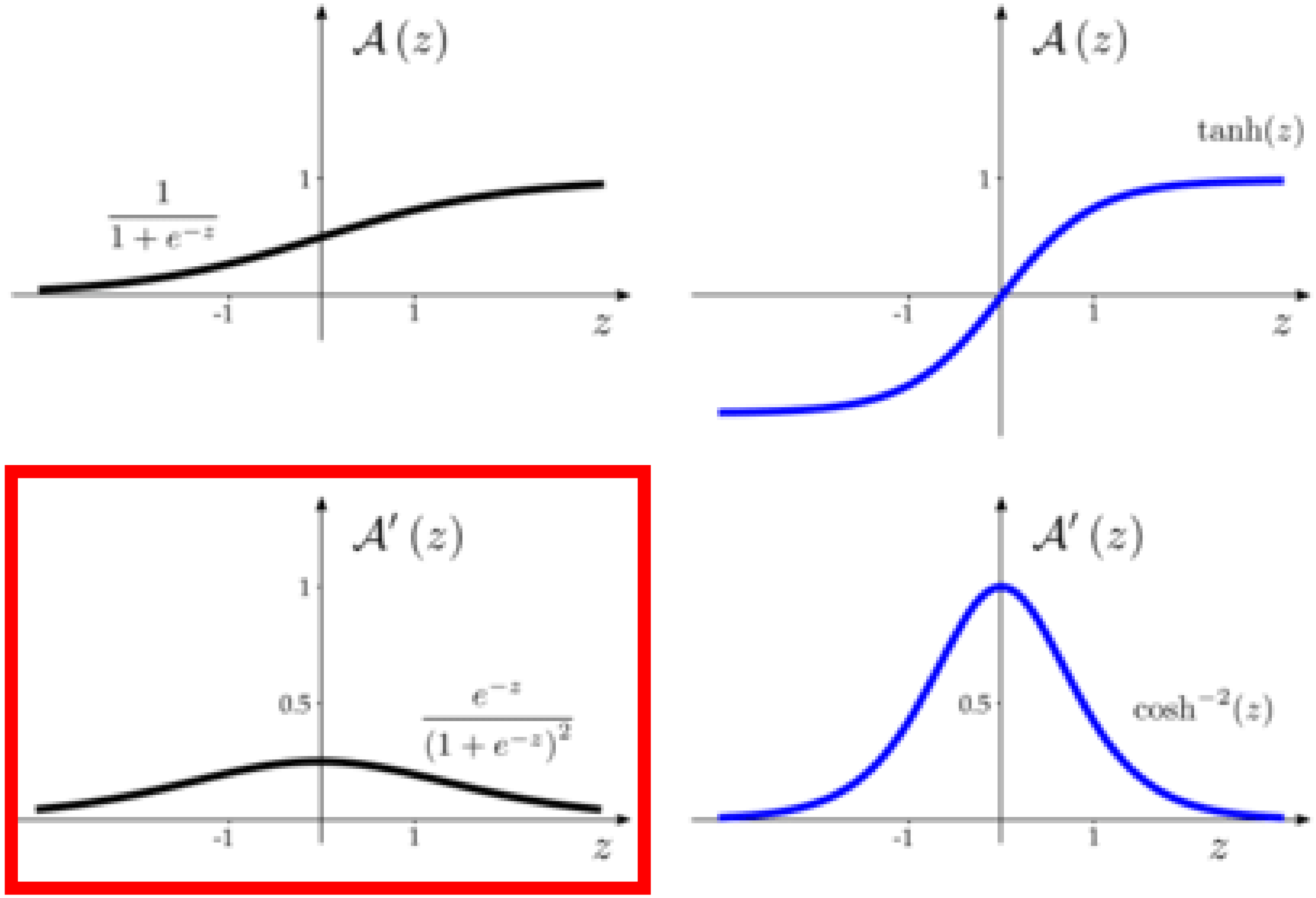
Key Ideas for Training a Large Capacity Model

- Enable learning: use non-saturating activation functions
- Prevent overfitting: incorporate regularization methods
- Make training feasible: speed it up with better hardware

Issue: Mainstream Activation Functions at the Time Were Unsuitable for Training

Sigmoid

Tanh



How are derivatives of these activation functions inadequate?

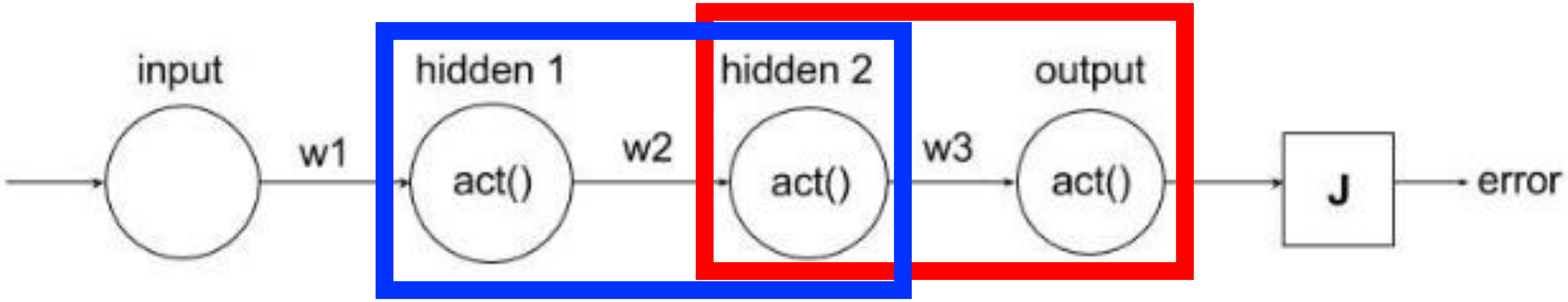
Derivative typically less than 1;

e.g., sigmoid range is 0 to 0.25

Masi et al. Journal of the Mechanics and Physics of Solids. 2021

Vanishing Gradient Problem; e.g., sigmoid

- Toy example:



- Error Derivative with respect to weight w1:

$$\frac{\partial error}{\partial w1} = \frac{\partial error}{\partial output} * \boxed{\frac{\partial output}{\partial hidden2}} * \boxed{\frac{\partial hidden2}{\partial hidden1}} * \frac{\partial hidden1}{\partial w1}$$

Derivative of sigmoid activation function: (0 to 1/4]

Derivative of sigmoid activation function: (0 to 1/4]

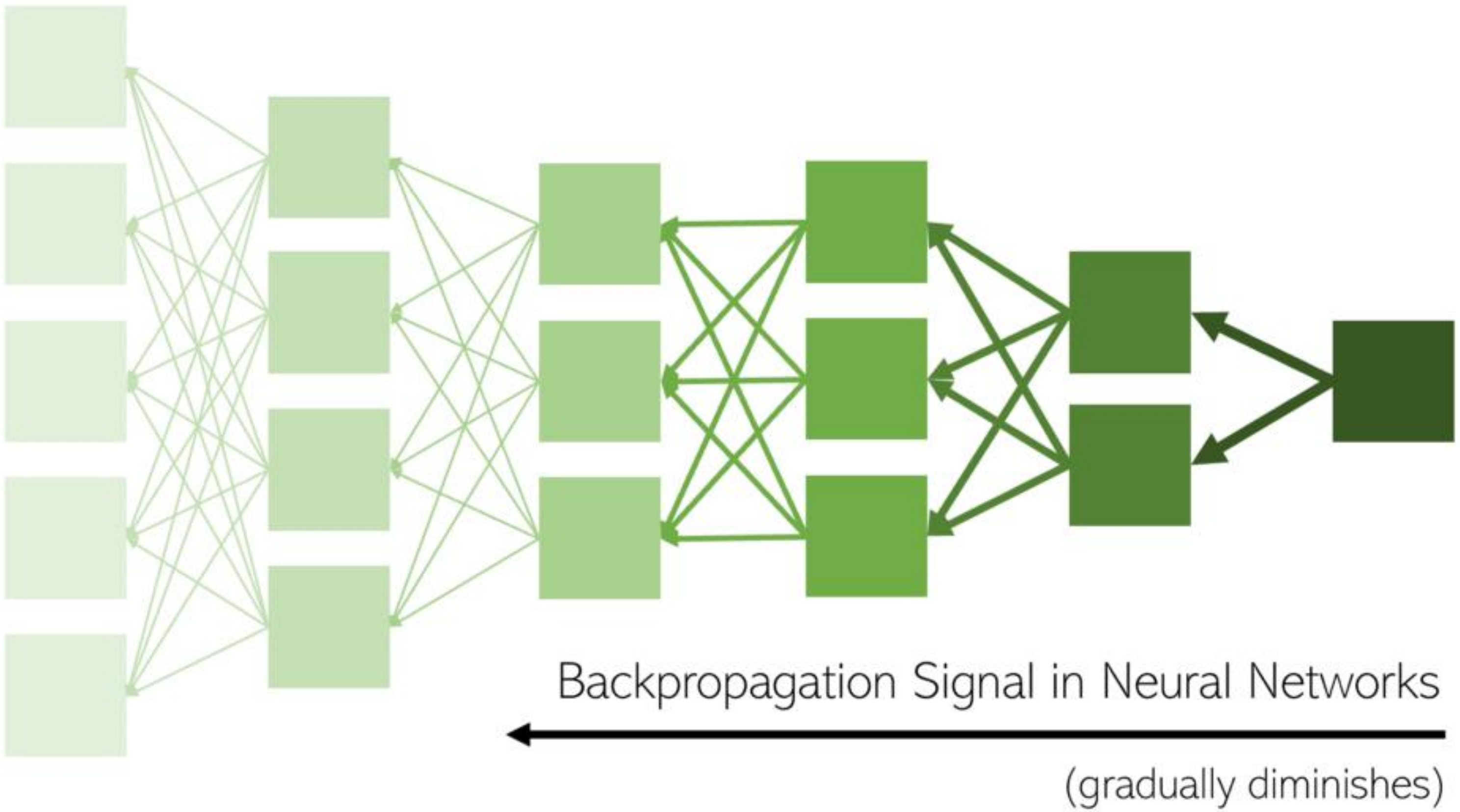
Problem: What happens when multiplying many numbers smaller than 1?

Gradient becomes smaller… and so weights can barely change at training!

https://ayearofai.com/rohan-4-the-vanishing-gradient-problem-ec68f76ffb9b

Vanishing Gradient Problem

Smaller gradients at **earlier layers make them slowest to train**, yet later layers depend on those earlier layers to do something useful; consequently, NNs struggle with garbage in means garbage out

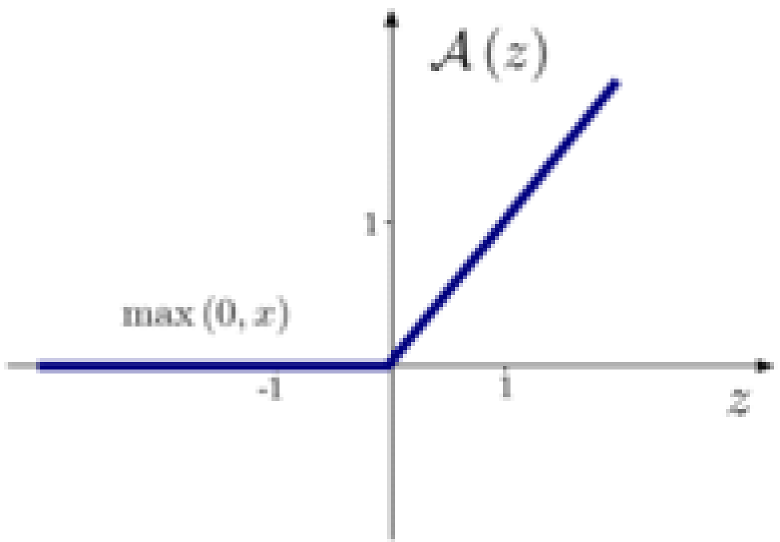


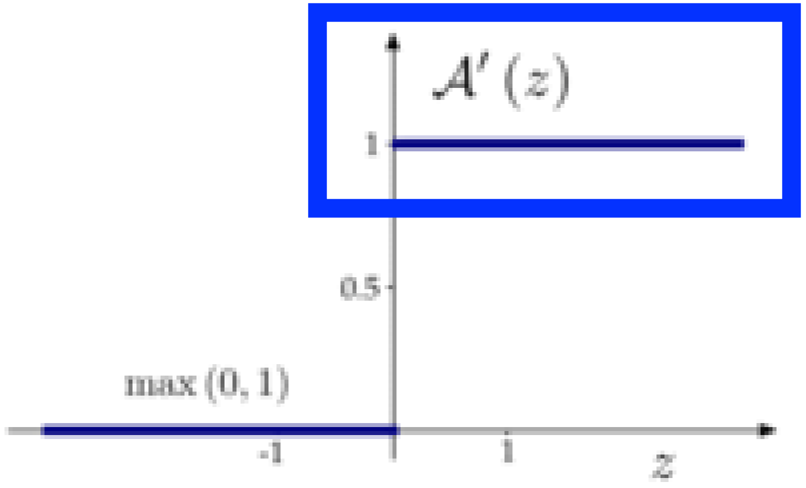
https://towardsdatascience.com/batch-normalization-the-greatest-breakthrough-in-deep-learning-77e64909d81d

Idea: Use Different Activation Function

Use activation function with derivative equal to 1: ReLU

- i.e., 1x1x1… means gradient won't vanish





- Further advantage: fast to compute!

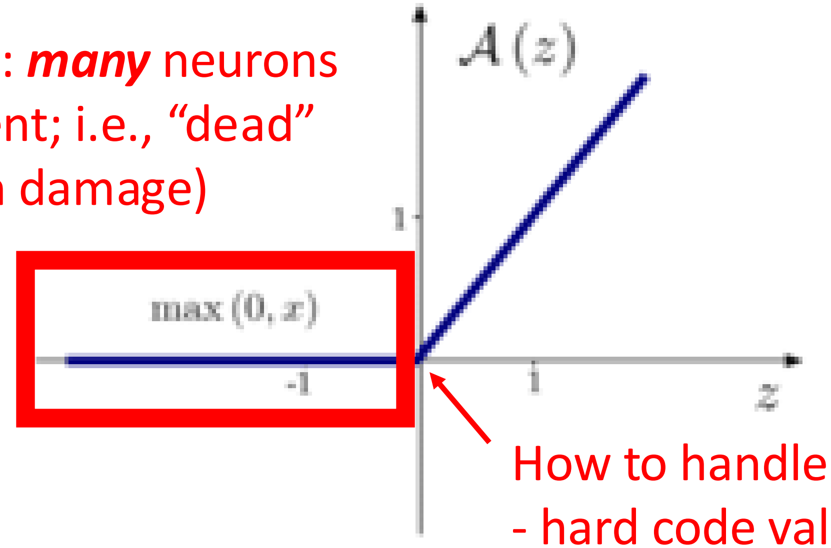
Masi et al. Journal of the Mechanics and Physics of Solids. 2021

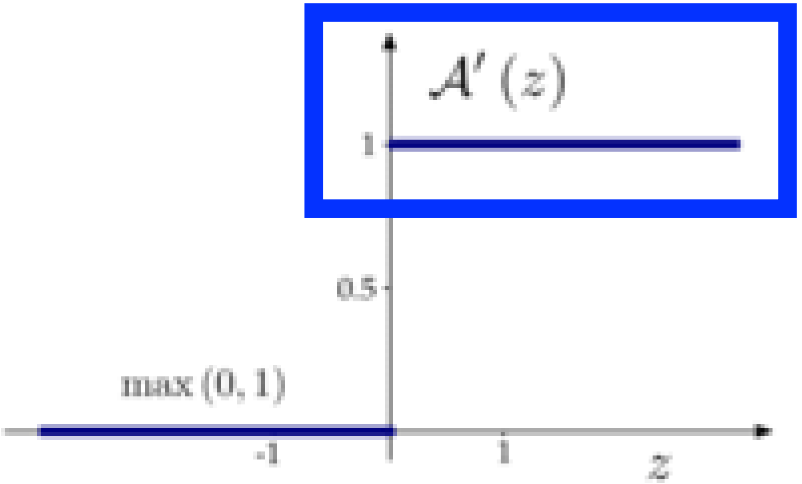
Idea: Use Different Activation Function

Use activation function with derivative equal to 1: ReLU

- i.e., 1x1x1… means gradient won't vanish

Potential issue: ***many*** neurons have no gradient; i.e., "dead" (analogy: brain damage)





How to handle when function is not differentiable (at z=0)?
- hard code value (i.e., 0 or 1)

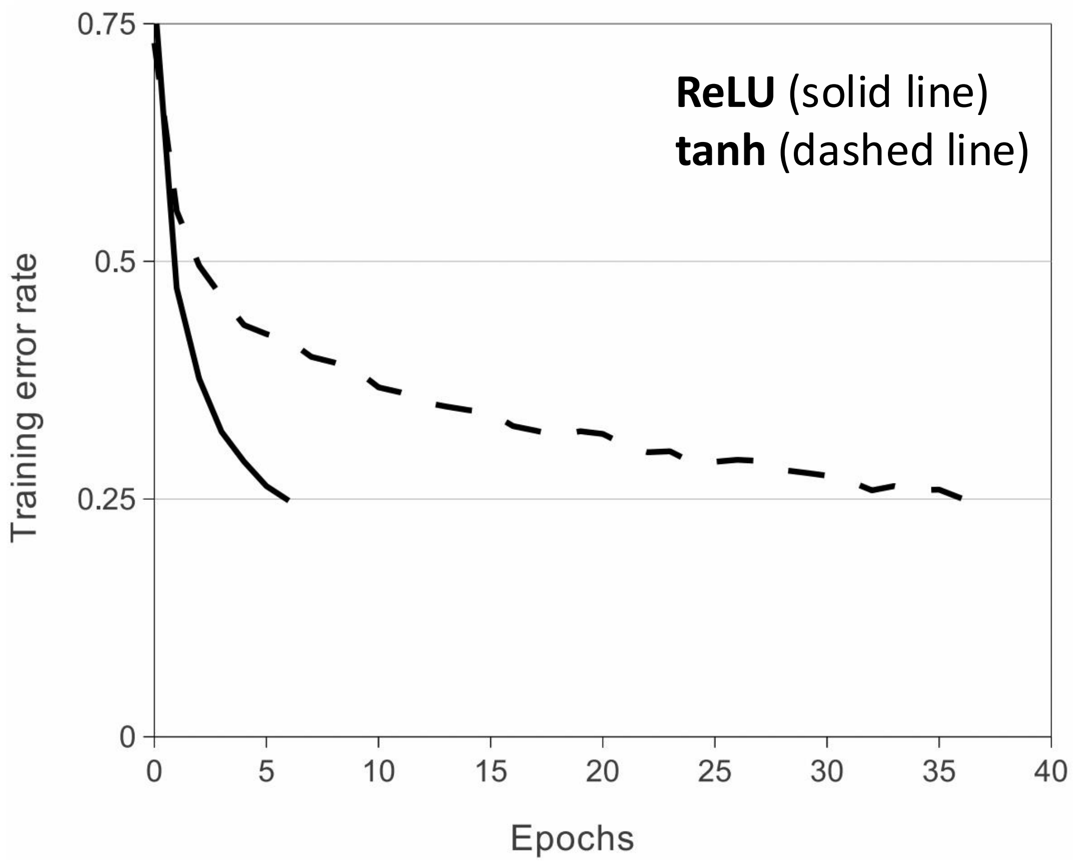
- When using backpropagation with ReLU, what are the possible values?

