Regularization

Danna Gurari

University of Colorado Boulder

Spring 2022

Review

- Last lecture:
  - Computer vision
  - Era of dataset challenges
  - MNIST challenge winner: LeNet
  - ImageNet challenge winners: deeper learning (AlexNet, VGG, ResNet)
  - Programming tutorial
- Assignments (Canvas)
  - Lab assignment 2 due next week
- Questions?

Today’s Topics

- Regularization
- Parameter norm penalty
- Early stopping
- Dataset augmentation
- Dropout
- Batch Normalization

Today's Topics

- Regularization
- Parameter norm penalty
- Early stopping
- Dataset augmentation
- Dropout
- Batch Normalization

What is Regularization?

"any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error."

- Ch. 5.2 of Goodfellow book on Deep Learning

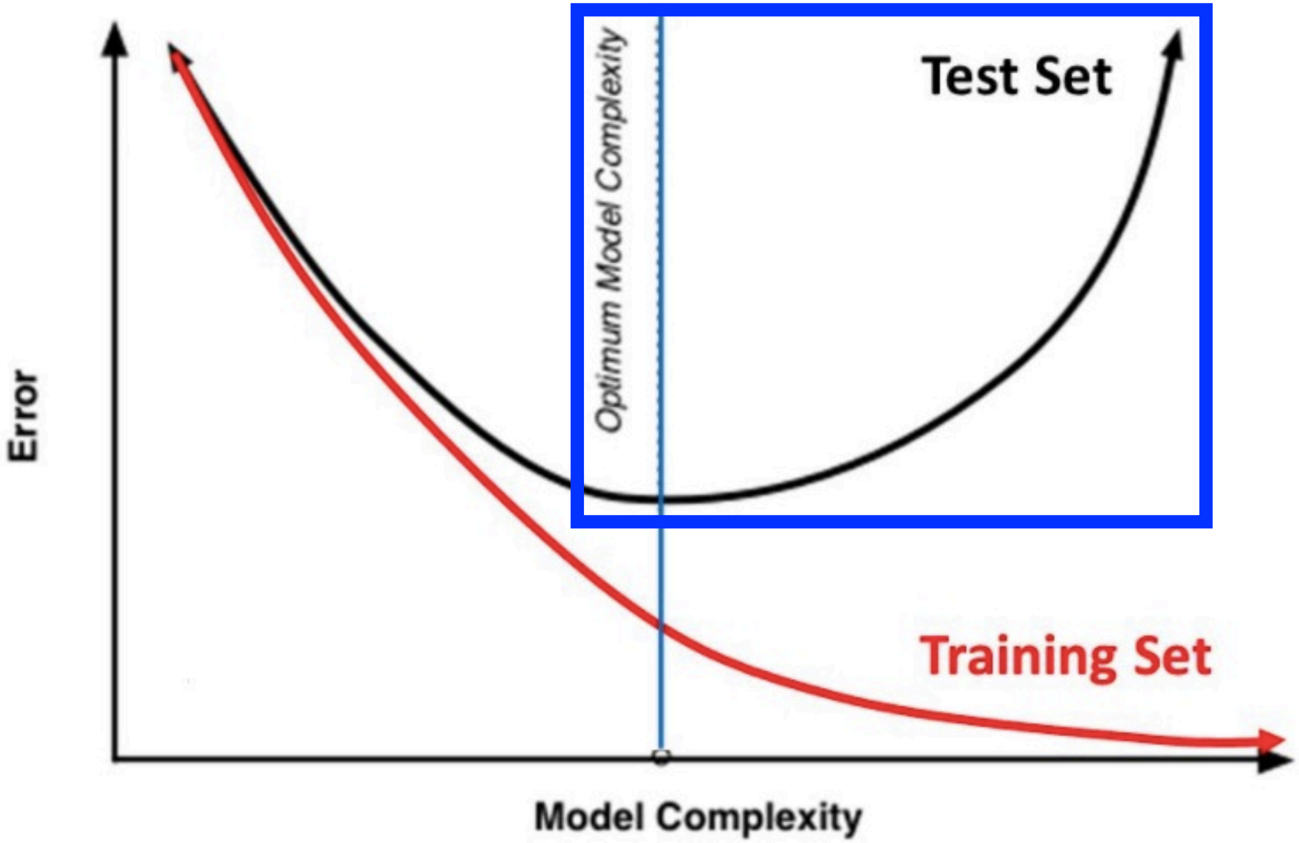
What are strategies for preferring one function over another?

Today’s Topics

- Regularization
- Parameter norm penalty
- Early stopping
- Dataset augmentation
- Dropout
- Batch Normalization

Goal

Rather than exclude functions from a hypothesis space, apply strategies that create a preference for one solution over another to reduce test error



https://cdn.analyticsvidhya.com/wp-content/uploads/2018/04/Screen-Shot-2018-04-04-at-2.43.37-PM.png

Goal

Rather than exclude functions from a hypothesis space, apply strategies that create a preference for one solution over another to reduce test error; e.g., regularize (c)

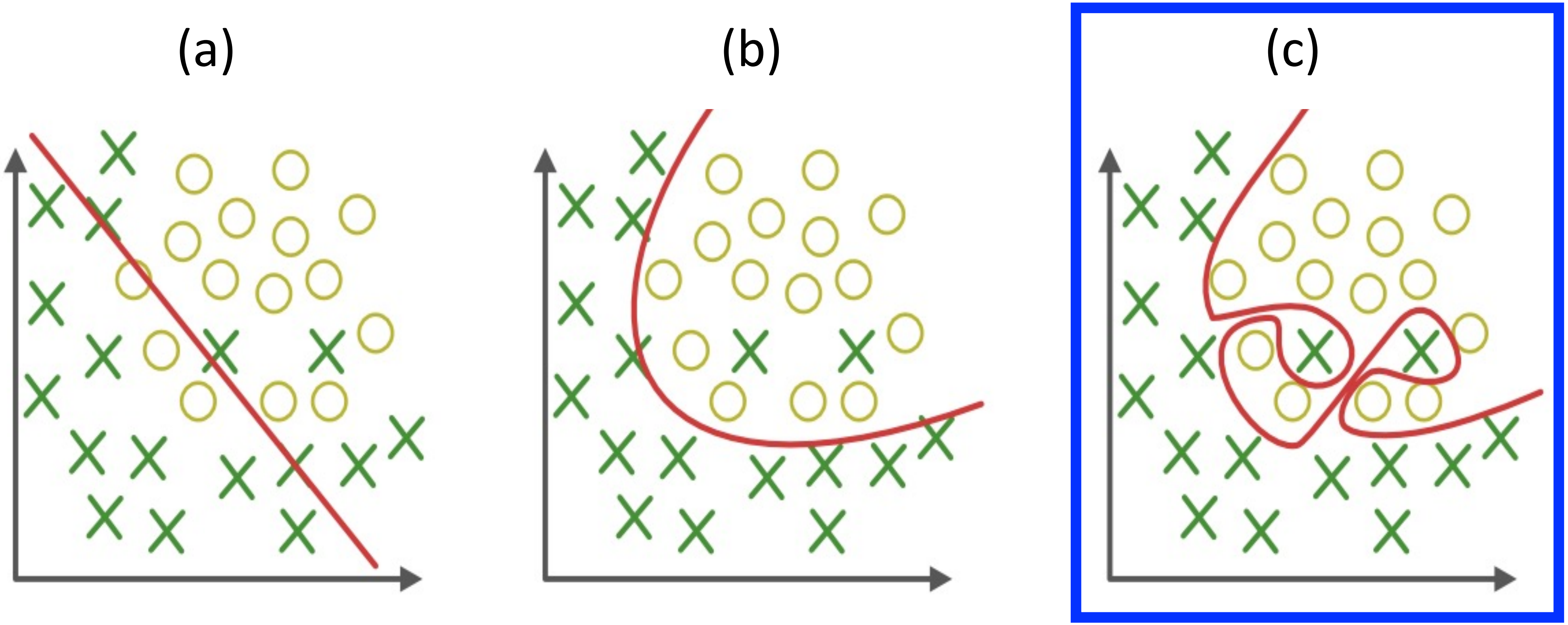
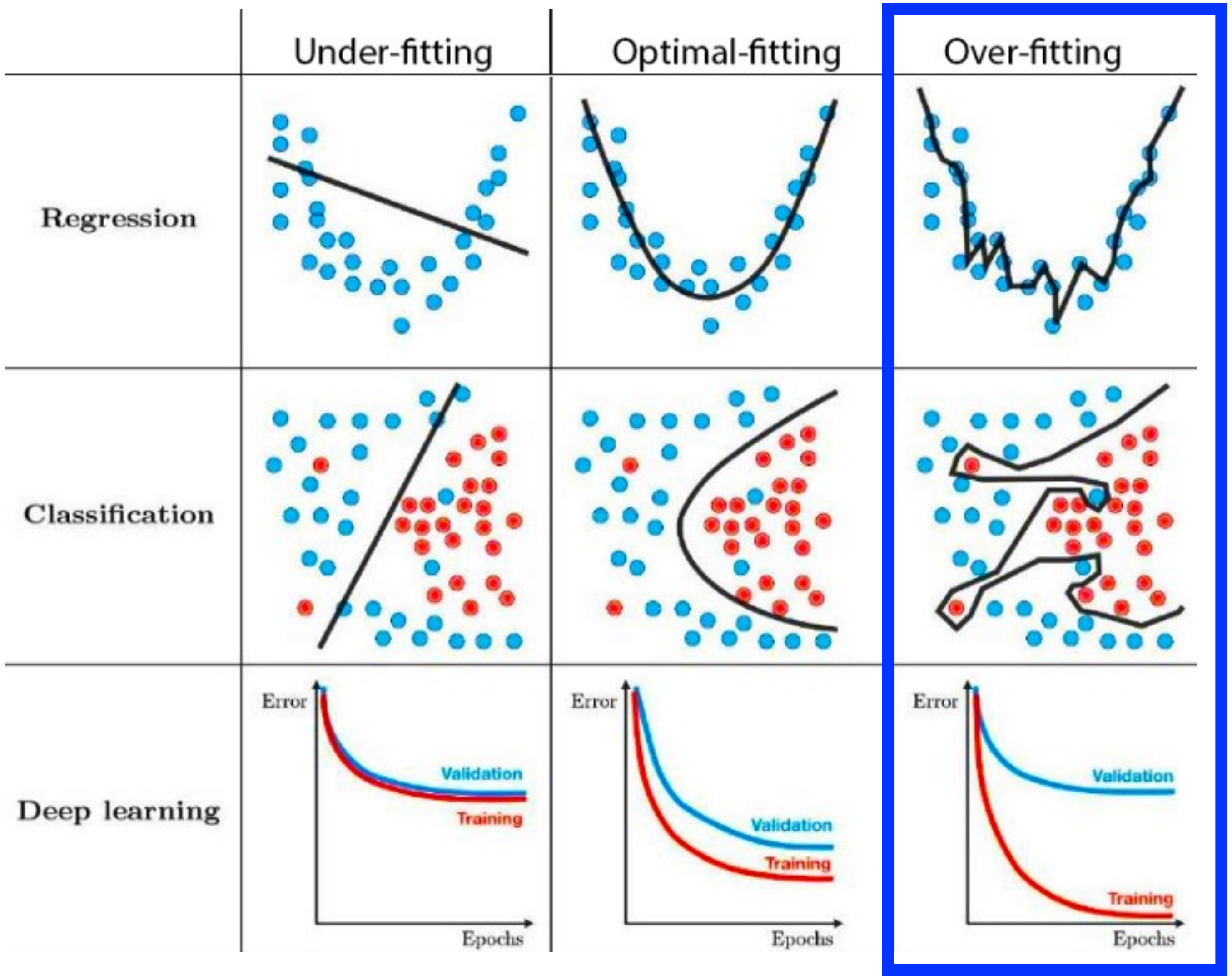