Masi et al. Journal of the Mechanics and Physics of Solids. 2021

Motivating Experimental Analysis

- Dataset: CIFAR-10
- Model Architecture: 4-layer convolutional network
- Evaluation metric: % correct

What is the key finding?



ReLU yields much faster learning than tanh, with the latter unsuitable for learning!

(e.g., ReLU is 6x faster in achieving 25% error rate)

Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks. NeurIPS 2012.

Key Ideas for Training a Large Capacity Model

- Enable learning: use non-saturating activation functions
- Prevent overfitting: incorporate regularization methods
- Make training feasible: speed it up with better hardware

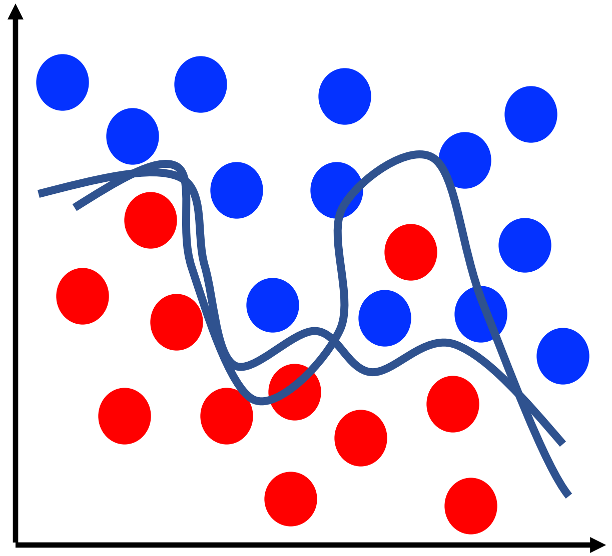
Regularization Methods

- Recall: regularization is “any modification we make to a learning algorithm that is intended to reduce its generalization error.”- Goodfellow book

- Two approaches leveraged by AlexNet: data augmentation & dropout

Data Augmentation: Intuition

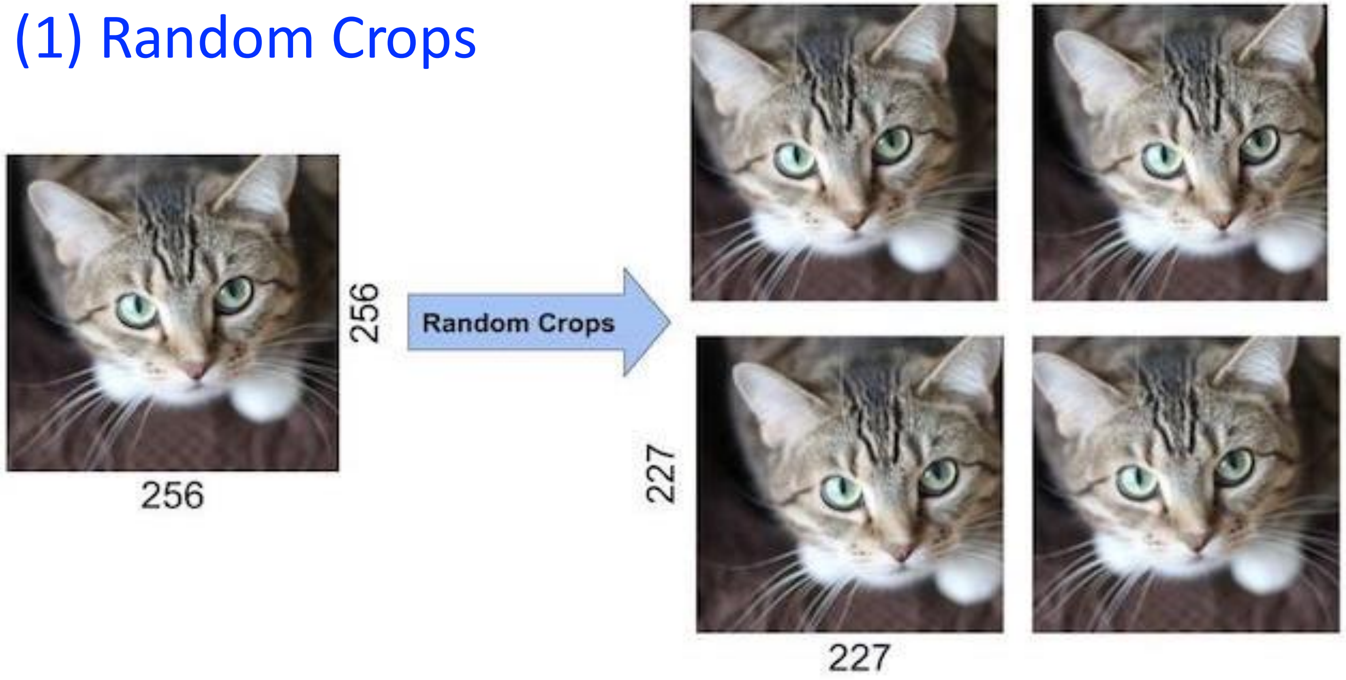
Adding training data



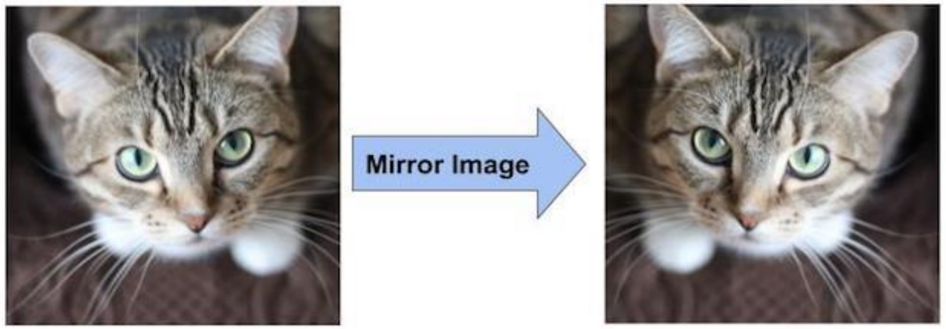
Data Augmentation: Approach

- Random patches and their mirror images (2048x more data)

(1) Random Crops



(2) Mirror Image



Generally, need to ensure augmentation scheme aligns with target application:

9 → 6

d → b

- Adjust RGB channels (using PCA-based method)

https://learnopencv.com/understanding-alexnet/

Dropout

(2012, arXiv)

Improving neural networks by preventing co-adaptation of feature detectors

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

Dropout: Idea

Use ensemble

Dropout: Idea

- Using ensemble reduces probability for making a wrong prediction
- Suppose:
  - n classifiers for binary classification task
  - Each classifier has same error rate ε
  - Classifiers are independent (not true in practice!)
  - Probability mass function indicates the probability of error from an ensemble:

Number of classifiers

Classifier error rate

$$P(y \geq k) = \sum_{k}^{\boxed{n}} \boxed{\binom{n}{k}} \varepsilon^k (1 - \boxed{\varepsilon})^{n-k} = \boxed{\varepsilon_{ensemble}}$$

Error probability

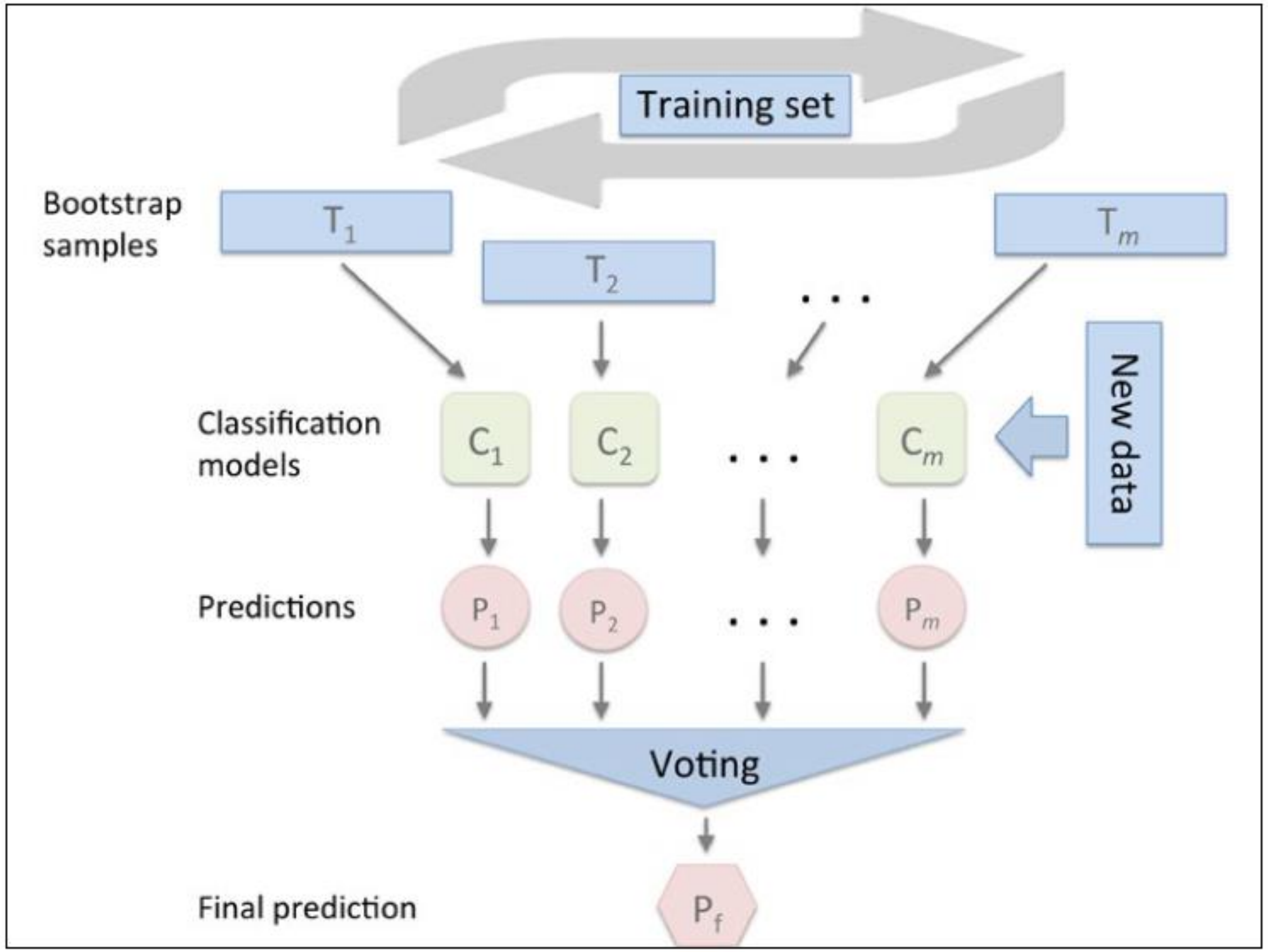
ways to choose *k* subsets from set of size *n*

  - e.g., n = 11, ε = 0.25; k = 6: probability of error is ~0.034 which is much lower than probability of error from a single algorithm (0.25)

Dropout: Precursor

Bootstrap Aggregation (1994)

Train algorithm repeatedly on different random subsets of the training set



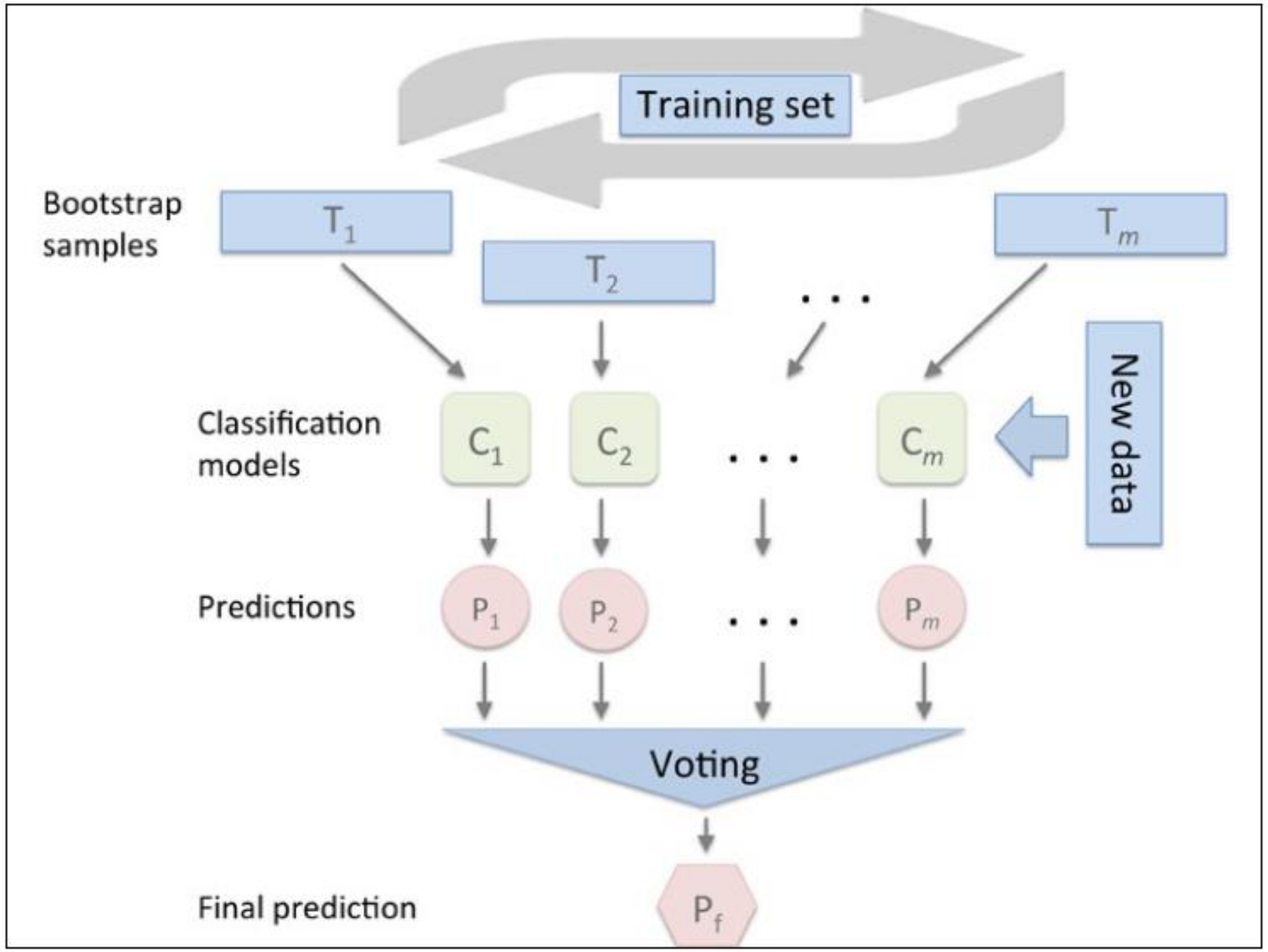
Raschka & Mirjalili, Python Machine Learning.

Dropout: Precursor

Train algorithm repeatedly on different random subsets of the training set

Why is bagging a poor approach for neural networks?

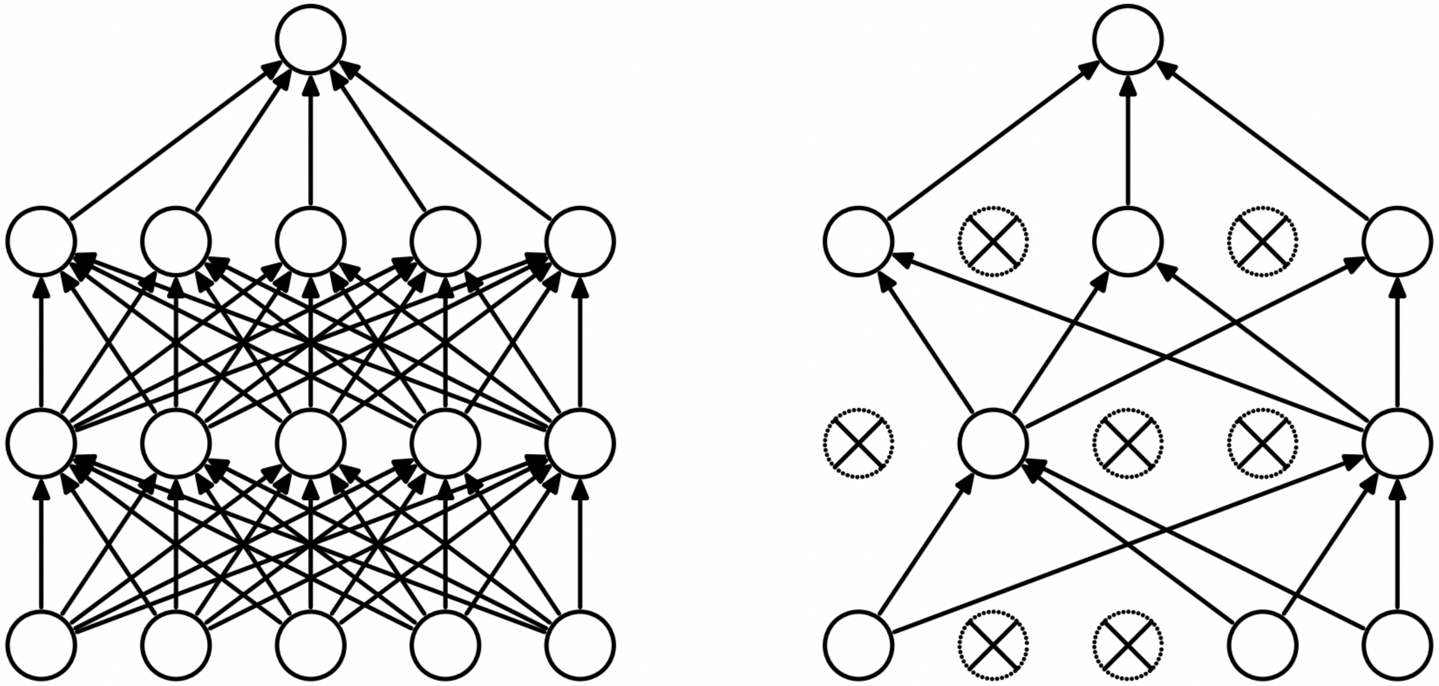
- Finding optimal hyperparameters for each architecture is time-consuming
- Applying multiple neural networks is often infeasible since the models require lots of memory and are computationally expensive to run



Raschka & Mirjalili, Python Machine Learning.

Dropout: Approach

- Approximates bagging with dropout during training so different sub-models in the network are trained with different training data



(a) Standard Neural Net (b) After applying dropout.

e.g., drop 50% of units in hidden layers

e.g., drop 20% of units in input layers

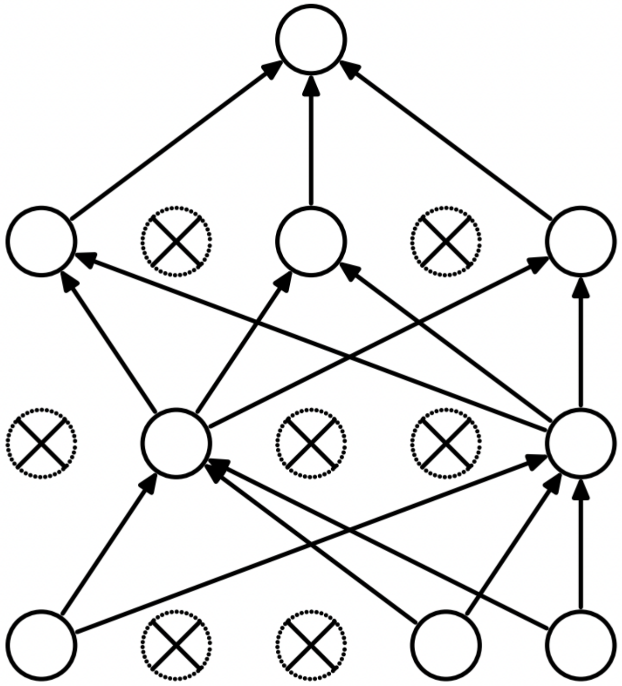
Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014

Dropout: Approach

- Approximates bagging with dropout during training so different sub-models in the network are trained with different training data

For training, the forward pass and backpropagation run only through the sub-network (with a different dropout per minibatch).

What might happen to loss curves?
- Bouncier since the underlying network continuously changes



(b) After applying dropout.

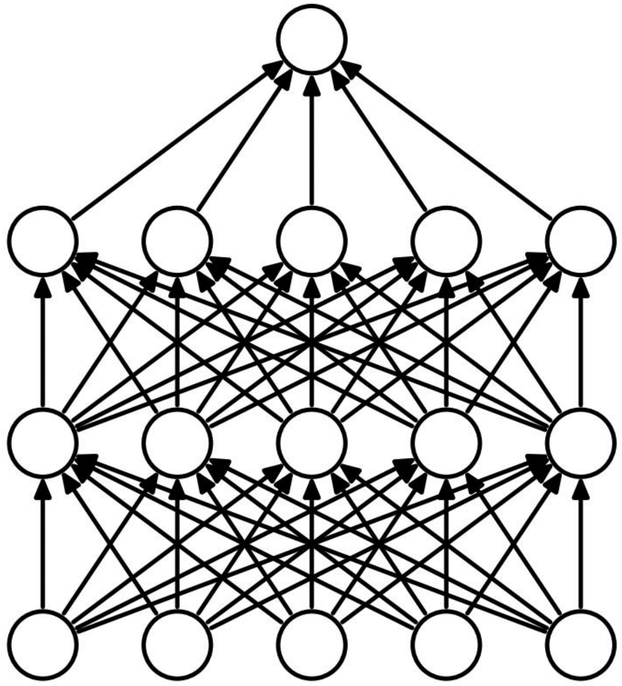
Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014

Dropout: Approach

- Approximates bagging with dropout during training so different sub-models in the network are trained with different training data

Ensemble is emulated at test time by applying the network without dropout

How to handle network's expectation for a smaller activation signal than observed at test time (e.g., input from 2 versus 5 neurons)?
- Multiply each unit's outgoing weights by probability of dropping at training

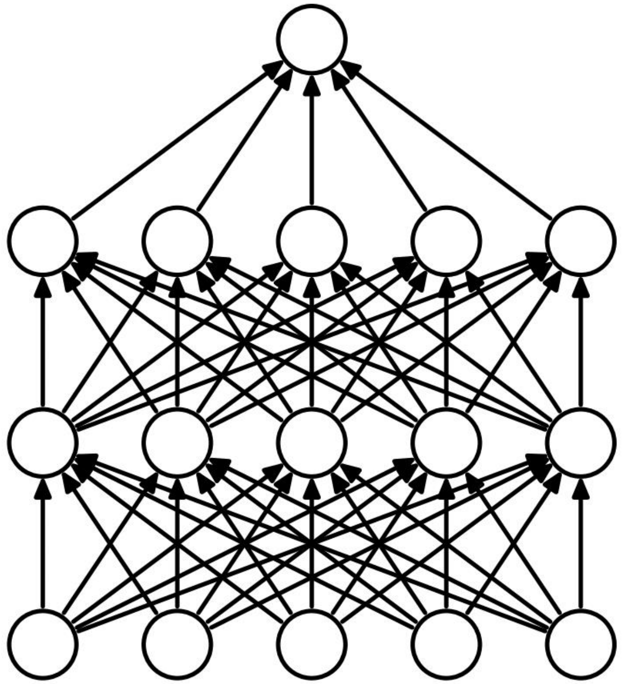


(b) After applying dropout.

Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014

Dropout: Dropout vs Bagging

- Dropout approximates bagging with many models inexpensively
  - Trains algorithm repeatedly on random subsets of the training set
- Dropout differences are that subnetworks are not:
  - Trained to convergence (instead, trained for one step)
  - Independent (instead, they all share parameters)

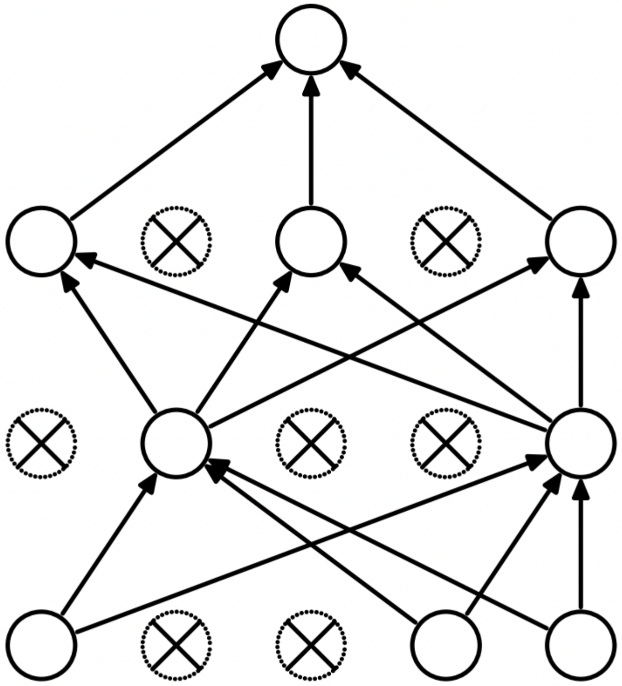


(b) After applying dropout.

Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014

Dropout: Motivation

This approach was motivated by the role of sex in evolution: "… the role of sexual reproduction is not just to allow useful new genes to spread throughout the population, but also to facilitate this process by reducing complex co-adaptations that would reduce the chance of a new gene improving the fitness of an individual."



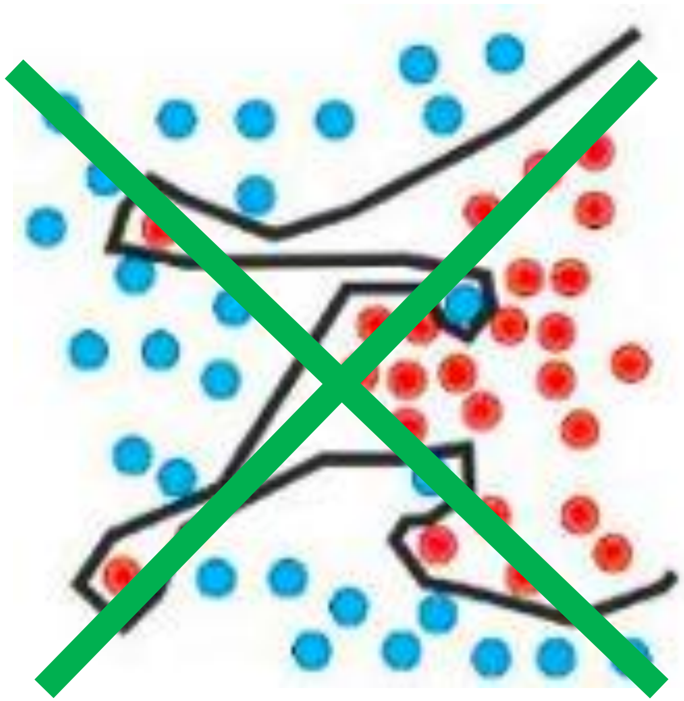
(b) After applying dropout.

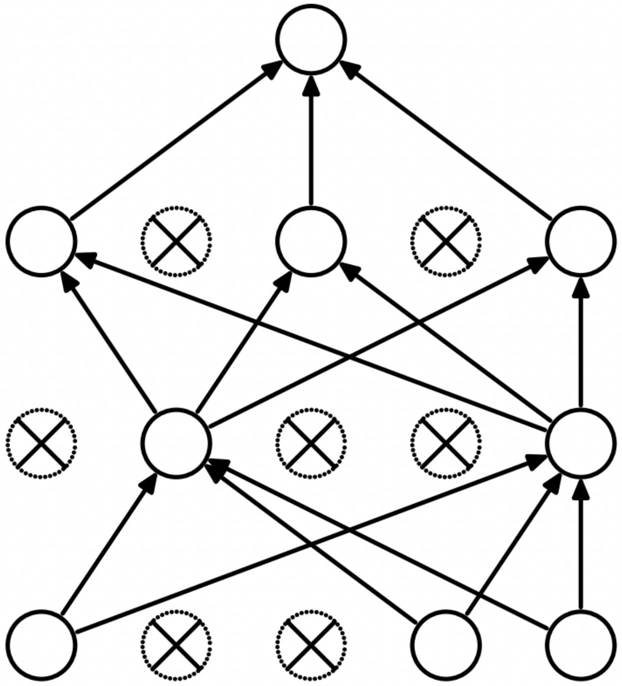
"Similarly, each hidden unit in a neural network trained with dropout must learn to work with a randomly chosen sample of other units. This should make each hidden unit more robust and drive it towards creating useful features on its own without relying on other hidden units to correct its mistakes."

Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014

Dropout: Motivation

Units in the network learn to be useful with many different subsets of other units rather than in conjunction with other units; e.g., mitigates very large positive weights canceling similarly large negative weights (a sign of overfitting)





(b) After applying dropout.

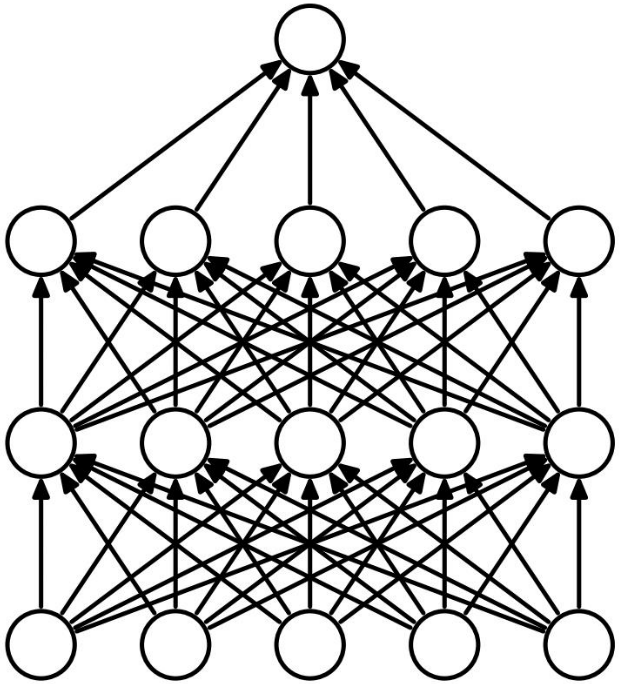
https://towardsdatascience.com/techniques-for-handling-underfitting-and-overfitting-in-machine-learning-348daa2380b9

Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014

Dropout: Alternative Approach

A generalization of zeroing units is to instead multiply units by noise

Relevant articles:
*https://towardsdatascience.com/dropout-on-convolutional-layers-is-weird-5c6ab14f19b2
*Wu and Gu. “Towards dropout training for convolutional neural networks.” Neural Networks, 2015.