Figure source: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

Insight: Sign of Overfitting is Large Weights

Very large positive weights get canceled by similarly large negative weights (i.e., due to correlated model parameters) in order to model noise



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

Idea: Analogous to Wearing Belt on Big Pants

Idea: Penalize Large Weights in Objective Function

e.g., objective is to *minimize* sum of squared errors over training examples

- L2 norm: penalize squared weight values

$$Error = \boxed{\sum_{i=1}^{n} (y^{(i)} - \widehat{y}^{(i)})^2} + \alpha \boxed{\sum_{j=1}^{m} w_j^2}$$

- L1 norm: penalize absolute weight values

$$Error = \boxed{\sum_{i=1}^{n} (y^{(i)} - \widehat{y}^{(i)})^2} + \alpha \boxed{\sum_{j=1}^{m} |w_j|}$$

- *Note: only weights are penalized, not bias terms*

Idea: Penalize Large Weights in Objective Function

e.g., objective is to *minimize* **s**um of **s**quared **e**rrors over training examples

- L2 norm: penalize squared weight values

$$Error = \sum_{i=1}^{n}(y^{(i)} - \widehat{y}^{(i)})^2 + \boxed{\alpha}\sum_{j=1}^{m} w_j^2$$

- L1 norm: penalize absolute weight values

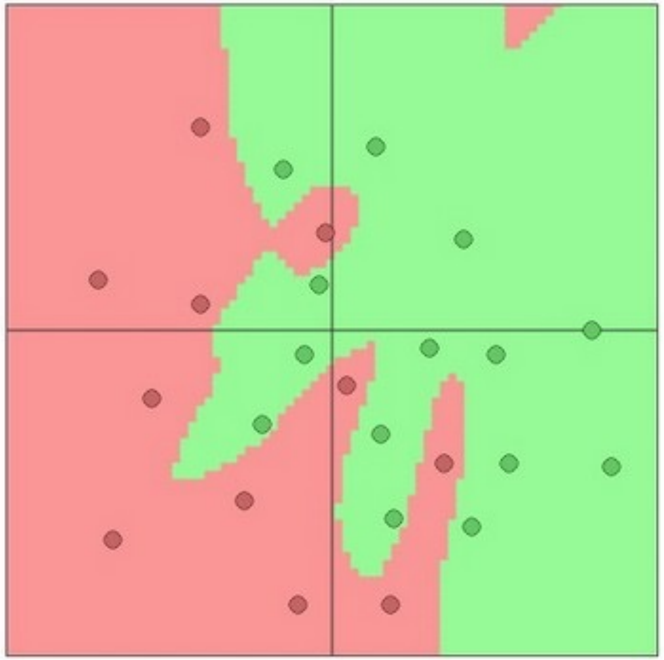
$$Error = \sum_{i=1}^{n}(y^{(i)} - \widehat{y}^{(i)})^2 + \boxed{\alpha}\sum_{j=1}^{m} |w_j|$$

- Hyperparameter determines relative contribution of norm penalty term

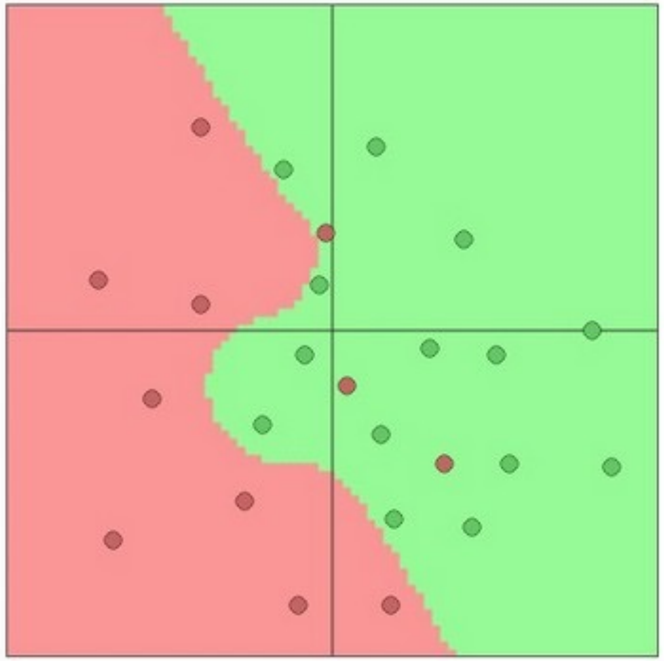
Regularization: How to Set Alpha?

Shown is the same neural network with different levels of regularization. Which model has the largest value for alpha (i.e., largest norm penalty contribution)?

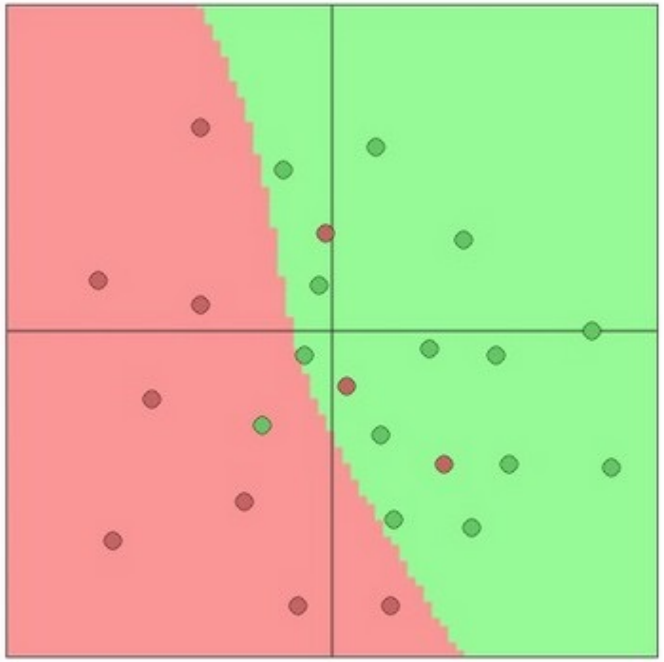
(a)



(b)