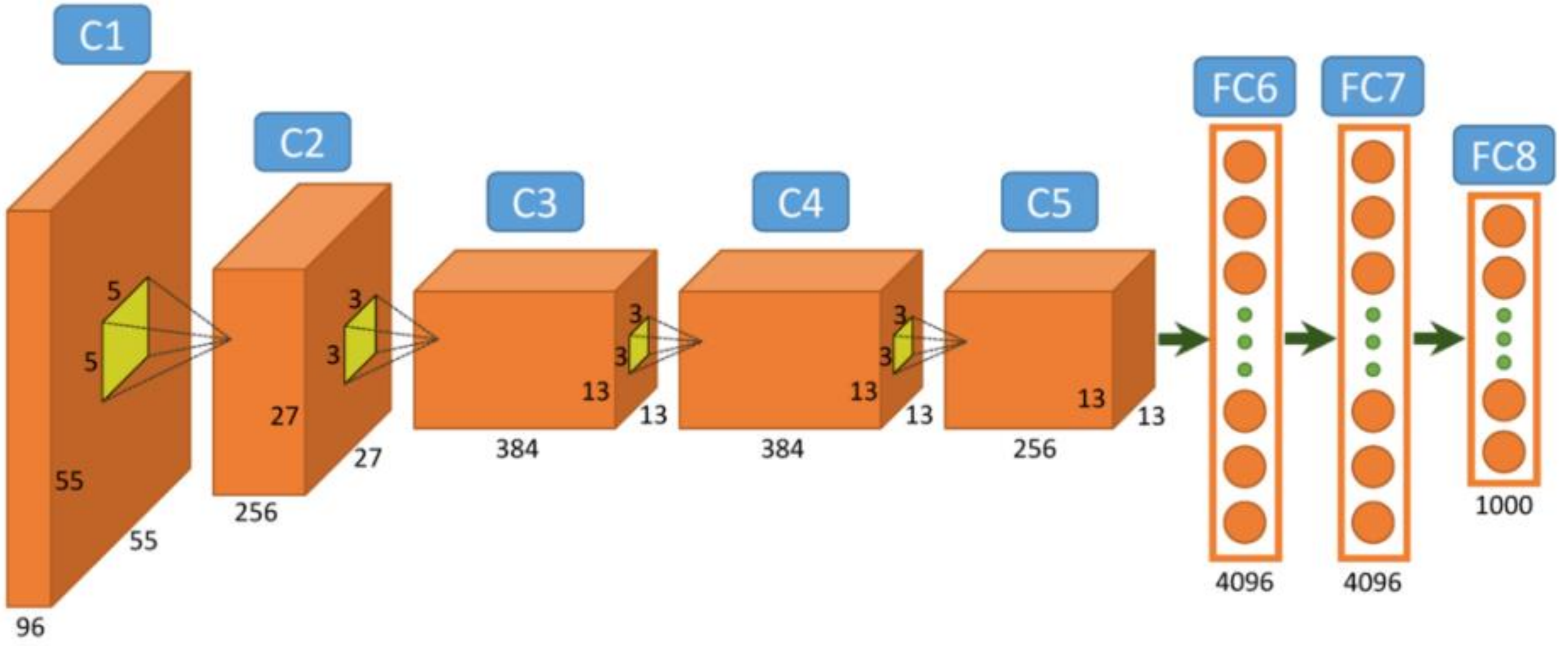


(b) After applying dropout.

Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014

Dropout: Implementation

- Only used in fully connected layers
- Why not use it in convolutional layers?
  - Parameter tying already reduces parameter count and so offers enough regularization

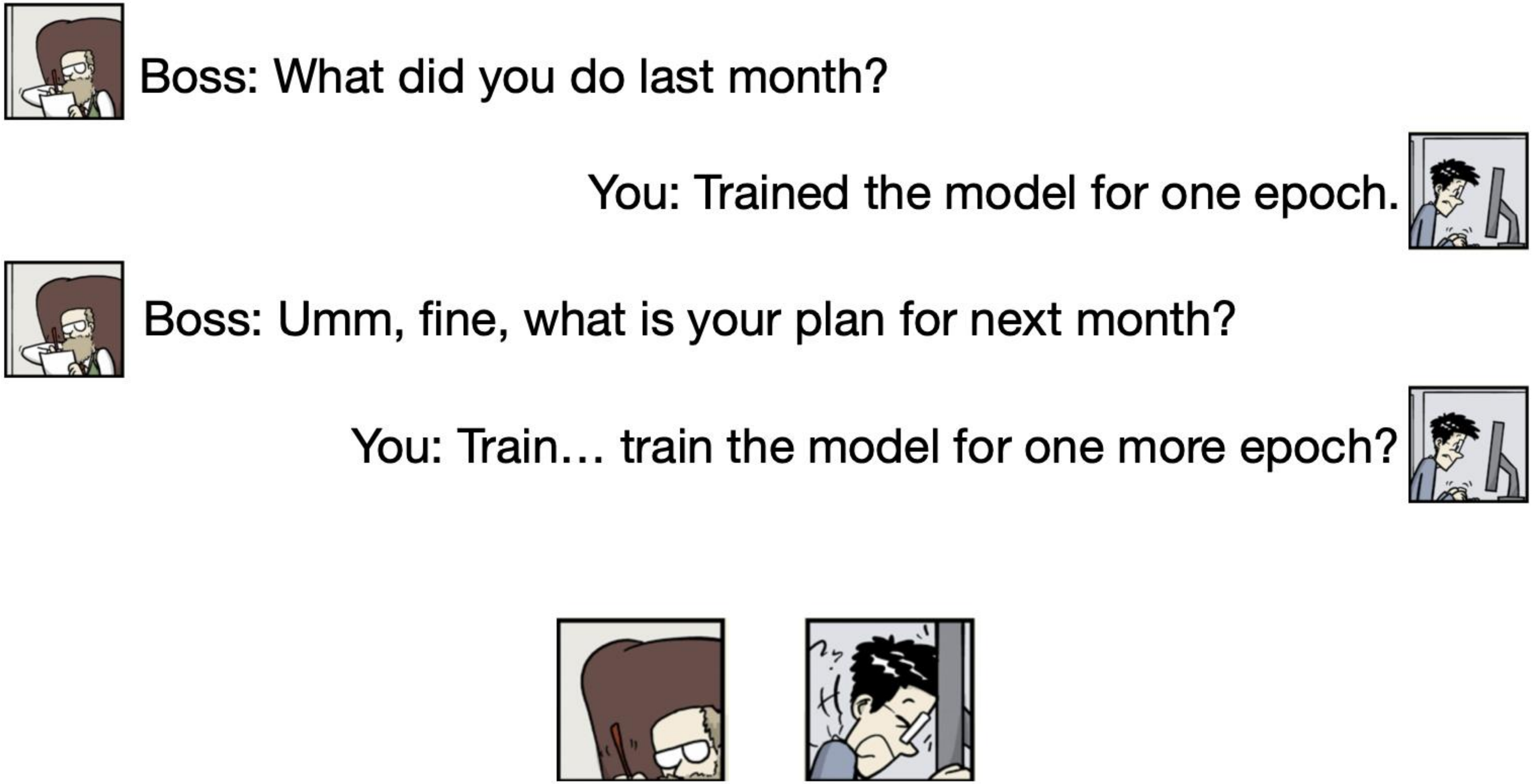


https://www.learnopencv.com/wp-content/uploads/2018/05/AlexNet-1.png
https://www.researchgate.net/figure/Architecture-of-Alexnet-From-left-to-right-input-to-output-five-convolutional-layers_fig2_312303454

Key Ideas for Training a Large Capacity Model

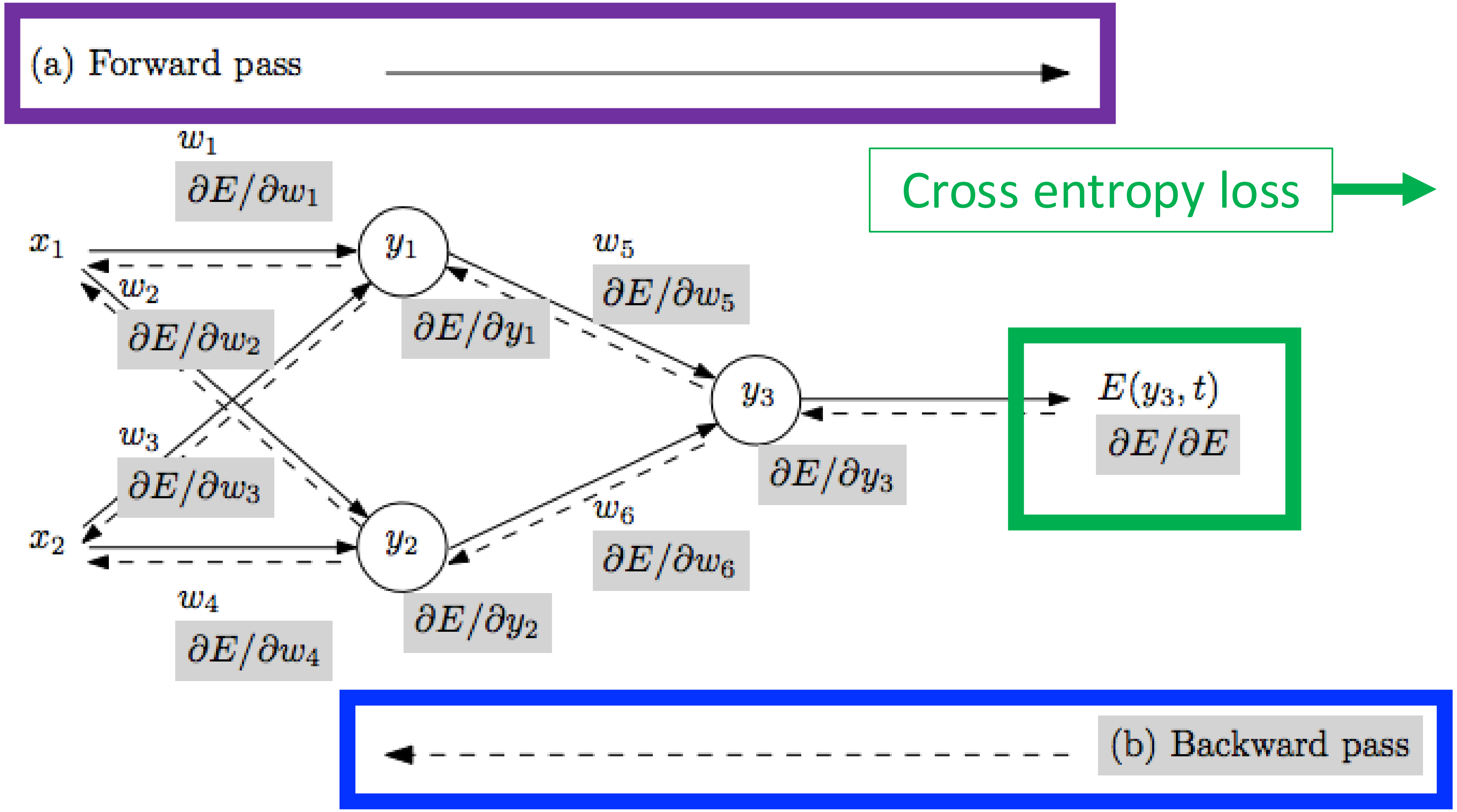
- Enable learning: use non-saturating activation functions
- Prevent overfitting: incorporate regularization methods
- Make training feasible: speed it up with better hardware

Onset of Era for Very Time-Consuming Training



https://hanlab.mit.edu/files/course/slides/MIT-TinyML-Lec13-Distributed-Training-I.pdf

Training: 90 Epochs took 5-6 Days on *2 GPUs*



Repeat until stopping criterion met:

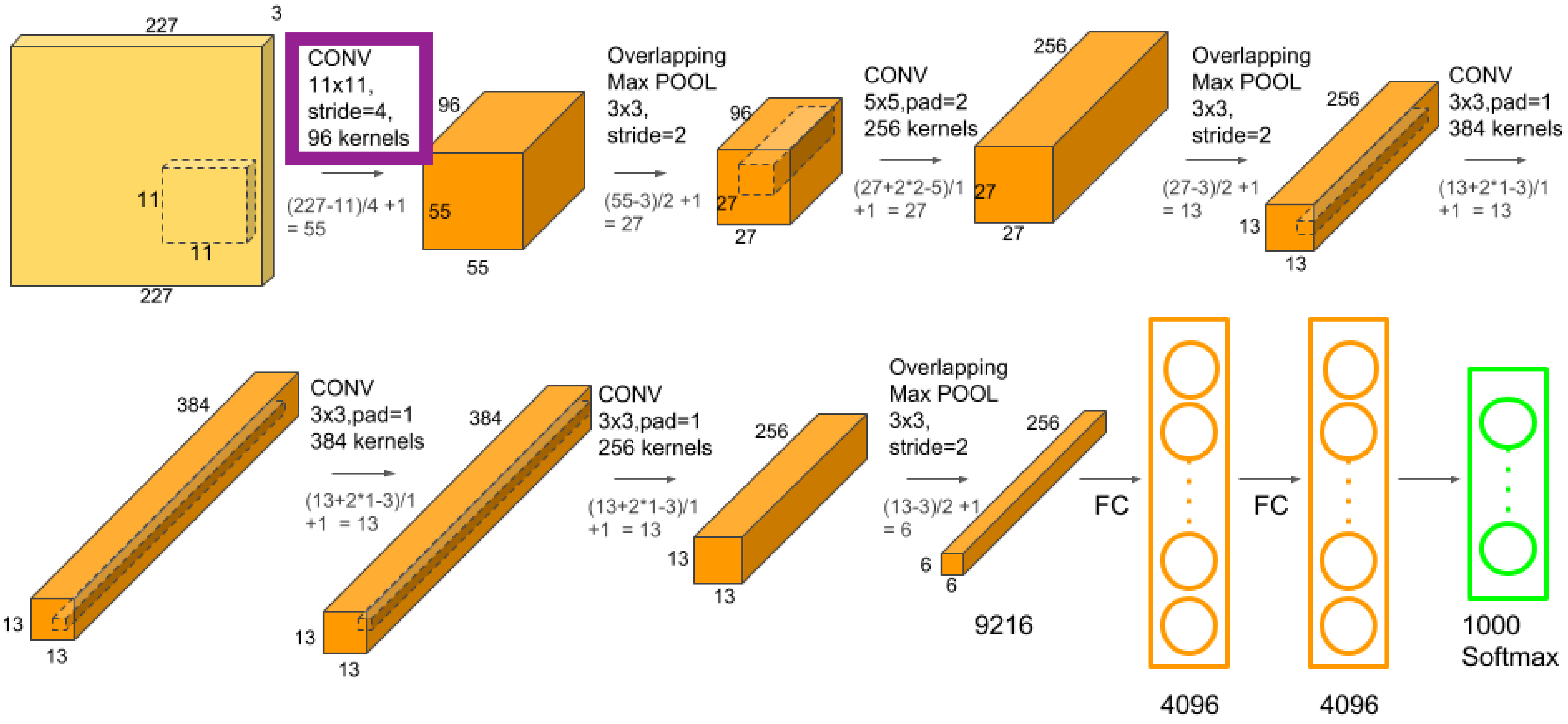
1. **Forward pass**: propagate training data through model to make predictions
2. **Error quantification**: measure error of the model's predictions on training data using a loss function
3. **Backward pass**: calculate gradients to determine how each model parameter contributed to model error
4. Account for weight sharing by using average of all connections for a parameter
5. Update each parameter using calculated gradients

Baydin et al. Automatic Differentiation in Machine Learning: a Survey. 2018

Training Settings

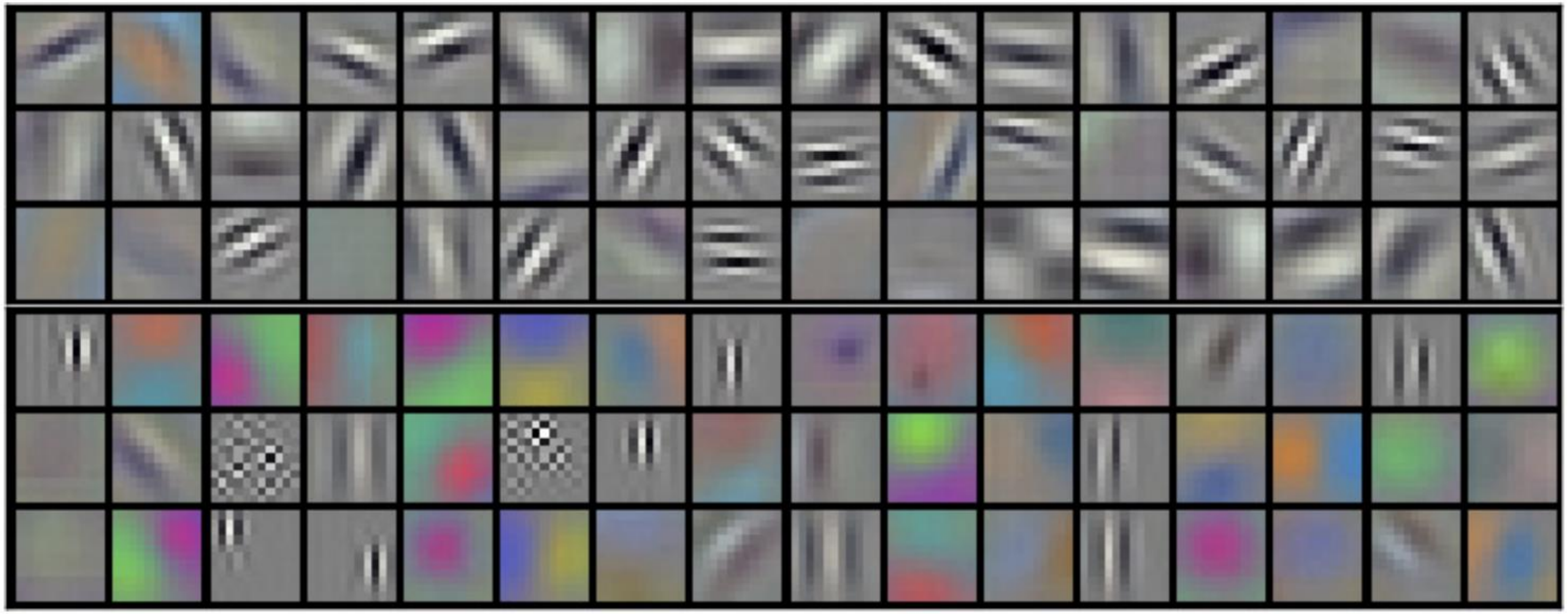
- Batch size: 128 examples
- Initialization: weight values drawn from zero-mean Gaussian distribution with standard deviation 0.01 and biases set to 0 and 1
- Momentum optimization: manually adjusted learning rate 3 times from initial value of 0.01, dividing it by 10 each time validation error stopped falling