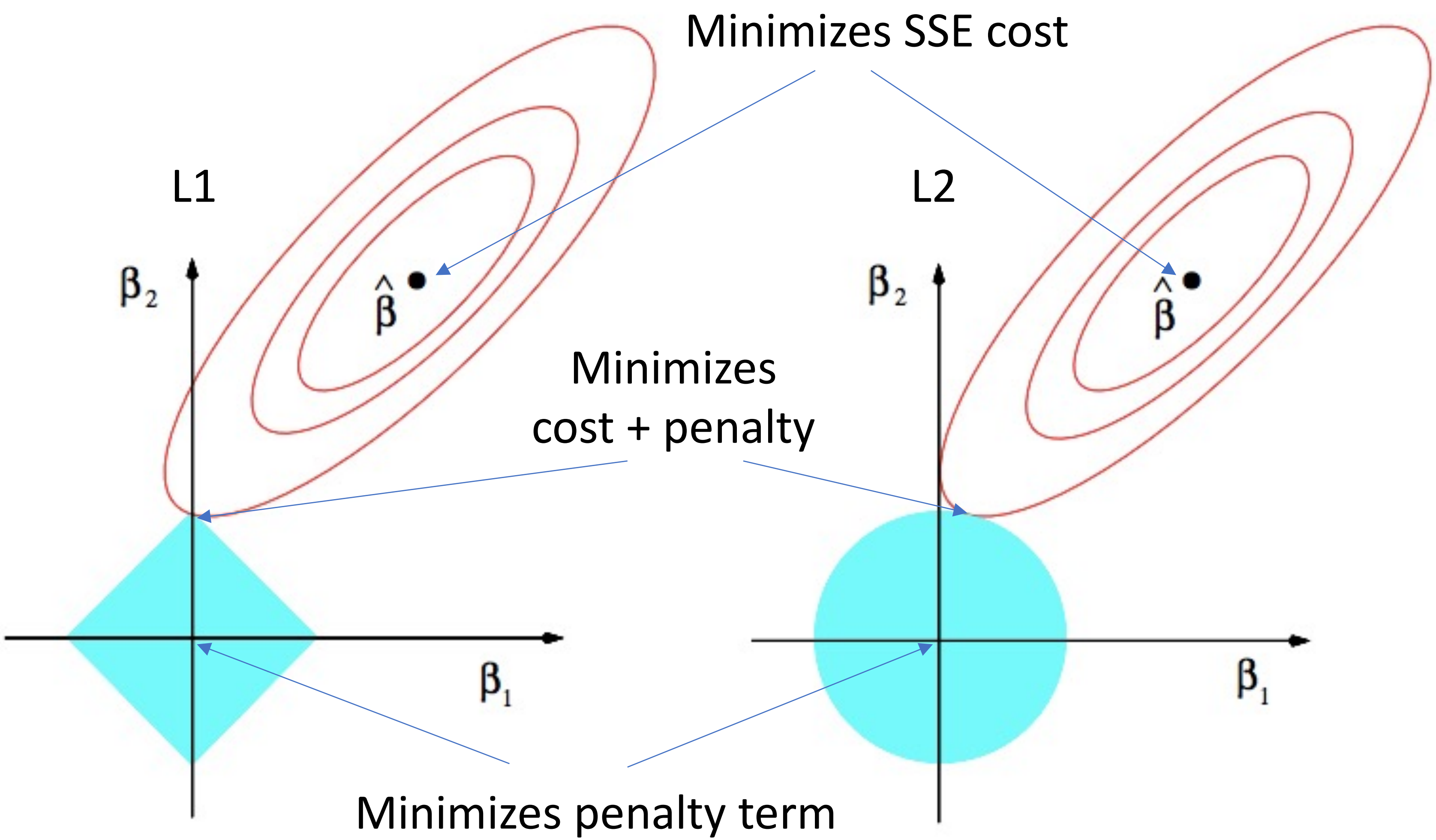


(c)

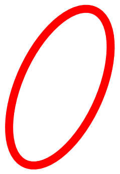


https://cs231n.github.io/neural-networks-1/

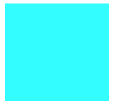
Geometric Interpretation in 2D



$|\beta_1| + |\beta_2| \leq t$ $\beta_1^2 + \beta_2^2 \leq t^2$



Contour of least square error function



Contour of regularization constraint functions shown below both plots

https://web.stanford.edu/~hastie/Papers/ESLII.pdf

Implementation Detail: Can Penalize Weights Globally as Well As Per Layer

e.g., objective is to *minimize* **s**um of **s**quared **e**rrors over training examples

- L2 norm: penalize squared weight values

$$Error = \sum_{i=1}^{n}(y^{(i)} - \widehat{y}^{(i)})^2 + \alpha \boxed{\sum_{j=1}^{m} w_j^2}$$

- L1 norm: penalize absolute weight values

$$Error = \sum_{i=1}^{n}(y^{(i)} - \widehat{y}^{(i)})^2 + \alpha \boxed{\sum_{j=1}^{m} |w_j|}$$

- *Note: only weights are penalized, not bias terms*

Today’s Topics

- Regularization
- Parameter norm penalty
- Early stopping
- Dataset augmentation
- Dropout
- Batch Normalization

Recall: Overfitting Solution is Early Stopping

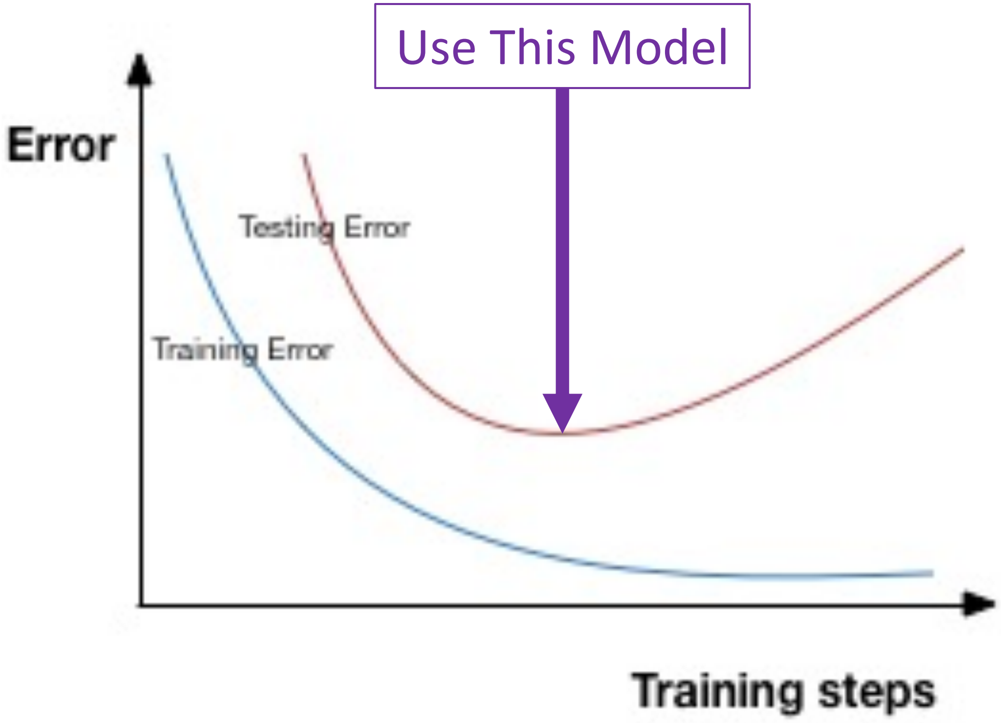
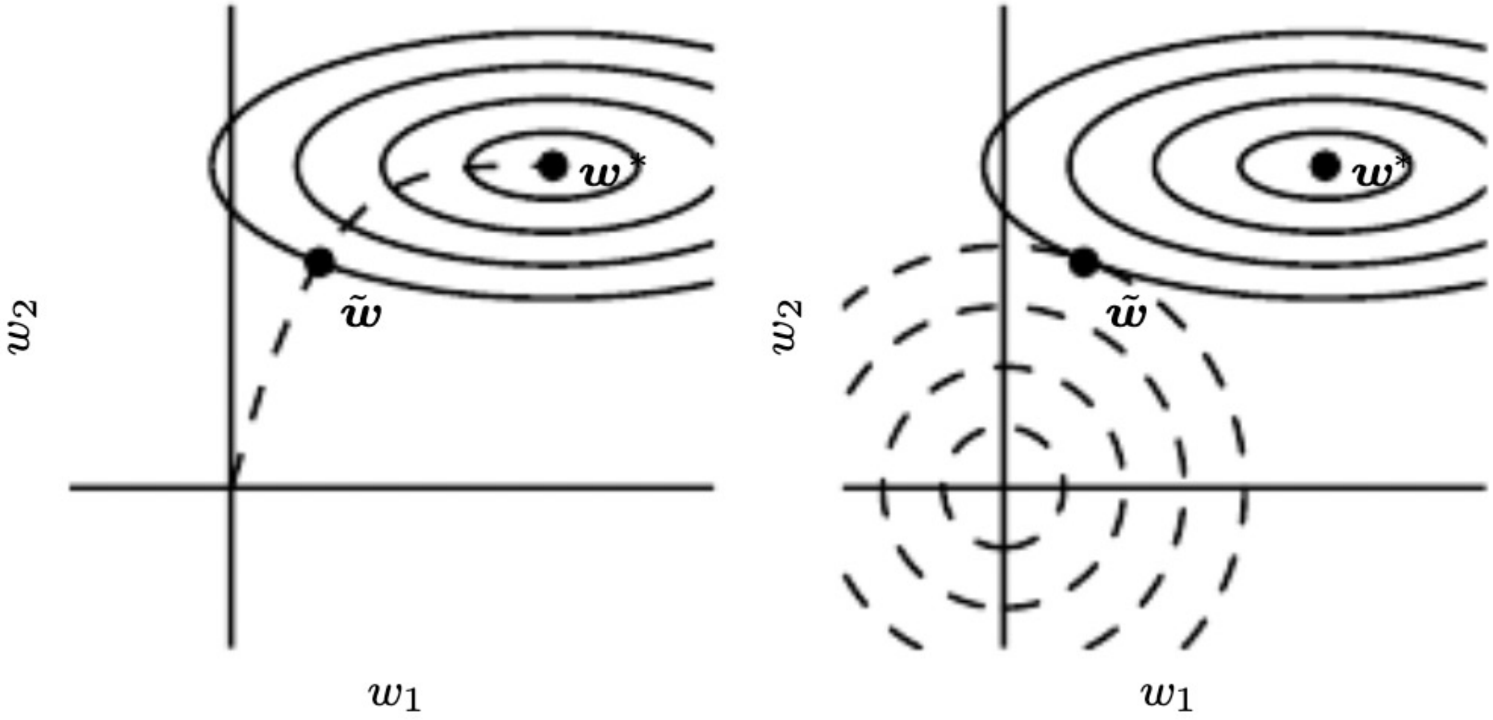


Image Source: https://chatbotslife.com/regularization-in-deep-learning-f649a45d6e0

Why Early Stopping Acts As a Regularizer

With parameters initialized around the origin, early stopping can behave like a parameter norm penalty (e.g., L2, without having a hyperparameter to tune); e.g.,



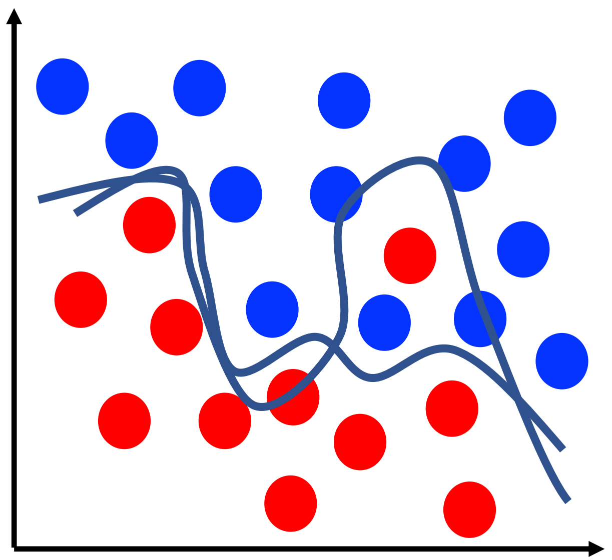
https://www.deeplearningbook.org/contents/regularization.html