AlexNet: Inspecting What It Learned



https://www.learnopencv.com/wp-content/uploads/2018/05/AlexNet-1.png

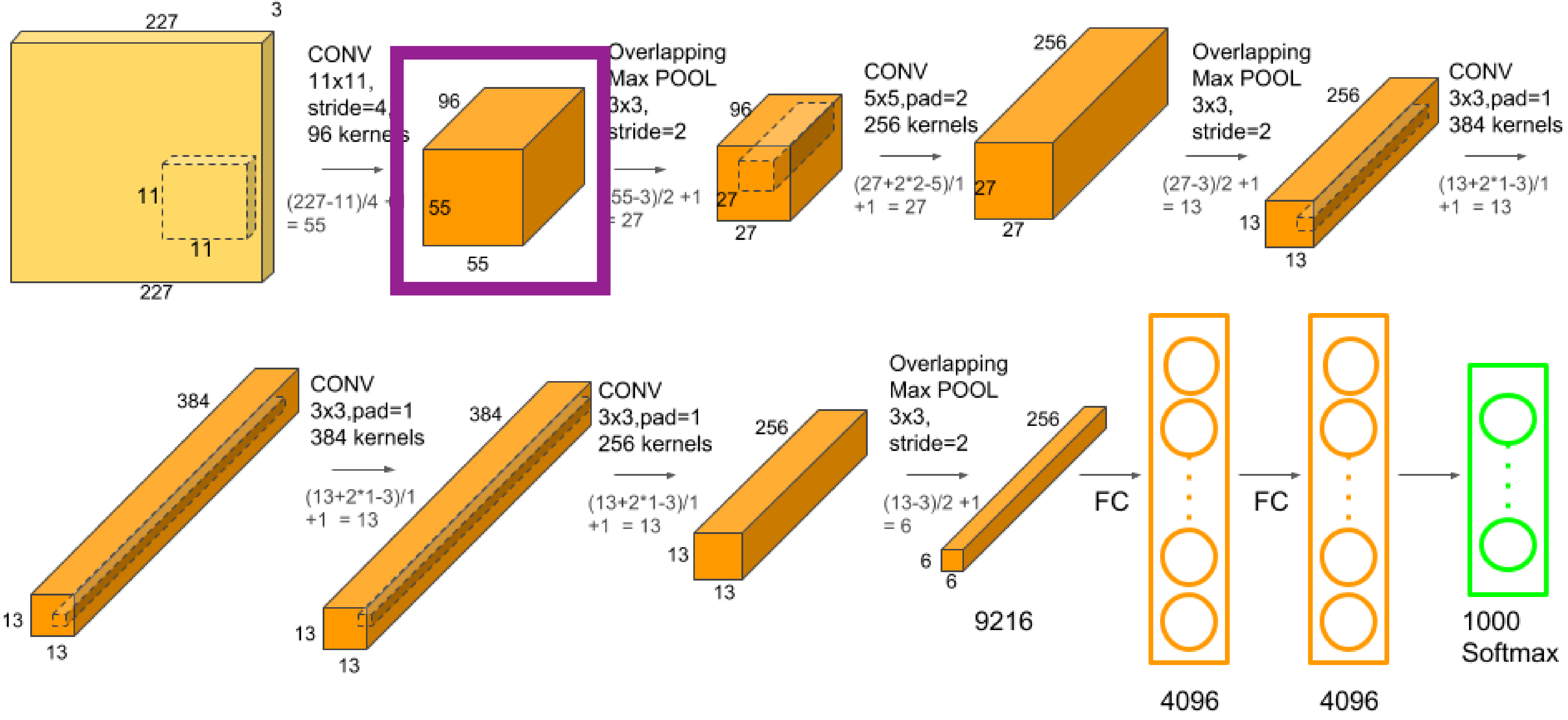
AlexNet: Inspecting What It Learned (96 Filters)



Learned model filters select based on frequency, orientation, and color!
(aligns with Hubel & Weisel's findings for how vision systems work)

Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks. NeurIPS 2012.

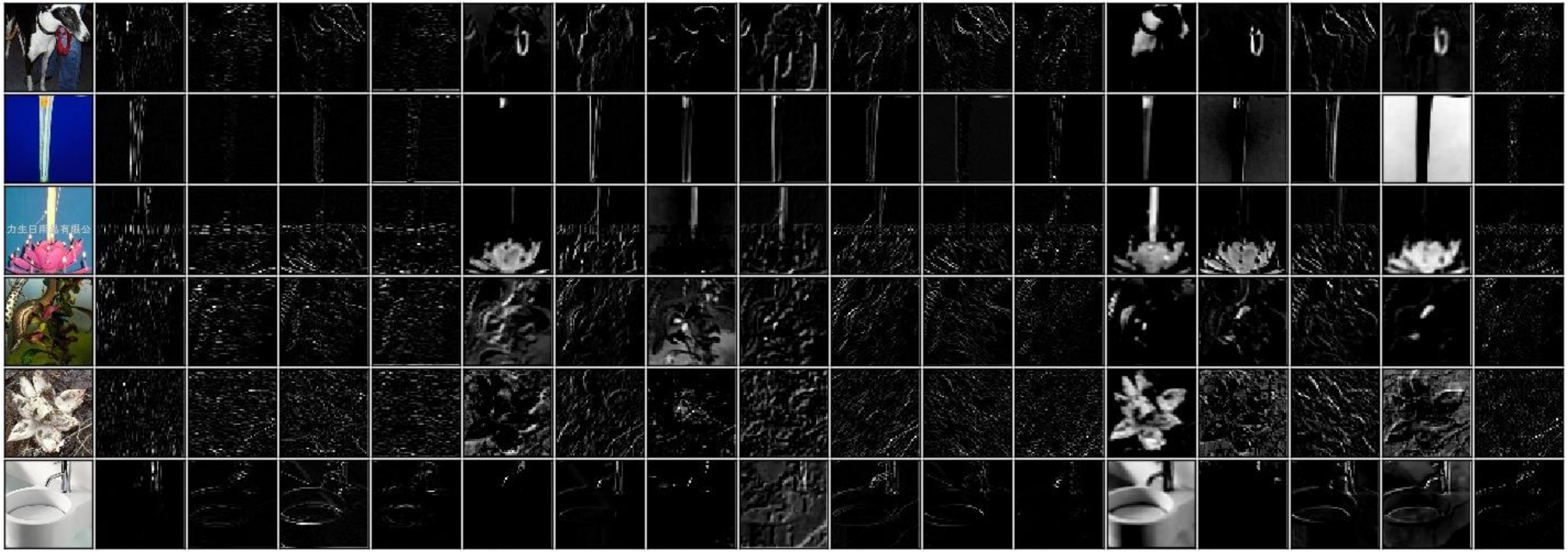
AlexNet: Example Activation Maps



https://www.learnopencv.com/wp-content/uploads/2018/05/AlexNet-1.png

AlexNet: Example Activation Maps (Recall Each Map Results from One Filter)

Images



Frequencies, orientations, and colors are detected

Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks. NeurIPS 2012.

AlexNet Analysis

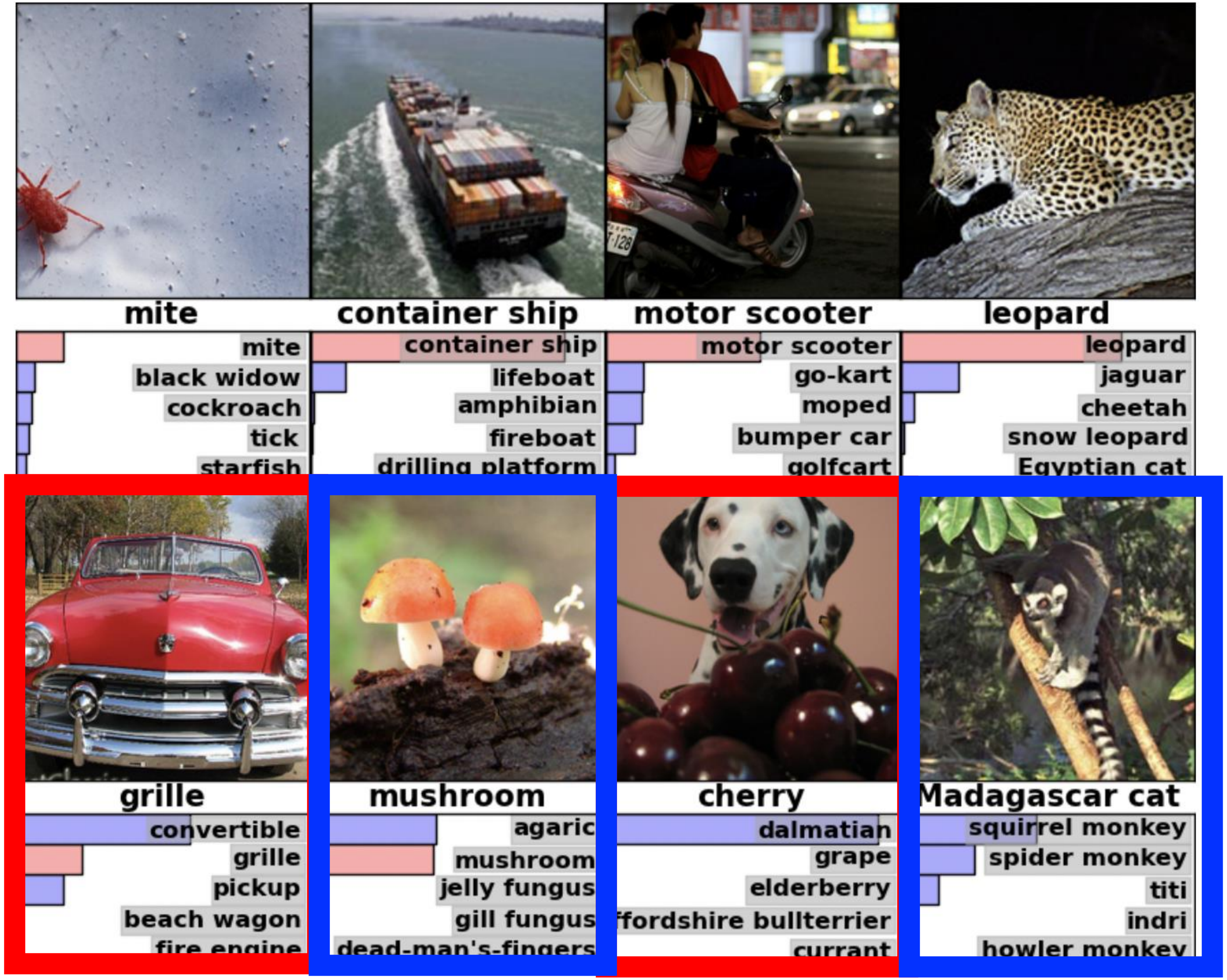
8 examples of predictions, correct and incorrect

When/why might it **succeed**?

- Single well-defined object (even if off-centered)

When/why might it **fail**?

- Ambiguity
- Similar categories

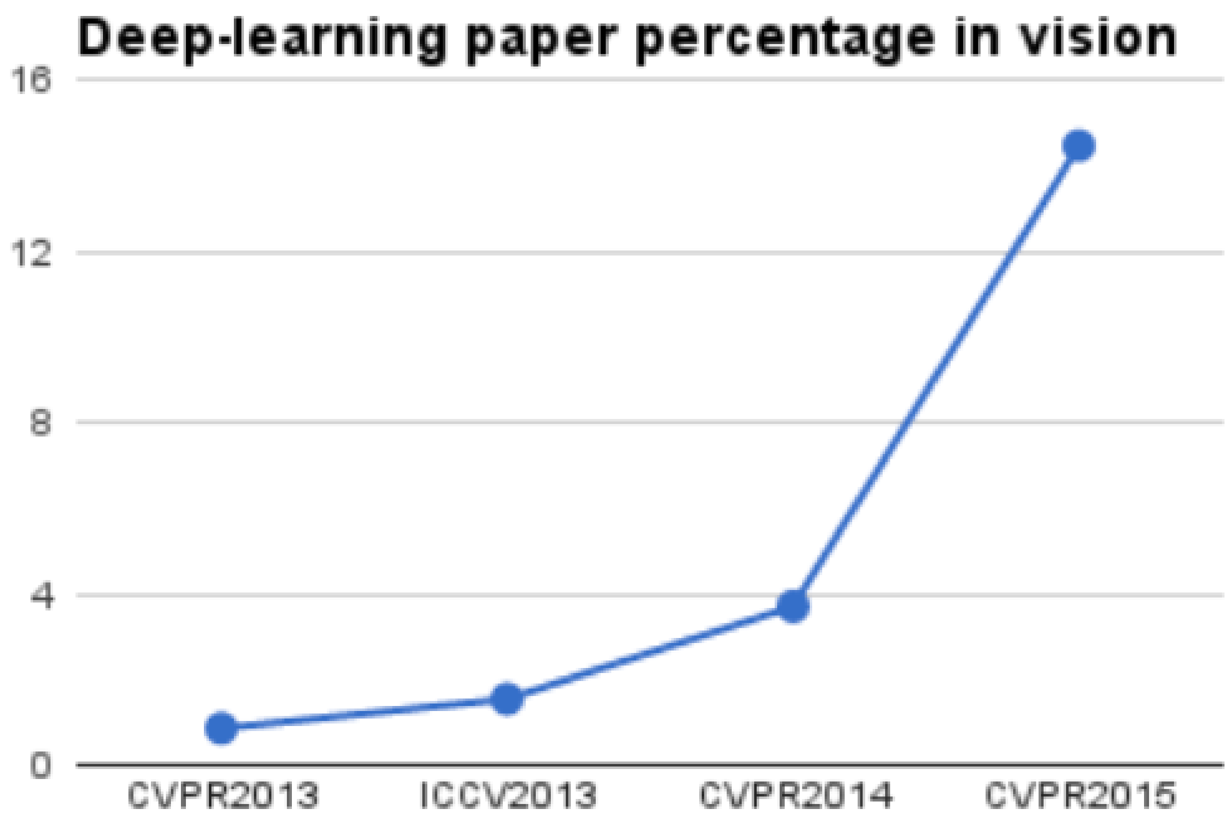


Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks. NeurIPS 2012.

AlexNet Analysis

- Achieved unexpected, unprecedented improvements on ImageNet
  - 9.6 percentage point drop in top-5 error to 16.4% compared to 2011’s best model
- Signified deeper models help, as removing any convolutional layer led to inferior performance
- Open challenge for going deeper: GPUs’ limited amount of memory and the excessive training time

AlexNet: Catalyst for Deep Learning Revolution



Inspired, many more researchers in the computer vision community focused on neural networks for many more vision problems!

https://www.slideshare.net/xavigiro/saliency-prediction-using-deep-learning-techniques

Recap: Ideas for Training a Large Capacity Model

- Enable learning: use non-saturating activation functions (ReLUs)
- Prevent overfitting: incorporate regularization methods (data augmentation and dropout)
- Make training feasible: speed it up with better hardware (GPUs)

Today’s Topics

- Key challenge: training large capacity, deep models
- AlexNet: key tricks for going 8 layers deep
- ResNet: key tricks for extending to 152 layers deep
- Programming tutorial

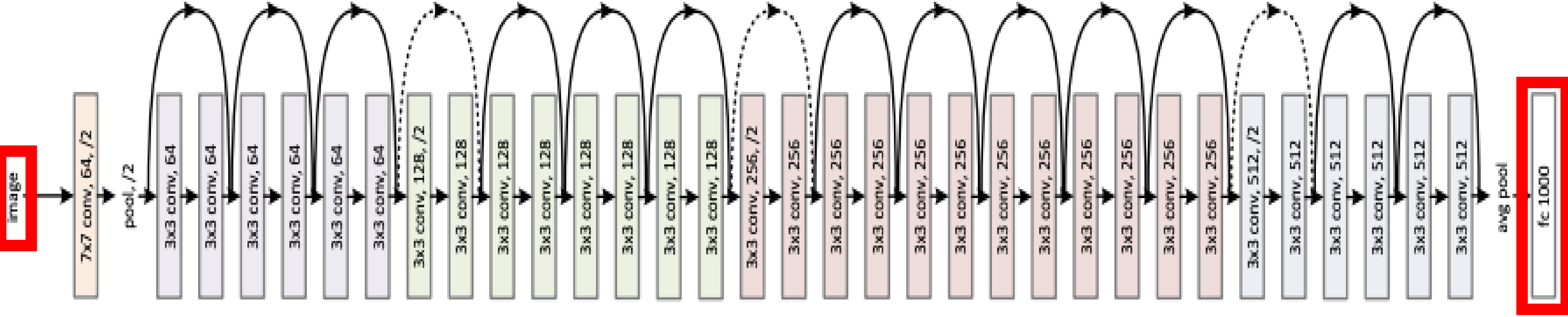
(Model Named After Method)

(2016, CVPR)

Deep Residual Learning for Image Recognition

Kaiming He   Xiangyu Zhang   Shaoqing Ren   Jian Sun
Microsoft Research

ResNet Architecture: Shown is Subset of Layers (34 of 152 Layers)



Input: RGB image resized to fixed input size

Output: 1000 class probabilities (sums to 1)