Today's Topics

- Regularization
- Parameter norm penalty
- Early stopping
- Dataset augmentation
- Dropout
- Batch Normalization

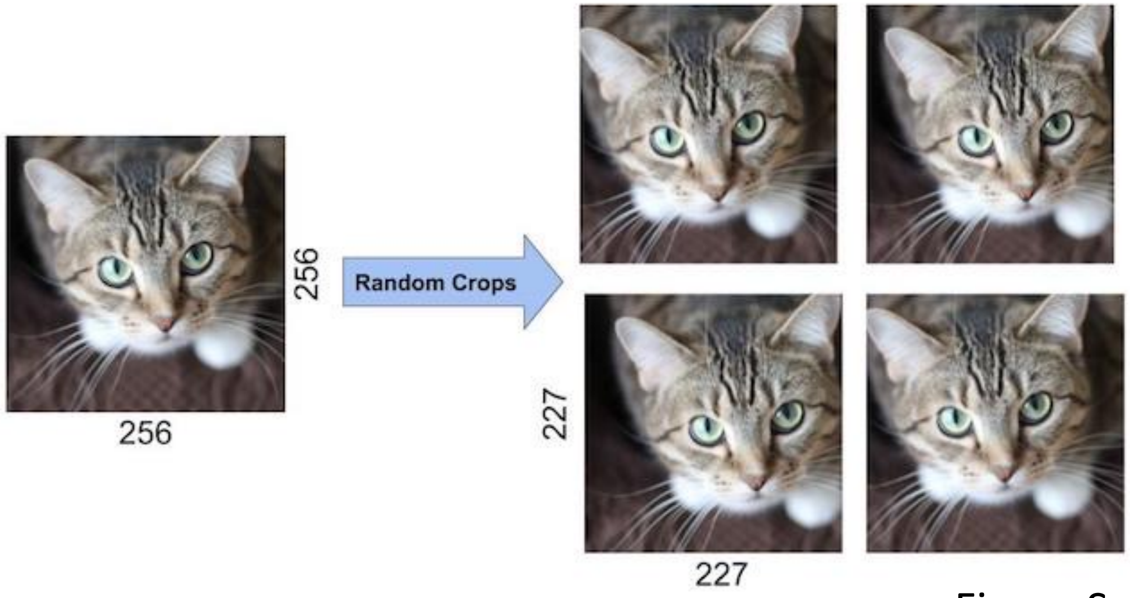
Recall: Overfitting Solution is to Add Data

Adding training data



AlexNet’s Data Augmentation Strategy

- Recall overfitting is risk for models with larger representational capacity, and AlexNet has 60 million parameters!
- Data augmentation strategy
    1. Random patches and their mirror images (2048x more data)
    2. Adjust RGB channels (using PCA to add multiples of principal components)



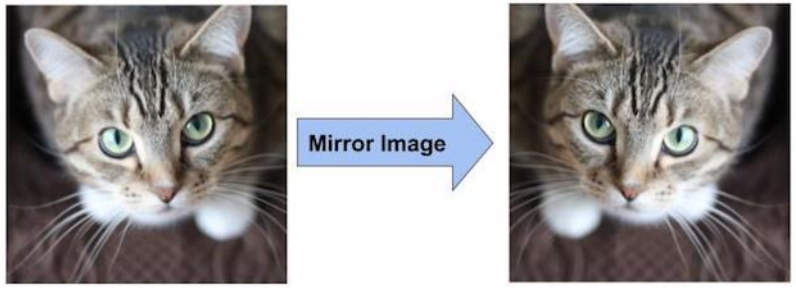
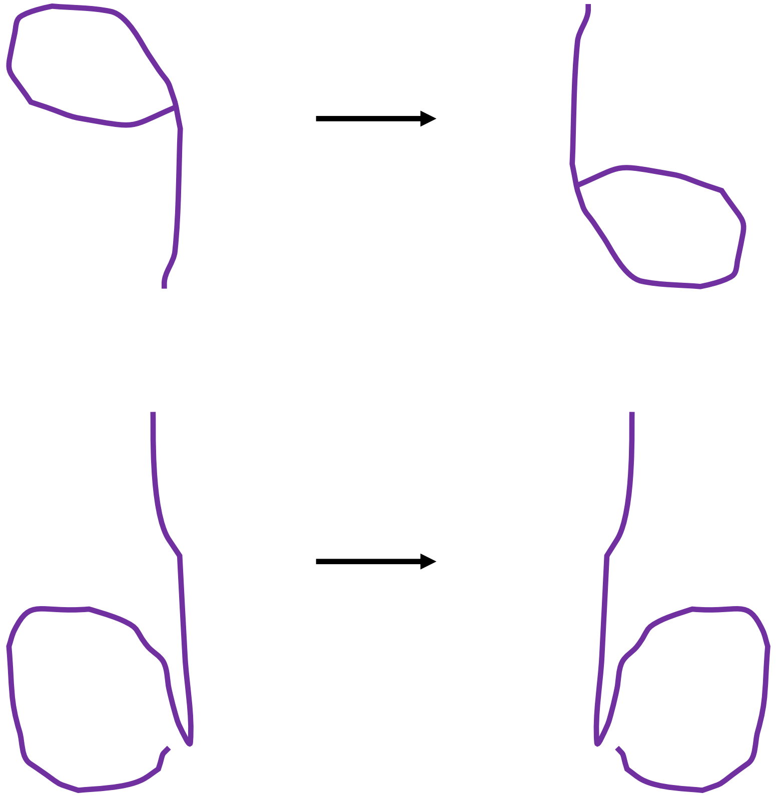


Figure Source: https://learnopencv.com/understanding-alexnet/

Caution: Match Augmentation Scheme to Data

- e.g., image mirroring and flipping could be poor choices for character recognition



Class Discussion

1. When/why are random patches a good/poor choice for data augmentation?

2. How else can you augment data for learning image classification models?

Today’s Topics

- Regularization
- Parameter norm penalty
- Early stopping
- Dataset augmentation
- Dropout
- Batch Normalization

Idea: Use Wisdom of the Crowds

More than 1: Ensemble

Why Choose Ensemble vs One Predictor?

- 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)

How to Produce an Ensemble? - Bagging

Bootstrap Aggregation (1994)

Train algorithm repeatedly on different random subsets of the training set

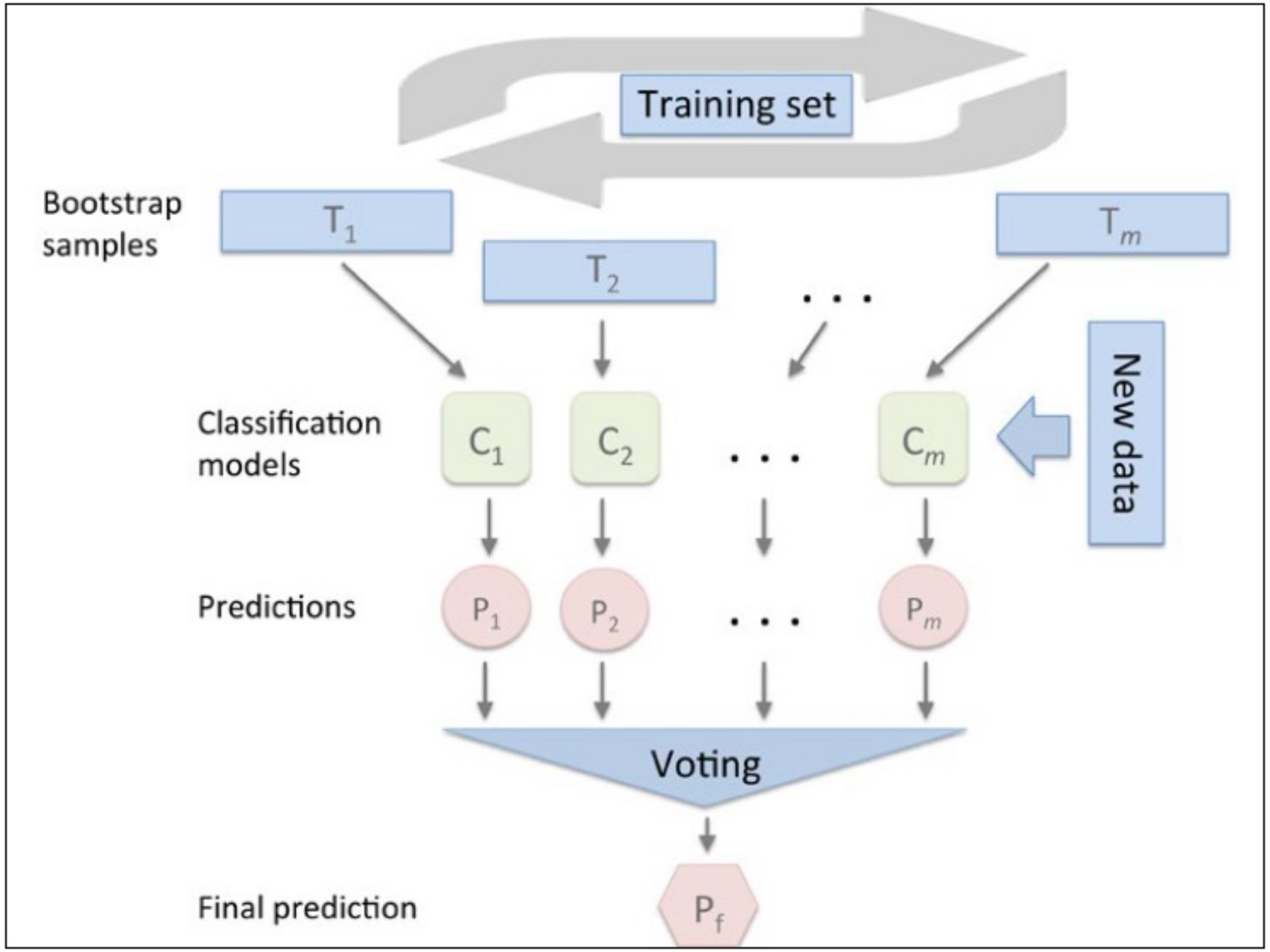


Figure Credit: Raschka & Mirjalili, Python Machine Learning.