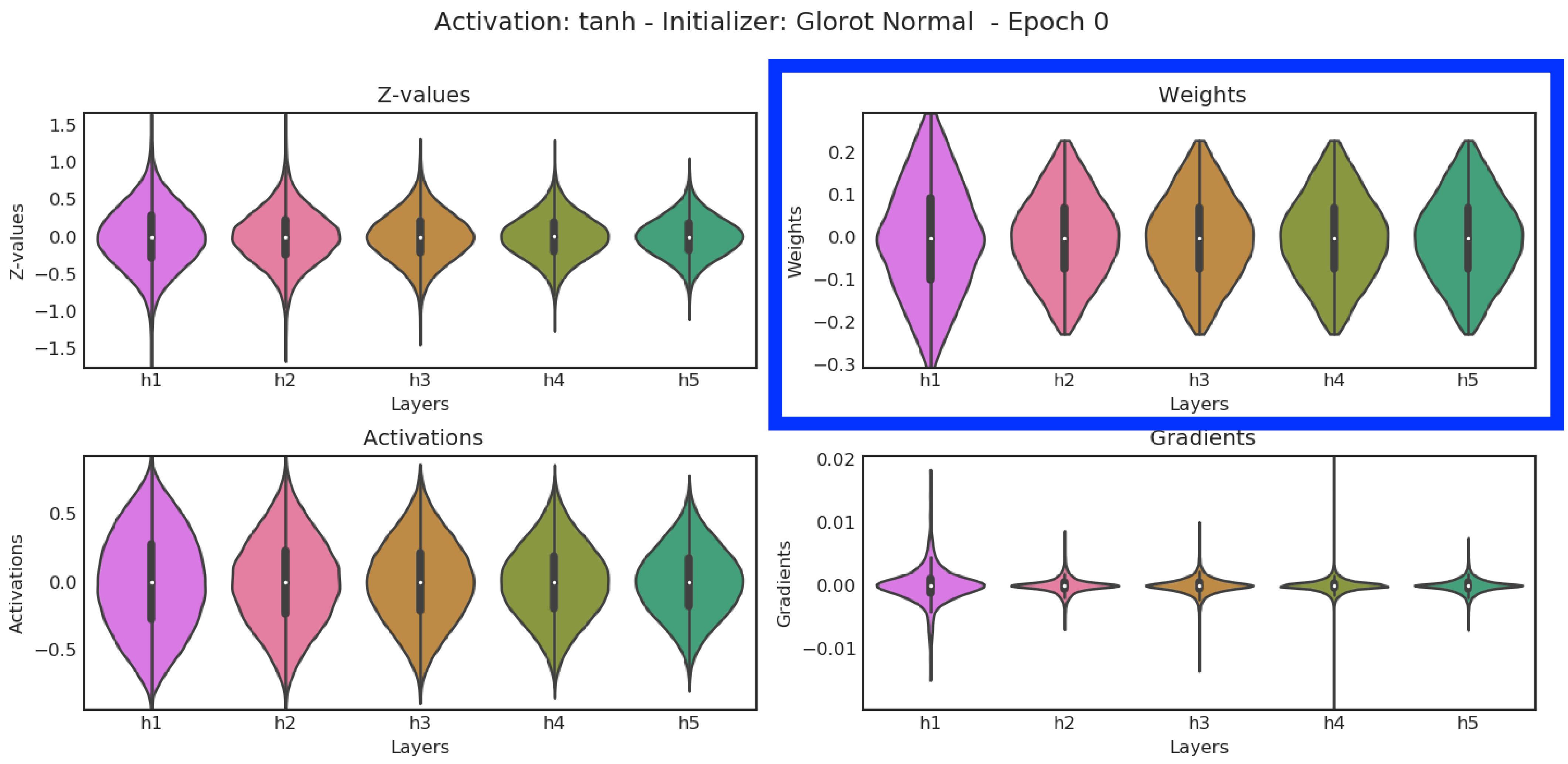
He et al. Deep Residual Learning for Image Recognition. CVPR 2016.

Key Ideas for Training a Large Capacity Model

- Remove vanishing gradient problem (when using ReLUs): use He initialization and batch normalization

- Resolve performance degradation problem (not overfitting): add shortcut connections and then learn residual functions

Idea 1: Better Initialization Method



Recall: want weights that lead to **gradients** that can support learning

Xavier/Glorot method is a poor match for ReLU, due to its non-linearity

https://towardsdatascience.com/hyper-parameters-in-action-part-ii-weight-initializers-35aee1a28404

Idea 1: He/Kaiming/MSRA Initialization

(2015, ICCV)

Delving Deep into Rectifiers:
Surpassing Human-Level Performance on ImageNet Classification

Kaiming He   Xiangyu Zhang   Shaoqing Ren   Jian Sun

Microsoft Research

Samples weight values from a zero-mean Gaussian with this standard deviation (biases set to 0):

$$\sigma = \sqrt{2.0/n_{in}}$$

fan in: # of neurons entering the layer

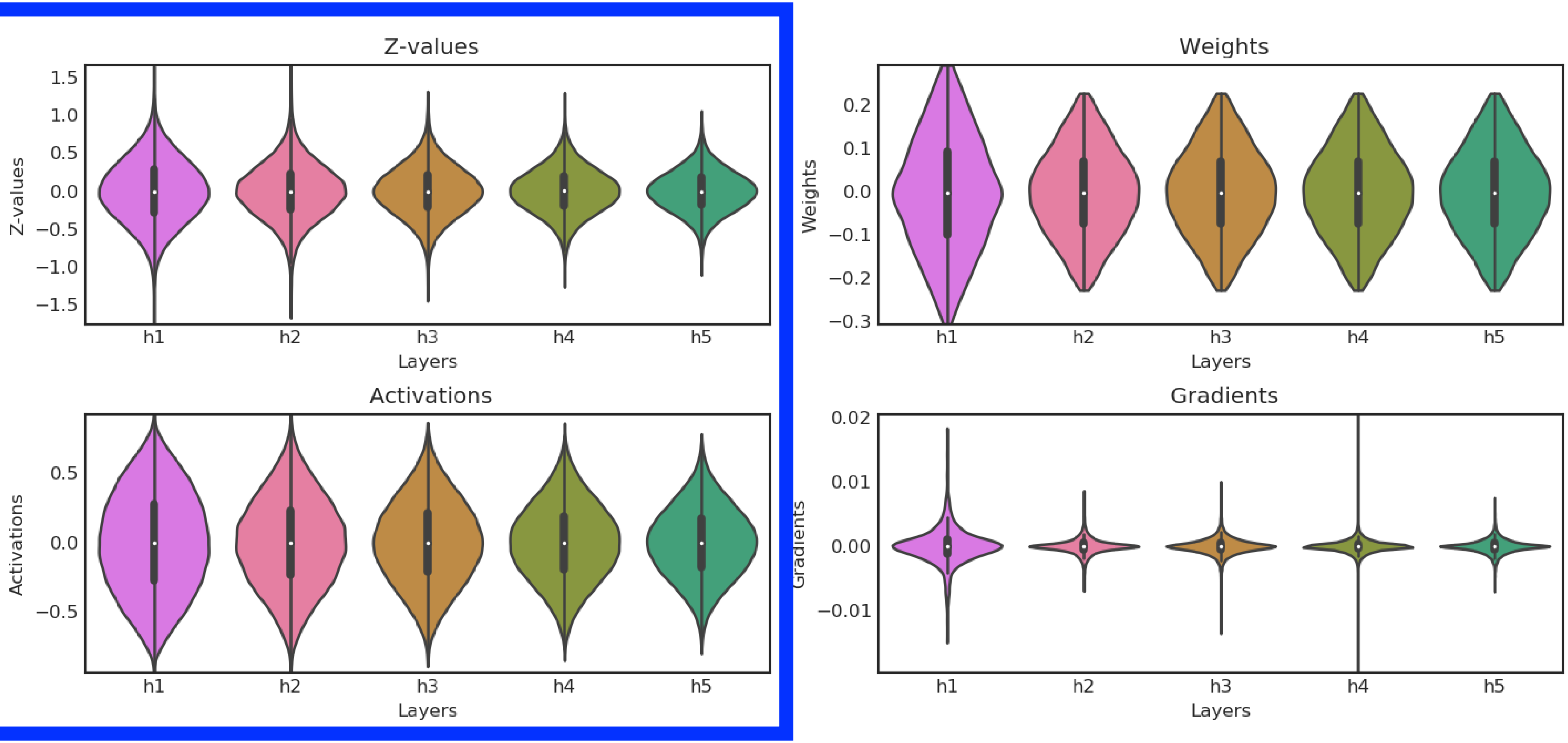
Idea 2: Batch Normalization

(2015, ICML)

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe SIOFFE@GOOGLE.COM
Christian Szegedy SZEGEDY@GOOGLE.COM
Google, 1600 Amphitheatre Pkwy, Mountain View, CA 94043

Idea 2: Batch Normalization

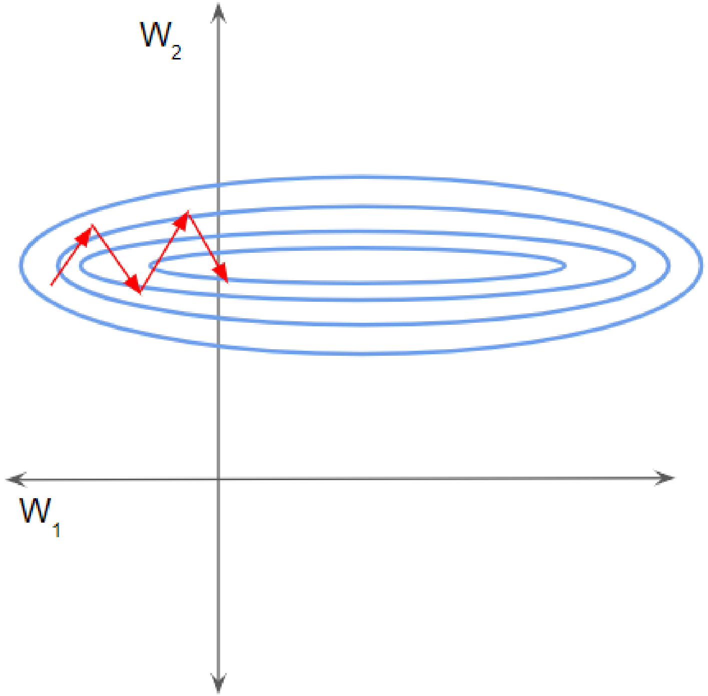


During training, shift values at each layer so resulting gradients support learning

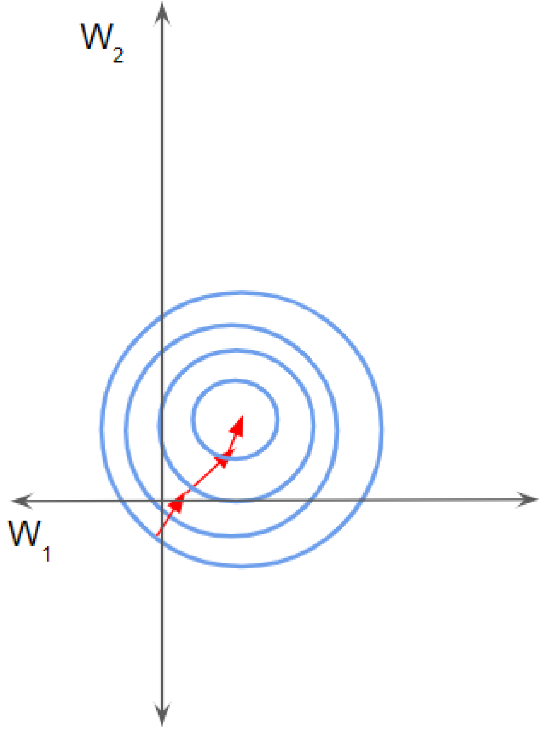
https://towardsdatascience.com/hyper-parameters-in-action-part-ii-weight-initializers-35aee1a28404

Idea 2: Batch Normalization

Simplifies learning by standardizing output of each hidden before passing them to the next layer data so mean and standard deviation accelerate learning (similar to data initialization)

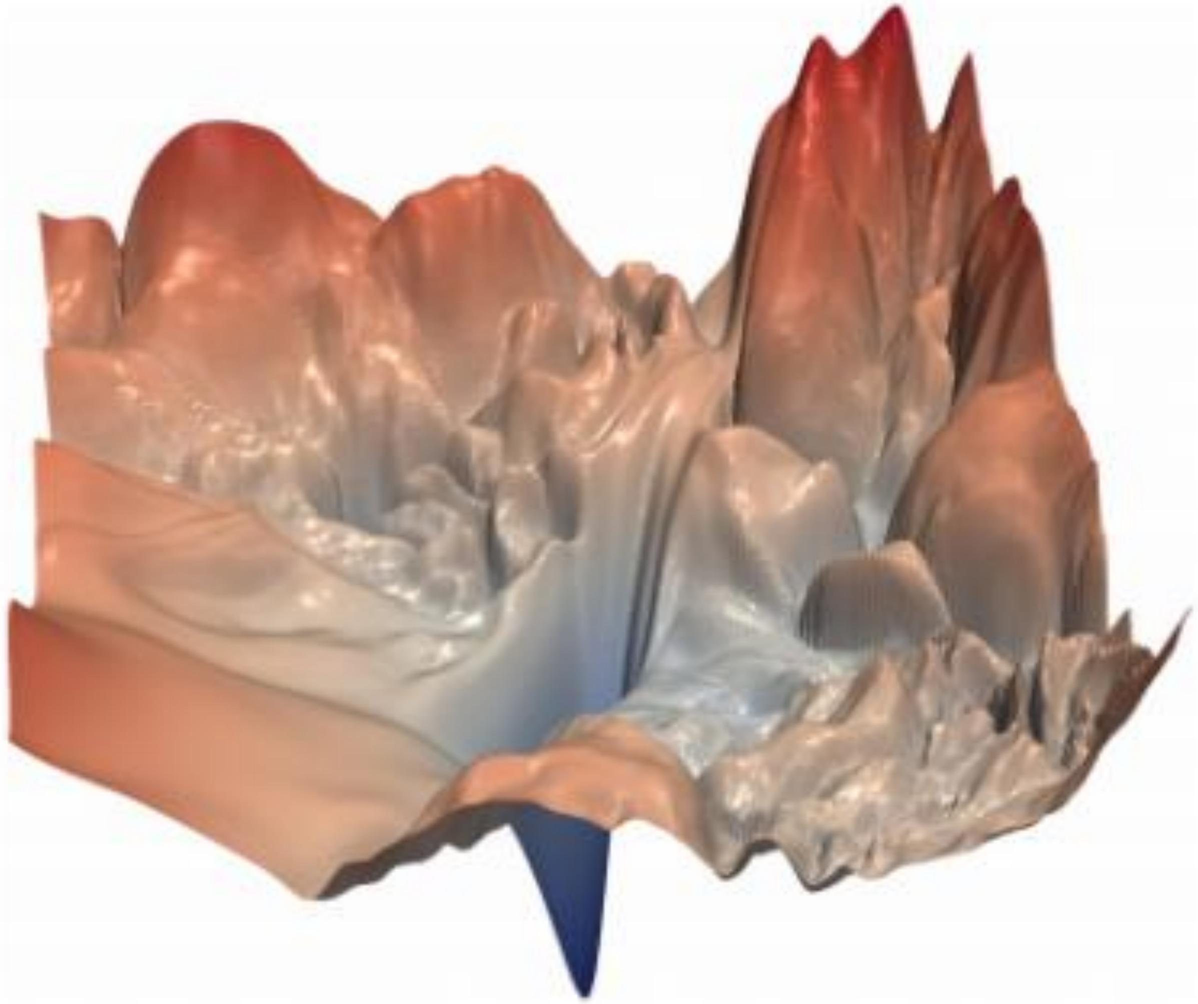


Feature standardization changes the loss function so gradient descent can more smoothly arrive at the minimum!



https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

Idea 2: Batch Normalization



Intuitively, smoothing a loss function's error surface removes:

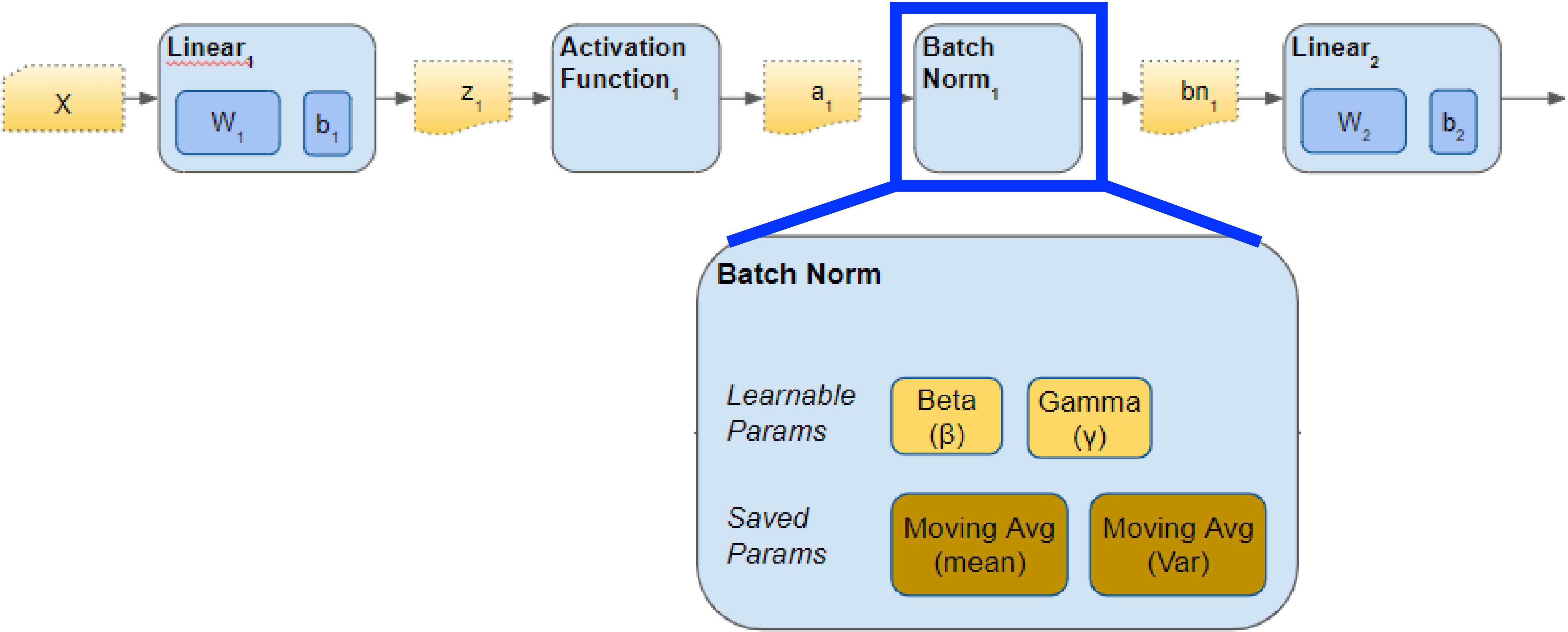
- flat regions, which cause vanishing gradients

- sharp local minima, which cause "exploding" gradients

Removing dangers in the surface means larger learning rates are more feasible, and so training can also be sped up

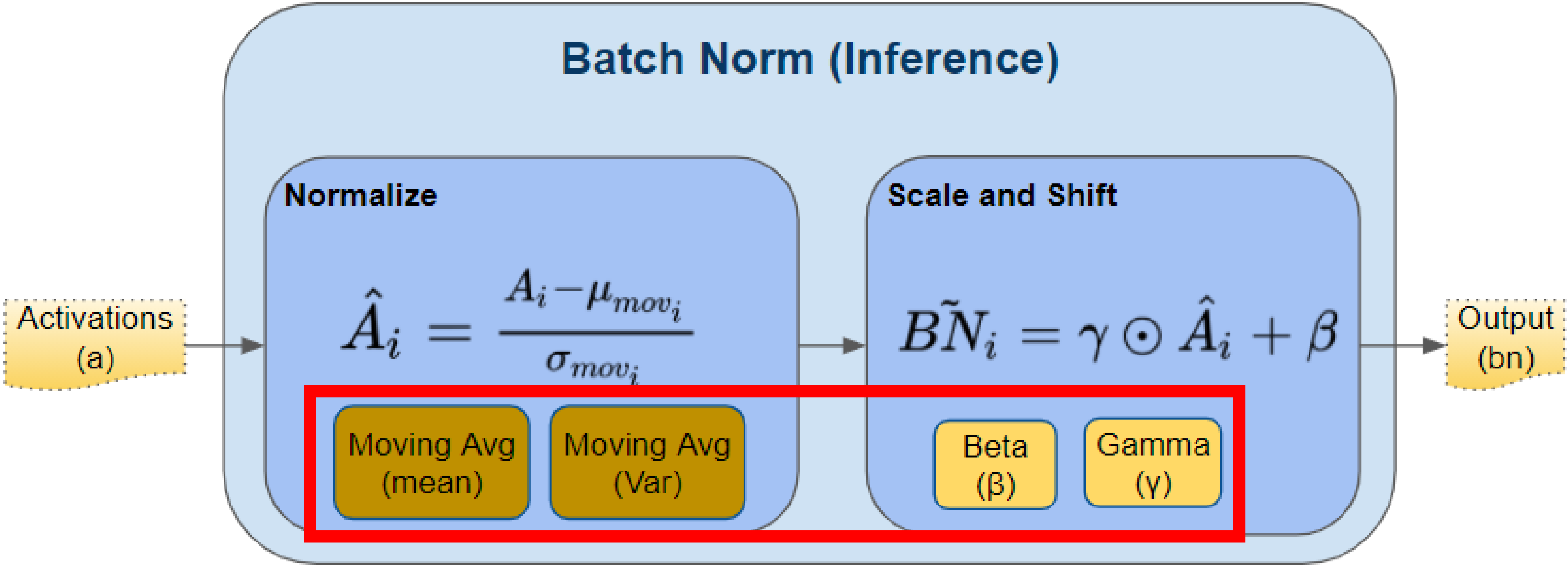
Li et al. Visualizing the Loss Landscape of Neural Nets. Neurips 2018.

Idea 2: Batch Normalization (Inference Time)



https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

Idea 2: Batch Normalization (Inference Time)

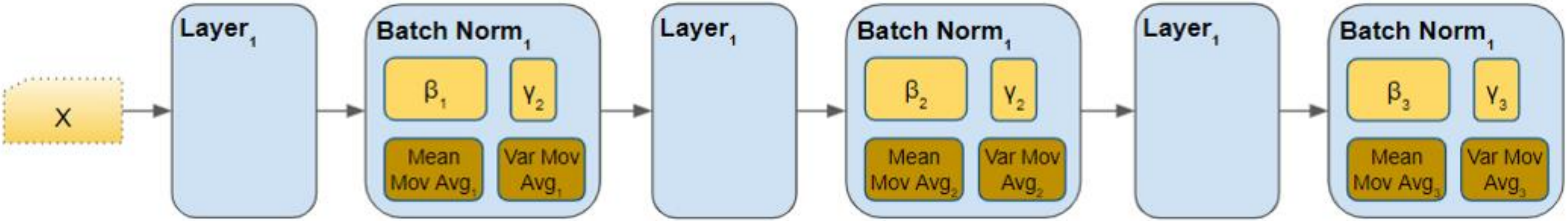


These 4 values would be learned during training

https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

Idea 2: Batch Normalization (Inference Time)

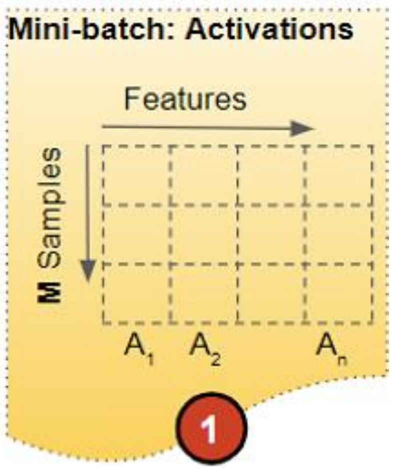
A unique BN module is applied to activation values for every layer!



How many parameters must be learned for this subnetwork?

https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

Idea 2: Batch Normalization (Training Time)

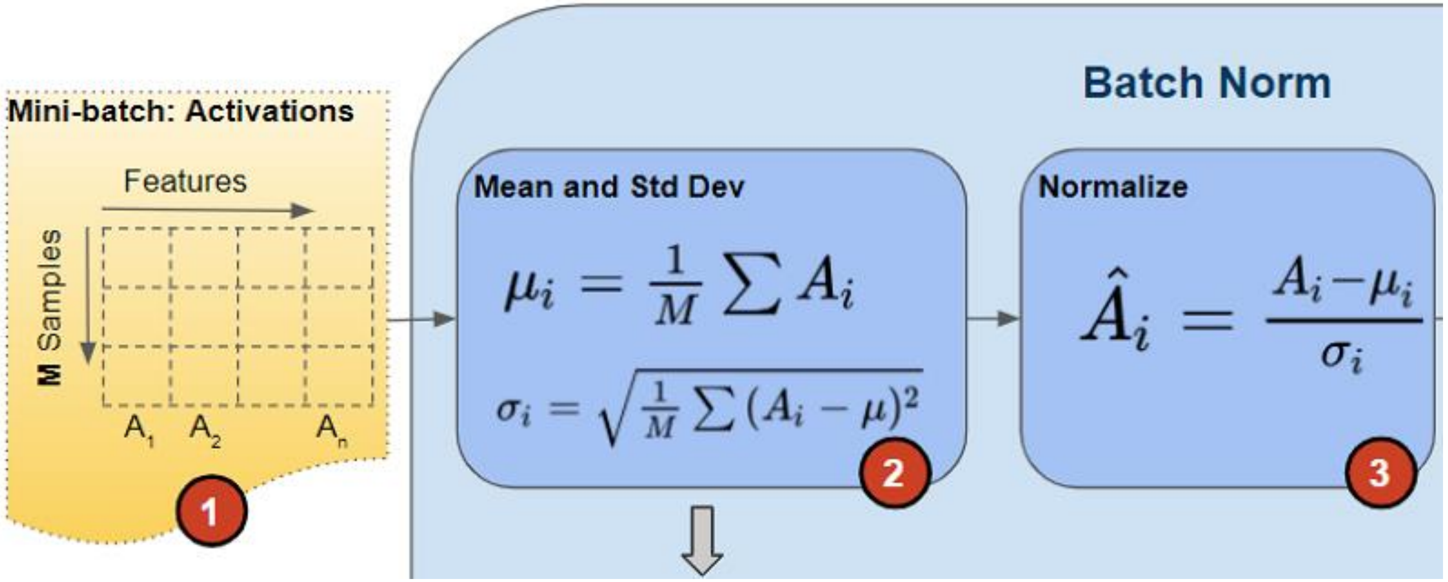


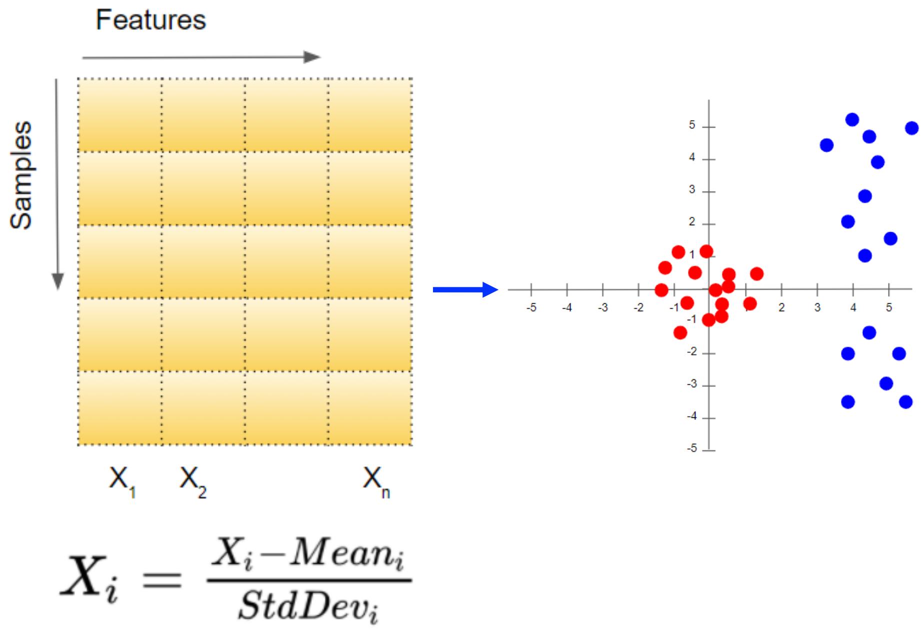
Input: mini-batch

- e.g., assume a fully connected layer, each row represents a unique example and each column represents all outputs (features) from one of the layer's neurons
- How many examples are in the toy example's mini-batch?
- How many neurons are leaving the toy example's layer?

https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

Idea 2: Batch Normalization (Training Time)

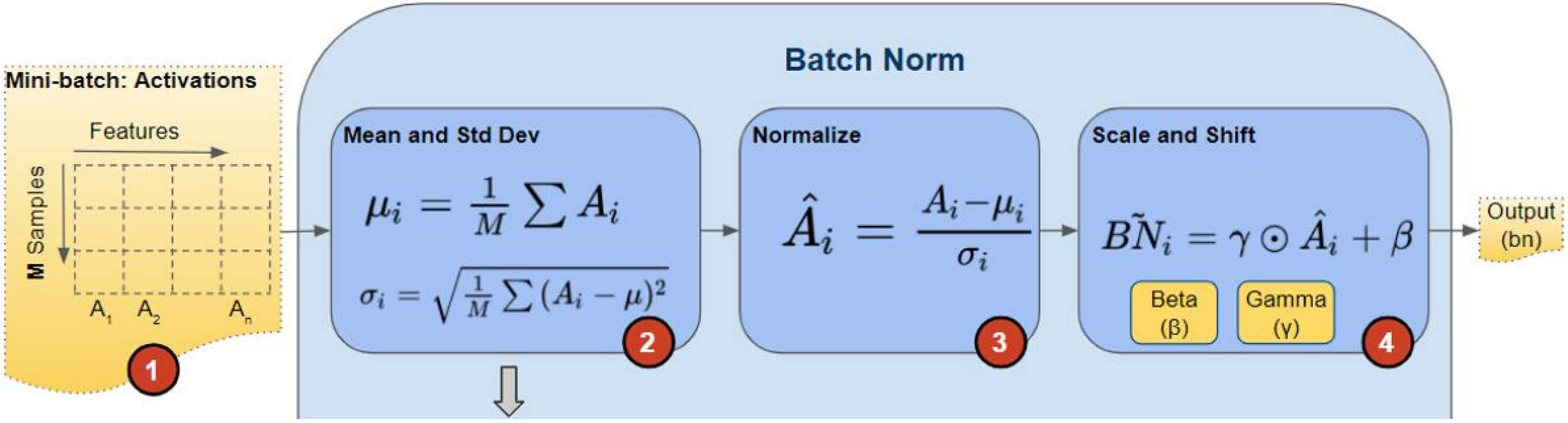




For each feature, compute mean and standard deviation values for all examples and then use those values to modify the features to target distributions

https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

Idea 2: Batch Normalization (Training Time)

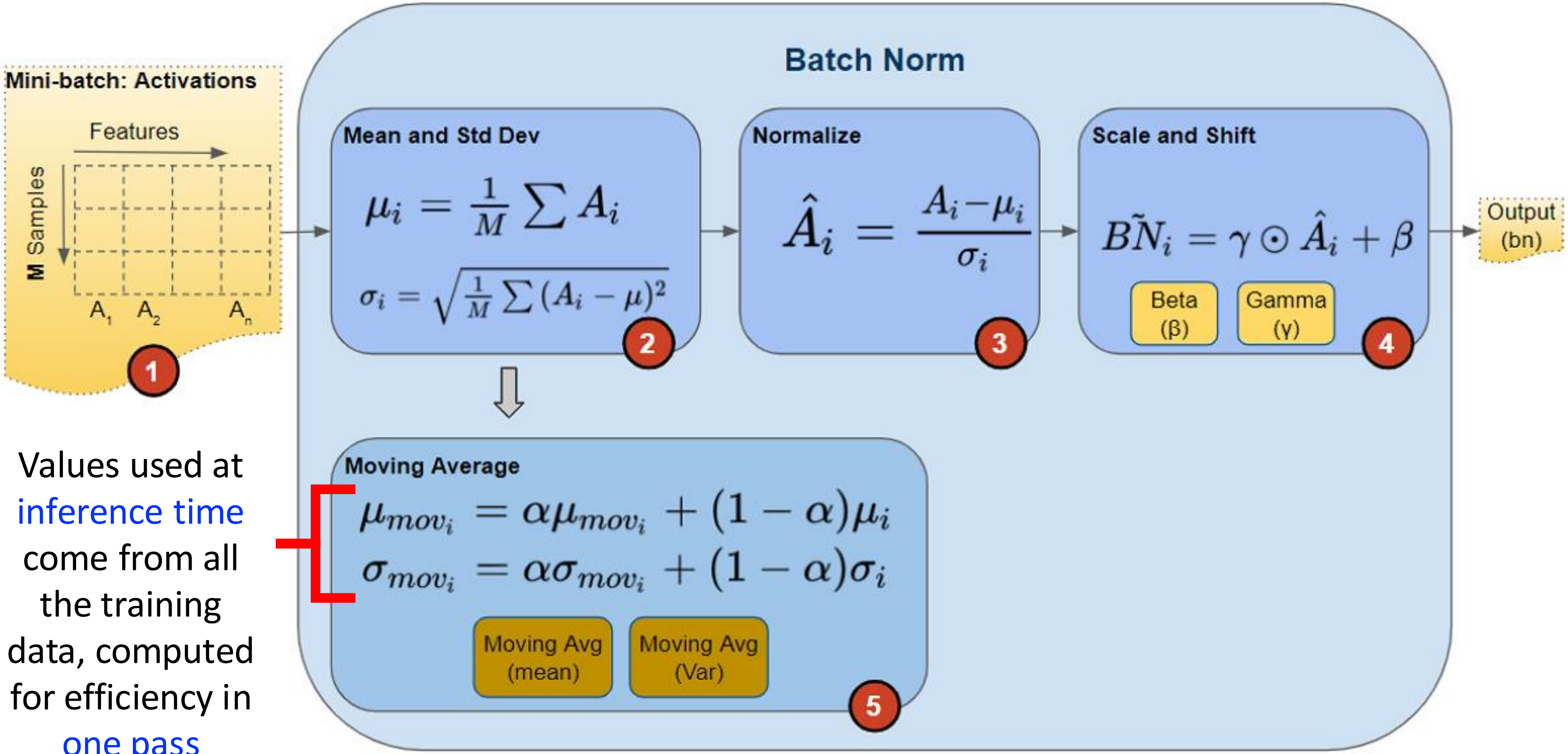


During backpropagation,
further modify distribution
to improve performance

(because we add bias here, we exclude it in earlier layers)

https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

Idea 2: Batch Normalization (Training Time)

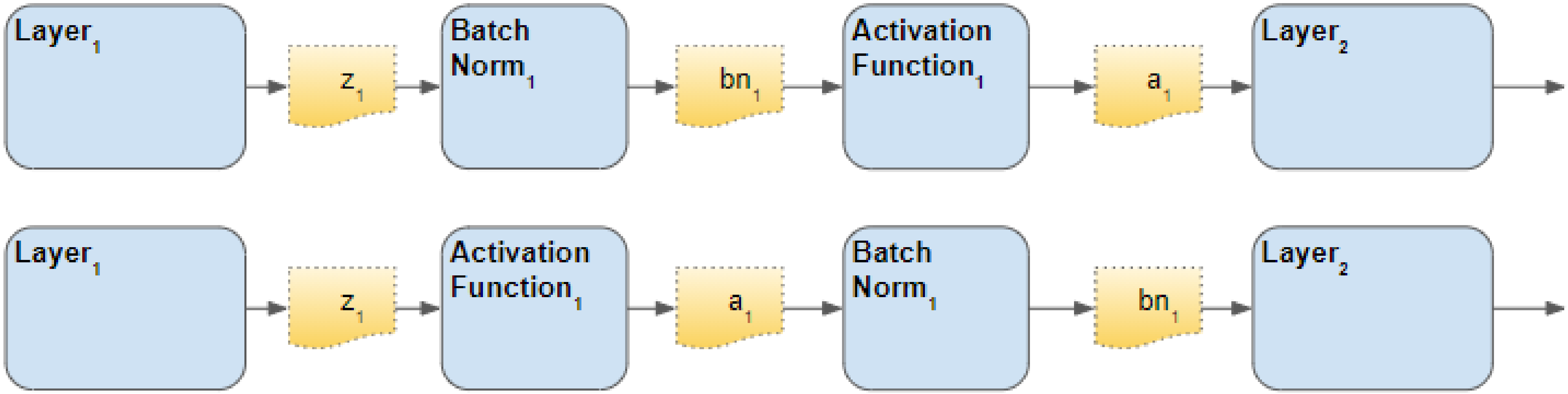


Values used at inference time come from all the training data, computed for efficiency in one pass

https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

Idea 2: Batch Normalization Implementation

ResNet adopts BN before activations are computed; however, it is also common to apply BN after computing activations



https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

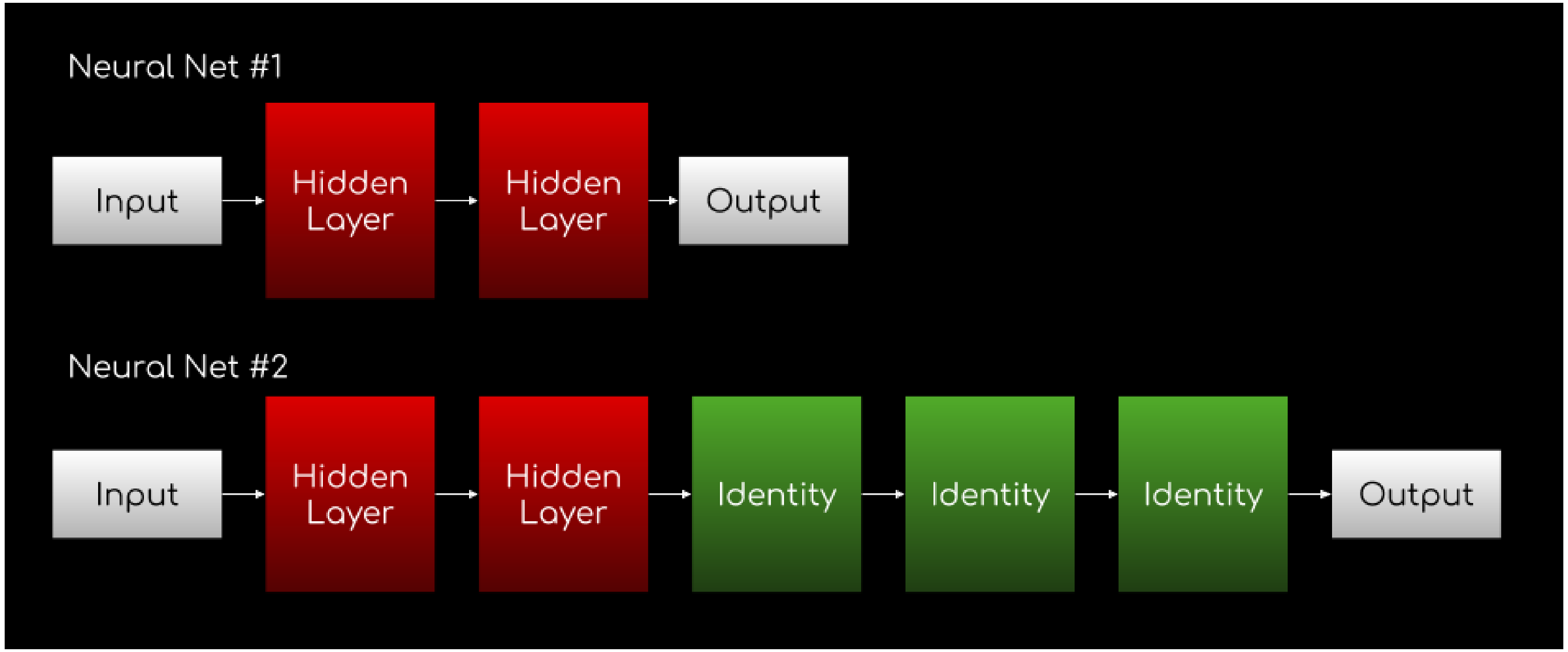
Key Ideas for Training a Large Capacity Model

- Remove vanishing gradient problem (when using ReLUs): use He initialization and batch normalization

- Resolve performance degradation problem (not overfitting): add shortcut connections and then learn residual functions

Motivating Observation

- A deeper network should perform at least as good as a shallower network since it can learn the shallower function alongside “identity” functions for later layers



- However, experimentally, adding more layers led to WORSE results!

https://medium.com/@realmichaelye/intuition-for-resnet-deep-residual-learning-for-image-recognition-39d24d173e78

What is the Problem for Learning?

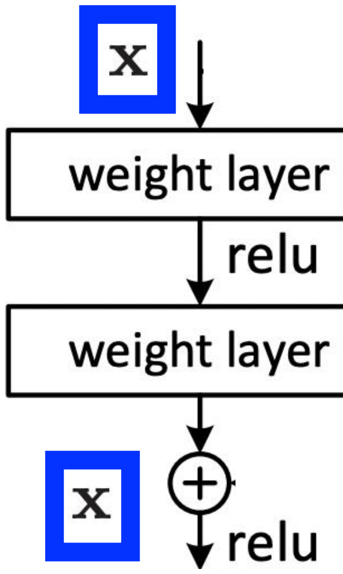
- Vanishing gradients? Unlikely; desired gradients observed when training with both aforementioned tricks (He initialization and BN)
- Overfitting? No

- Degradation problem! Accuracy saturates before declining rapidly from more layers
- **Hypothesis:** difficult to learn identity mappings

He et al. Deep Residual Learning for Image Recognition. CVPR 2016.

Problem: Difficult to Perform Identity Mapping

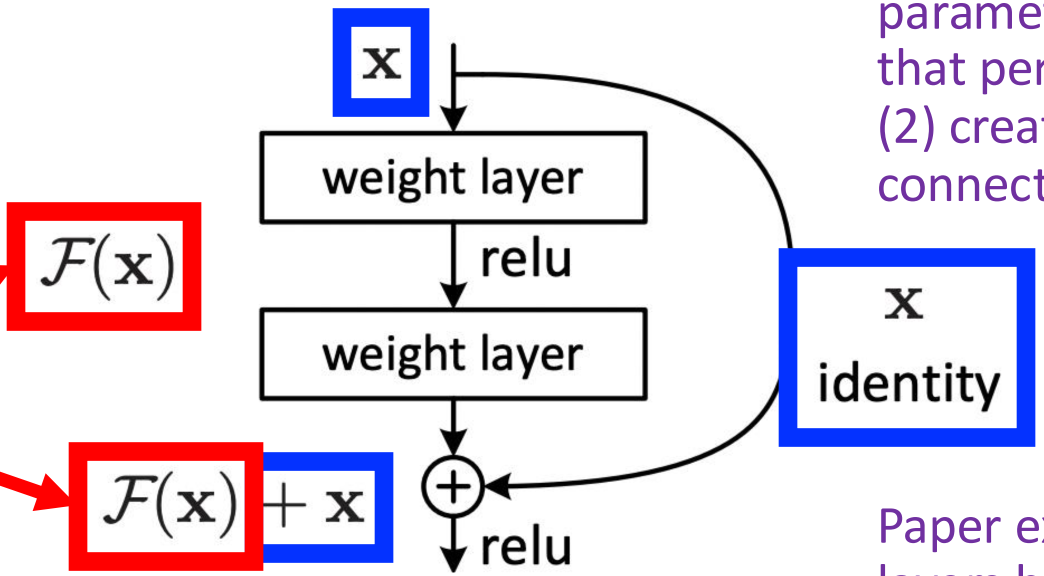
e.g.,



He et al. Deep Residual Learning for Image Recognition. CVPR 2016.

Idea: Add Skip Connections to Enable Learning Identity Mapping

Residual function to be learned



Hypotheses: (1) easier to learn parameters close to 0 than a function that performs identity mapping and (2) creating shorter paths with skip connections will simplify learning

Paper explores when number of layers between skip connections (i.e., residual block) is 2 or 3

Forces model to learn the identity function when minimizing the loss

(Recall: derivative for an addition operation means there is no change to the gradient flow, as incoming gradients are multiplied by one)

He et al. Deep Residual Learning for Image Recognition. CVPR 2016.

Idea: Add Skip Connections to Enable Learning Identity Mapping

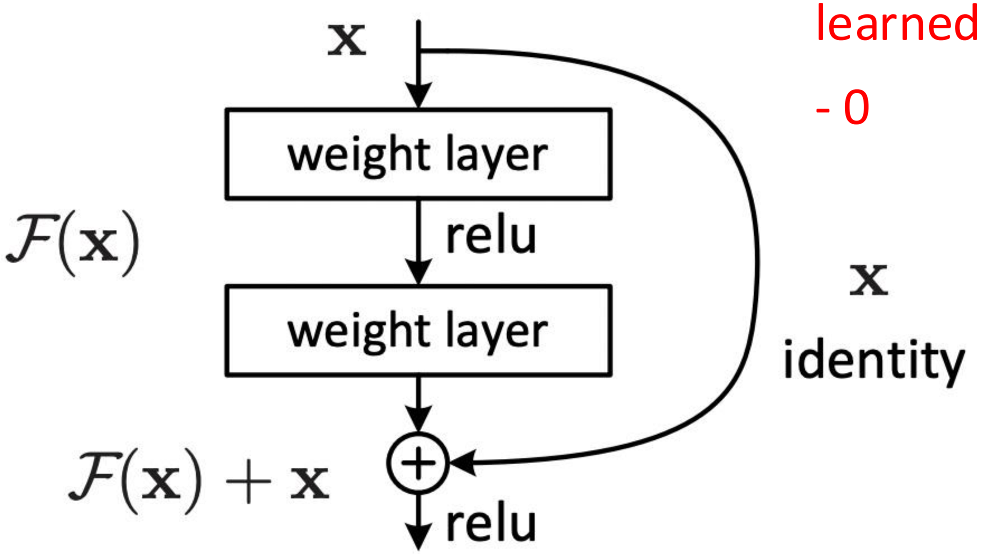


Figure 2. Residual learning: a building block.

How many model parameters need to be learned when adding a skip connection?

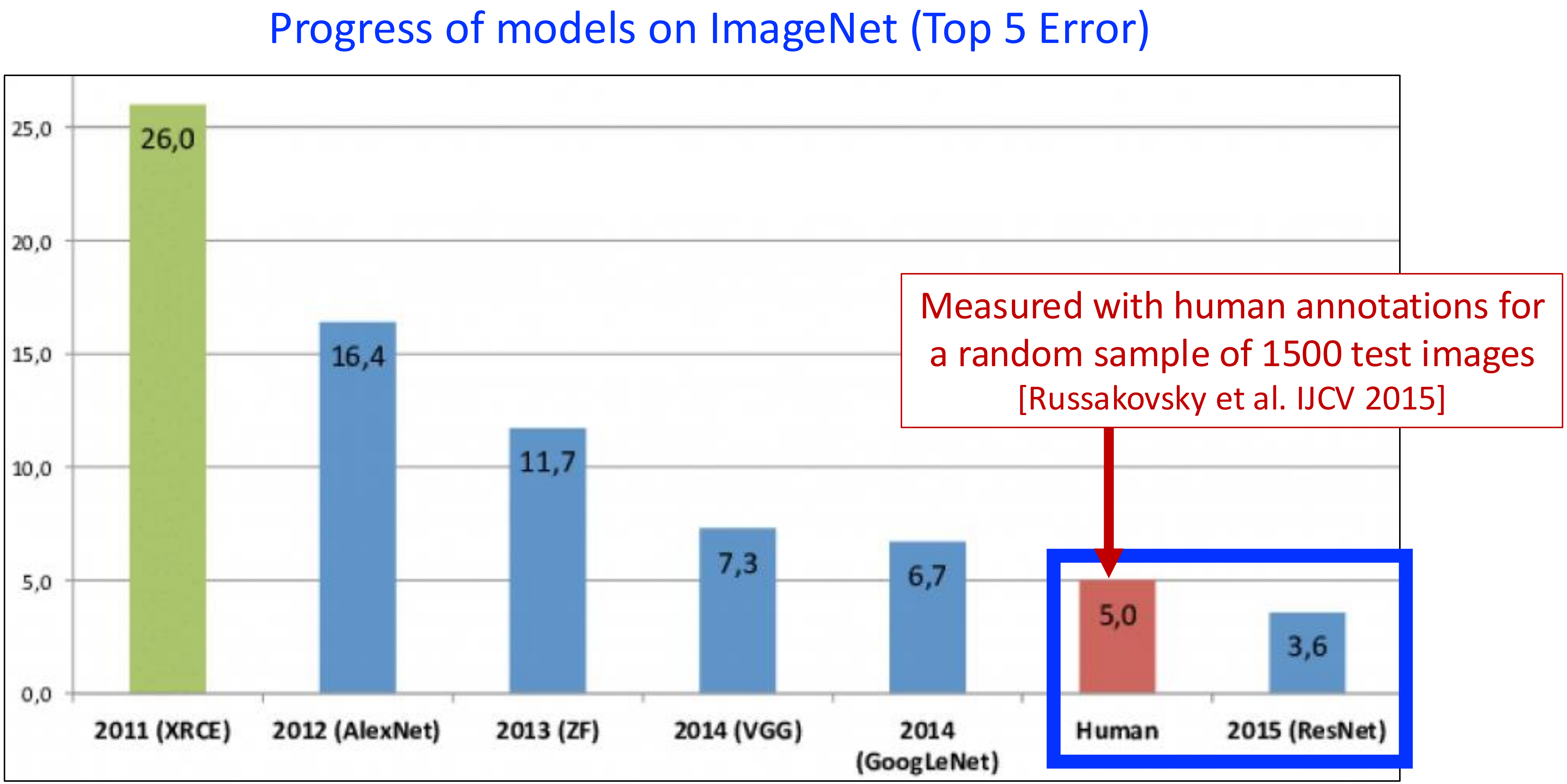
- 0

He et al. Deep Residual Learning for Image Recognition. CVPR 2016.

Training Setting: Slight Change from AlexNet

- Batch size: 256 examples (double amount for AlexNet)
- Initialization: He method
- Momentum optimization: manually adjusted learning rate from initial 0.1 (vs 0.01 for AlexNet), dividing by 10 when validation error plateaued
- No dropout used in order to isolate analysis on overcoming optimization issues with skip connections and residual learning

ResNet Performance: Exceeded Humans!



https://www.edge-ai-vision.com/2018/07/deep-learning-in-five-and-a-half-minutes/

Recap: Ideas for Training a Large Capacity Model

- Remove vanishing gradient problem (when using ReLUs):
  use He initialization and batch normalization

- Resolve performance degradation problem (not overfitting):
  add shortcut connections and then learn residual functions

Today’s Topics

- Key challenge: training large capacity, deep models
- AlexNet: key tricks for going 8 layers deep
- ResNet: key tricks for extending to 152 layers deep
- Programming tutorial

Today’s Topics

- Key challenge: training large capacity, deep models
- AlexNet: key tricks for going 8 layers deep
- ResNet: key tricks for extending to 152 layers deep
- Programming tutorial


The End