How to Produce an Ensemble? - Bagging

- Build ensemble from “bootstrap samples” drawn with replacement
- e.g.,

Sample indices	Bagging round 1	Bagging round 2	...
1	2	7	...
2	2	3	...
3	1	2	...
4	3	1	...
5	7	1	...
6	2	7	...
7	4	7	...

C_1 C_2 C_m

Duplicate data can occur for training

Some examples missing from training data; e.g., round 1

Each classifier trained on different subset of data

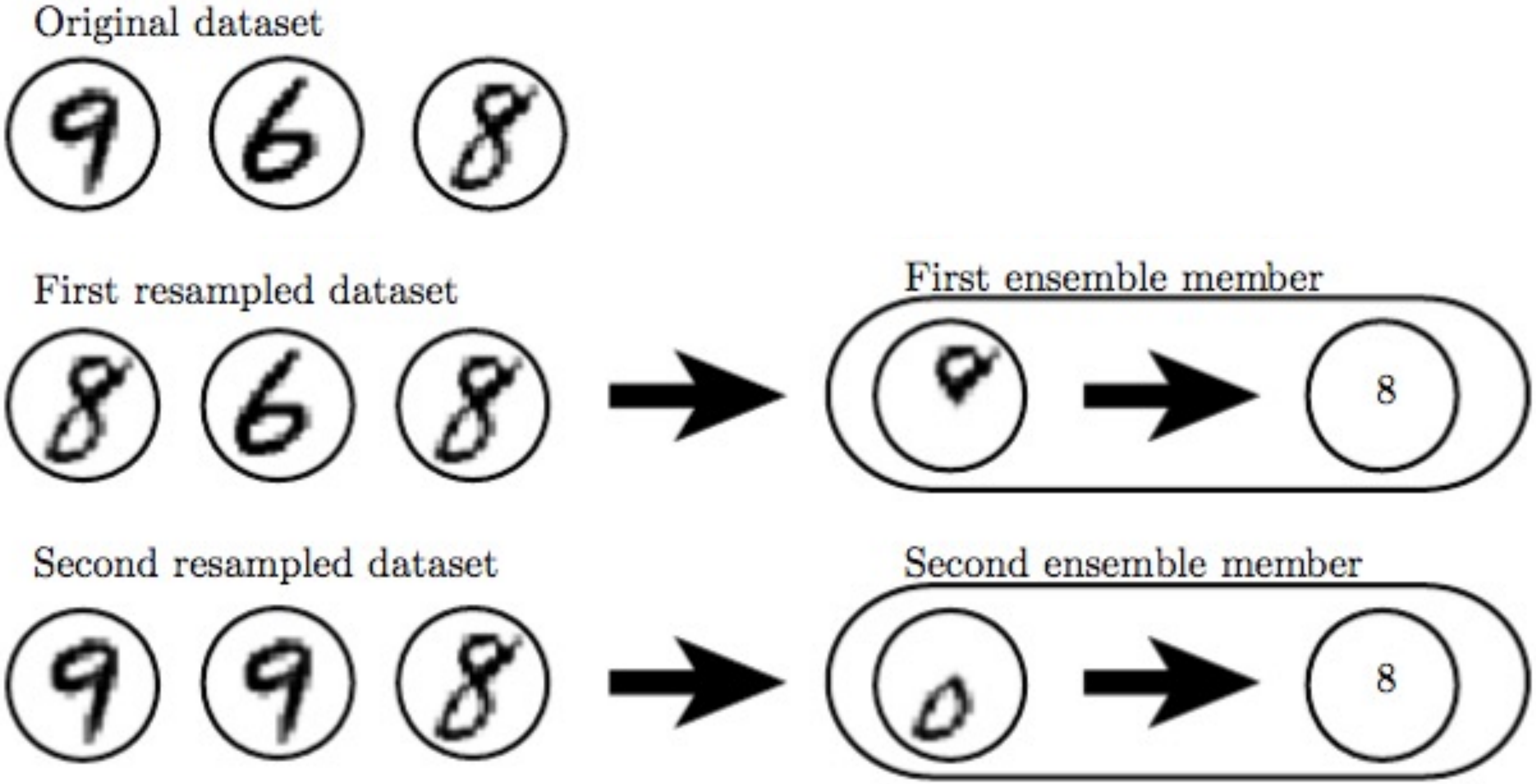
Predict using “hard” voting or averaging values for regression and “soft” voting

Breiman, Bagging Predictors, 1994.

Ho, Random Decision Forests, 1995.

Figure Credit: Raschka & Mirjalili, Python Machine Learning.

Intuition of Bagging (Train an 8 detector)



Goodfellow et al., Deep Learning (chapter 7), 2016.

Bagging Limitations

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

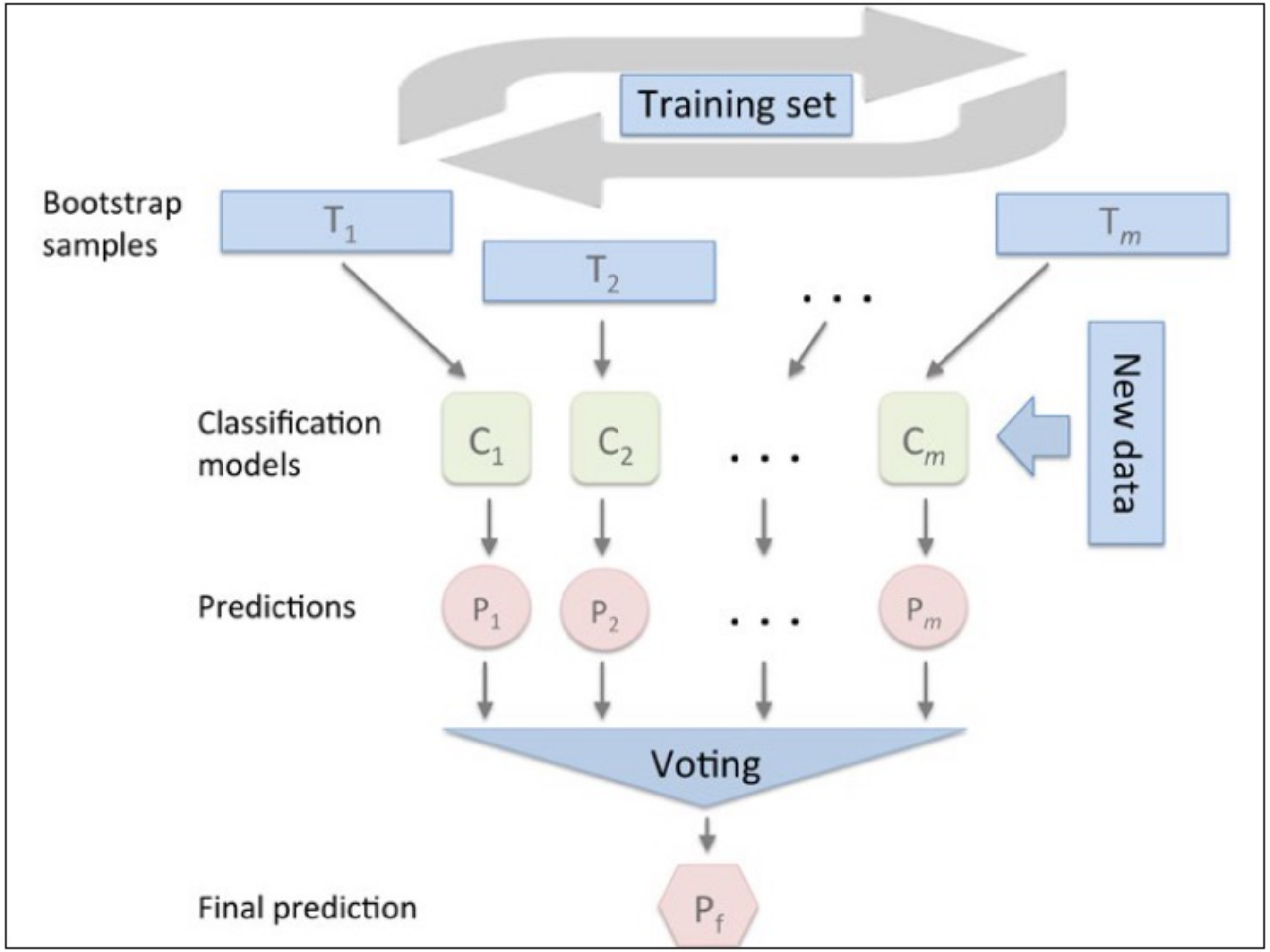
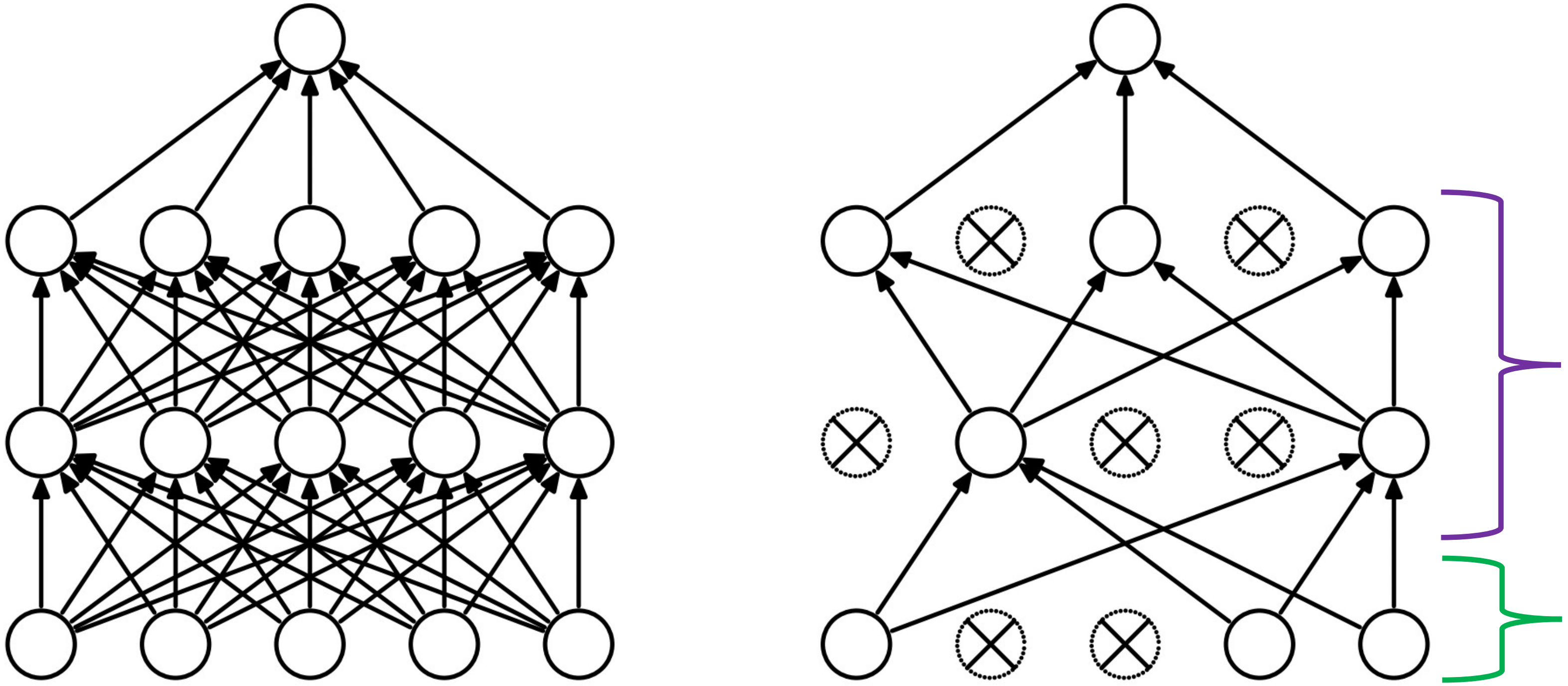


Figure Credit: Raschka & Mirjalili, Python Machine Learning.

How to Produce an Ensemble?

- Idea: approximate 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

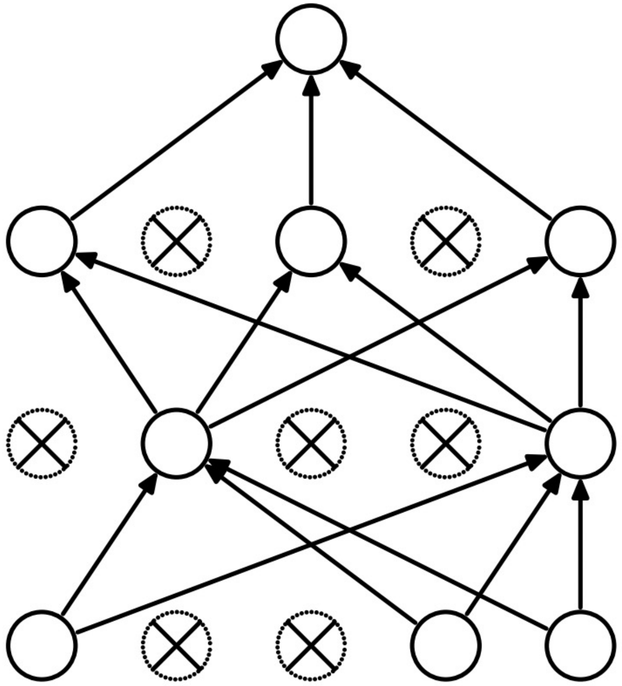
e.g., drop 20% of units in input layers

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

How to Produce an Ensemble?

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

For training, the only change from what has been discussed is that the forward pass and backpropagation run only through the sub-network.



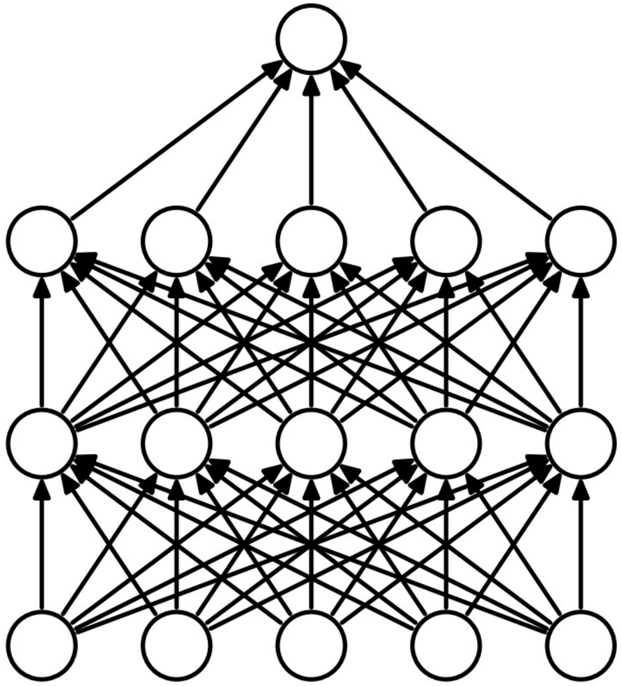
(b) After applying dropout.

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

How to Produce an Ensemble?

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

An ensemble is emulated at test time by applying the network without dropout. To reflect the network's expectation for a smaller amount of activation signal than observed at test time (e.g., input from 2 or 3 units instead of 5 units), each unit's outgoing weights should be multiplied by the probability it was dropped at training.

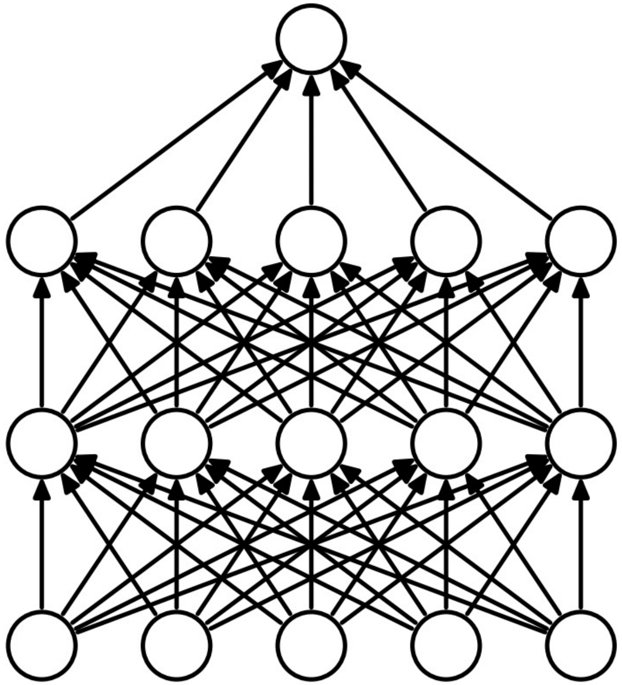


(b) After applying dropout.

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

Dropout vs Bagging

- Dropout approximates bagging with many models inexpensively
  - Trains algorithm repeatedly on different 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)

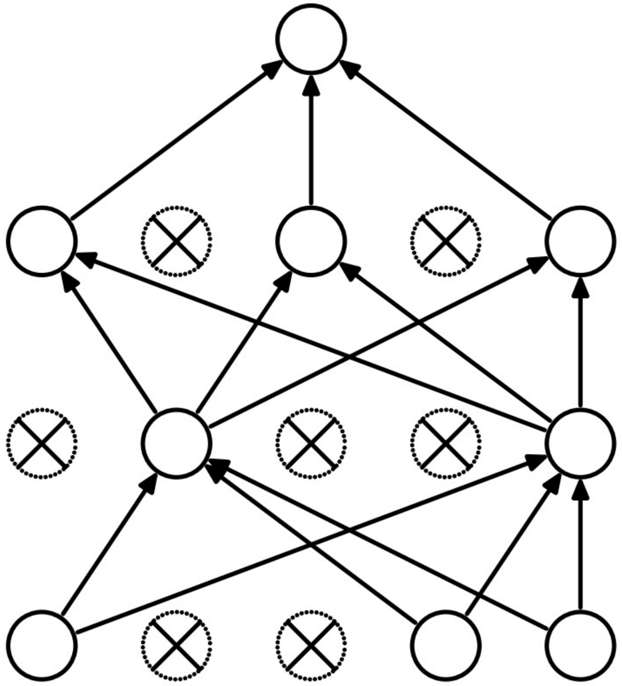


(b) After applying dropout.

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

Motivation for Dropout

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."



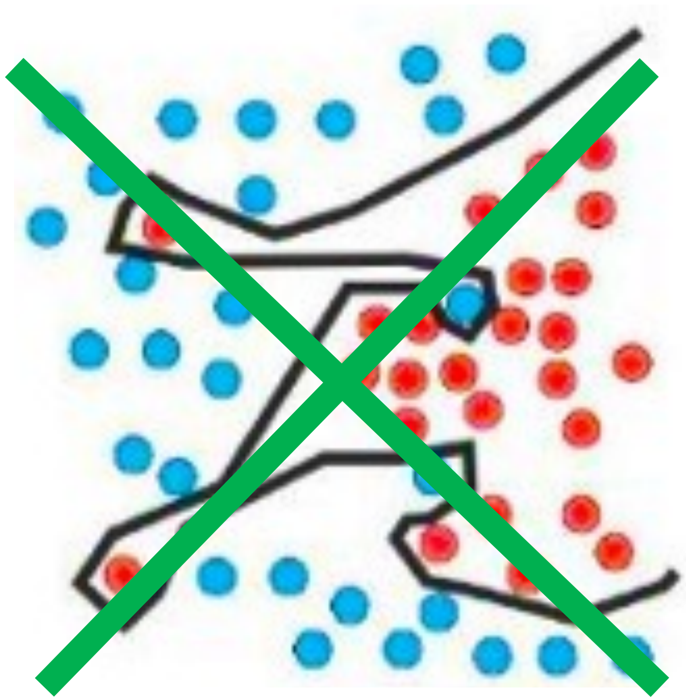
(b) After applying dropout.

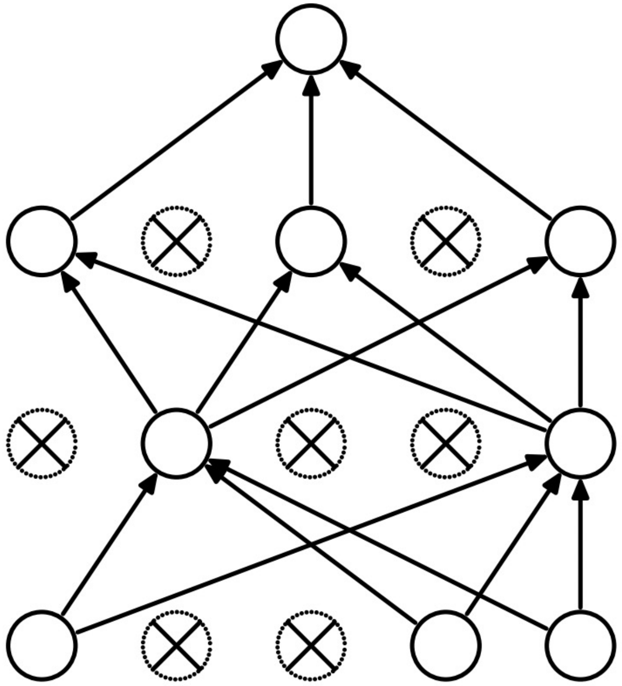
"Similarly, each hidden unit in a neural network trained with dropout must learn to work with a randomly chosen sample of other units. This should make each hidden unit more robust and drive it towards creating useful features on its own without relying on other hidden units to correct its mistakes."

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

Motivation for Dropout

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 the situation where large positive weights cancel similarly large negative weights, a sign of overfitting.



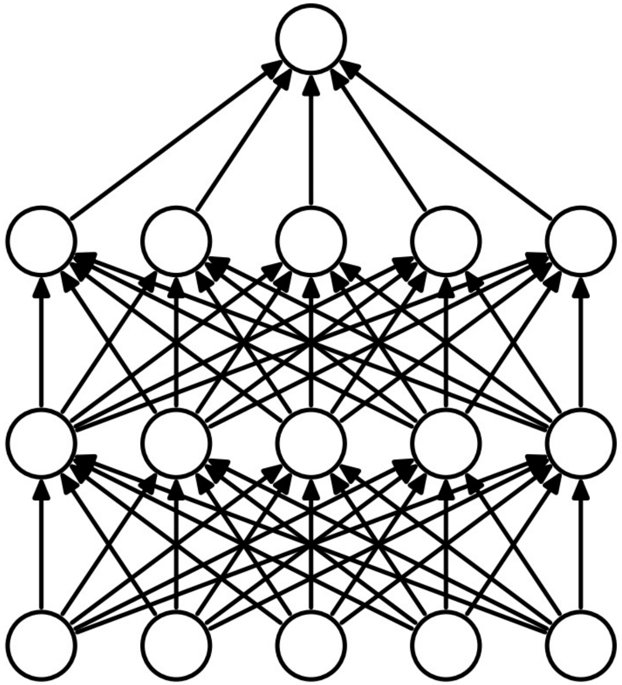


(b) After applying dropout.

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

How to Produce an Ensemble?

- A generalization of zeroing units out is to instead multiply units by noise, and this is a common approach for convolutional layers



(b) After applying dropout.

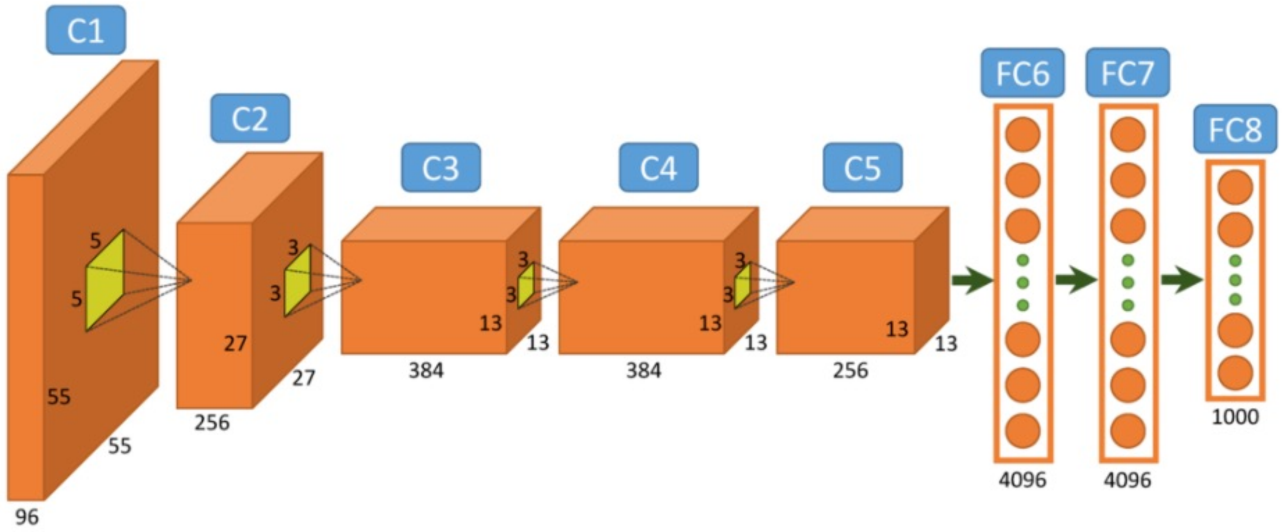
Relevant articles:
*https://towardsdatascience.com/dropout-on-convolutional-layers-is-weird-5c6ab14f19b2
*Wu and Gu. “Towards dropout training for convolutional neural networks.” Neural Networks, 2015.

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

Dropout for CNNs

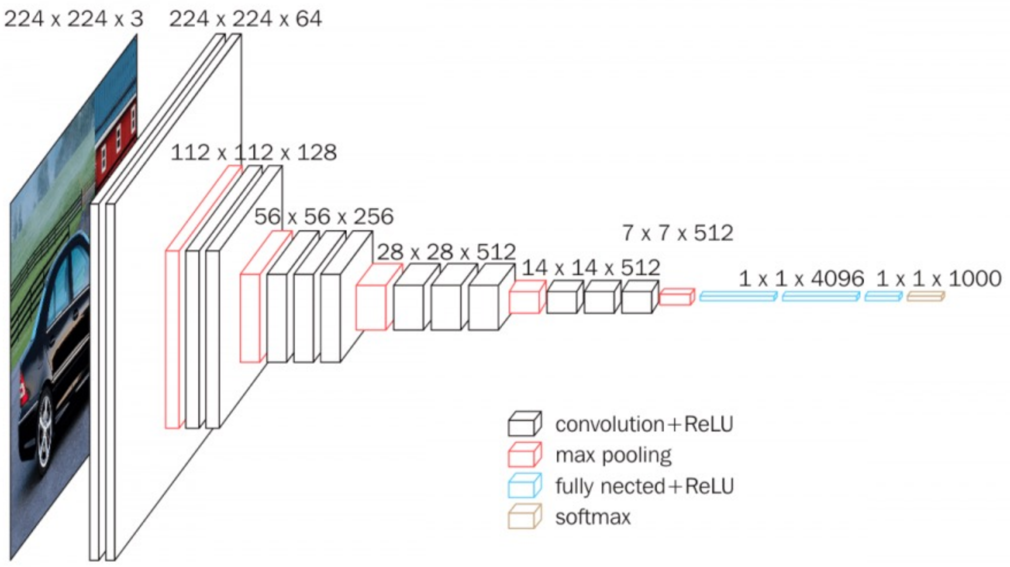
Why do you think dropout is typically not used in convolutional layers? e.g., for the image classification algorithms we discussed, it is used only in fully connected layers:

AlexNet (2012)



https://www.researchgate.net/figure/Architecture-of-Alexnet-From-left-to-right-input-to-output-five-convolutional-layers_fig2_312303454

VGG (2014)

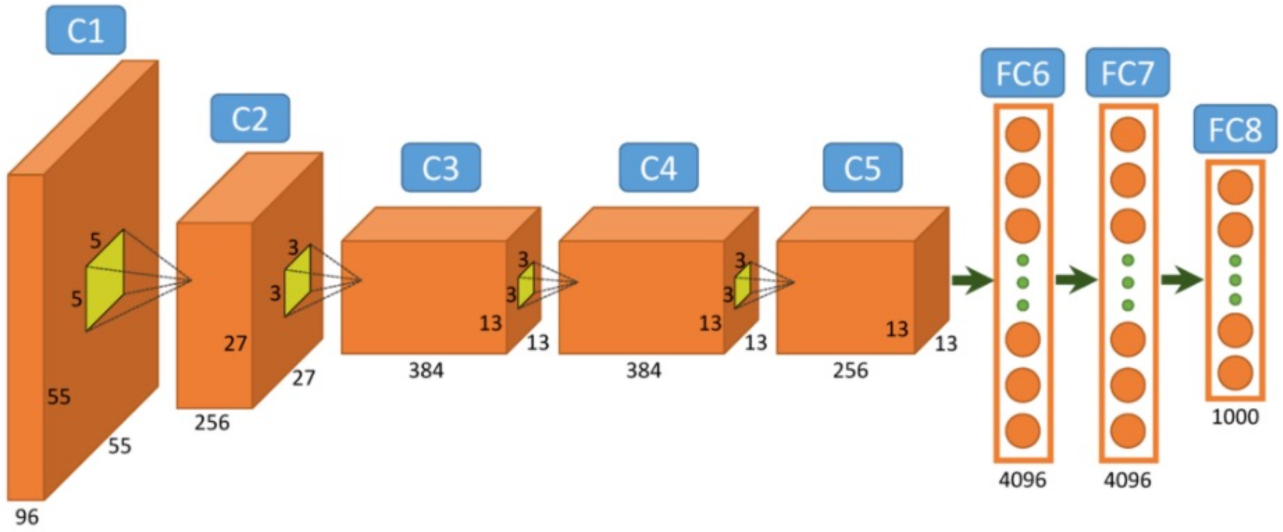


https://neurohive.io/en/popular-networks/vgg16/

Dropout for CNNs

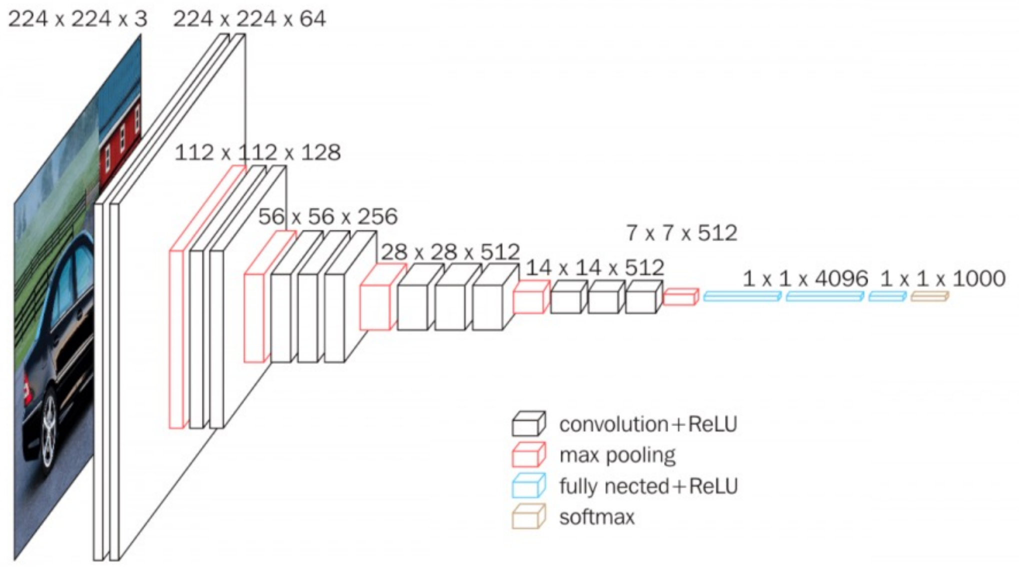
Parameter tying reduces parameter count and so already offers strong regularization

AlexNet (2012)



https://www.researchgate.net/figure/Architecture-of-Alexnet-From-left-to-right-input-to-output-five-convolutional-layers_fig2_312303454

VGG (2014)



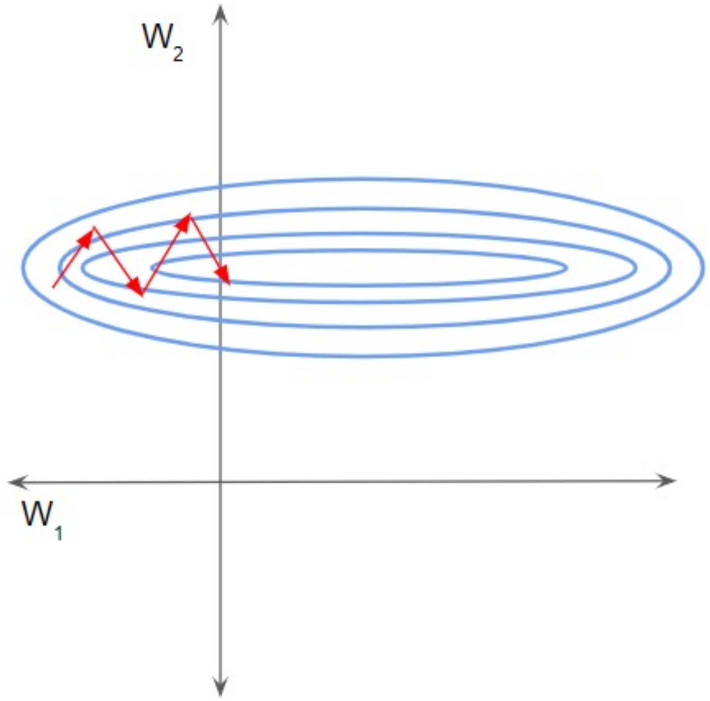
https://neurohive.io/en/popular-networks/vgg16/

Today’s Topics

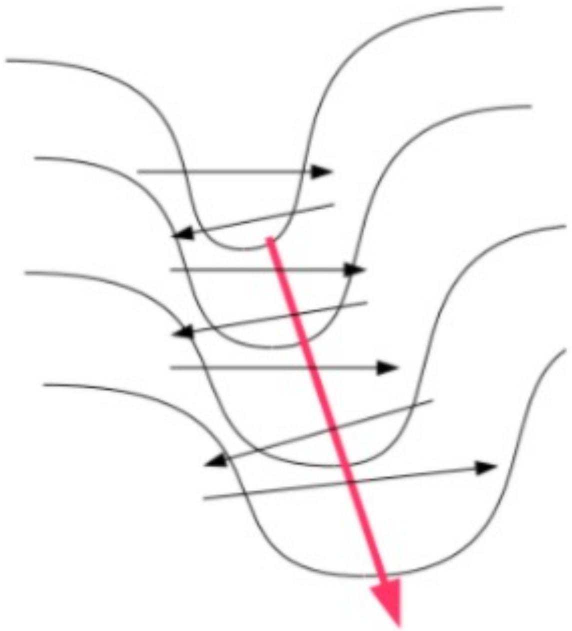
- Regularization
- Parameter norm penalty
- Early stopping
- Dataset augmentation
- Dropout
- Batch Normalization

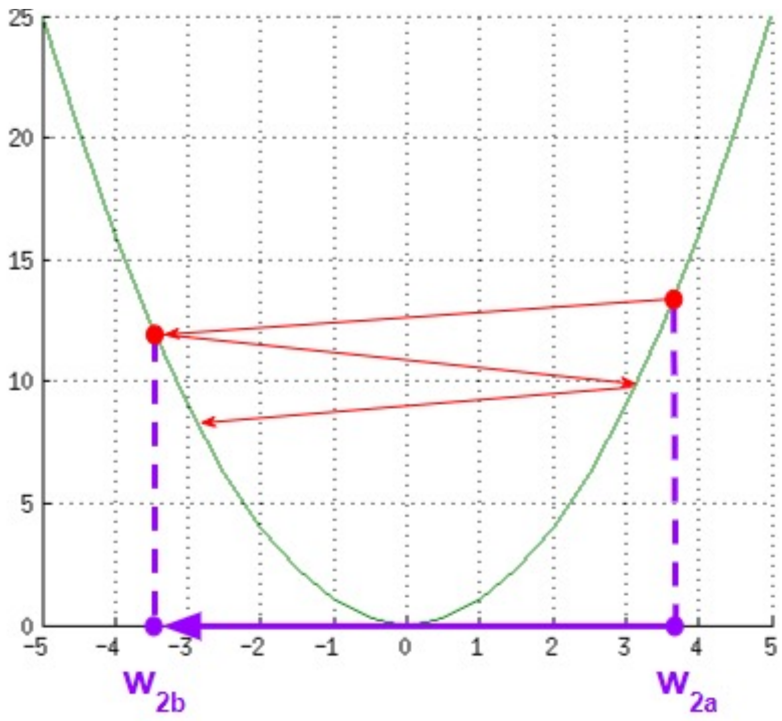
Motivation: Features On Different Scales Can Cause Learning To Be Slower and Poor Performance

e.g., 2D loss function:



Inefficient bouncing can occur during learning when larger updates are needed for some weights to minimize the loss during gradient descent



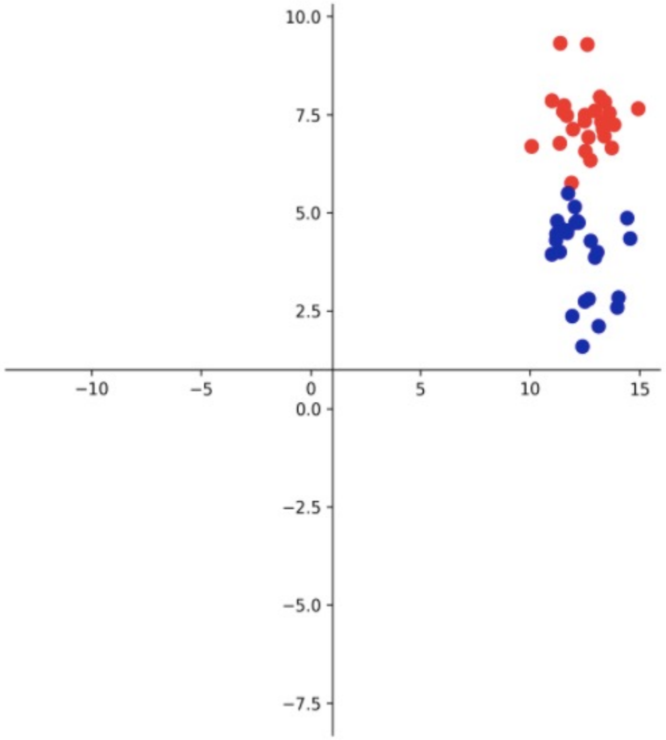


https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

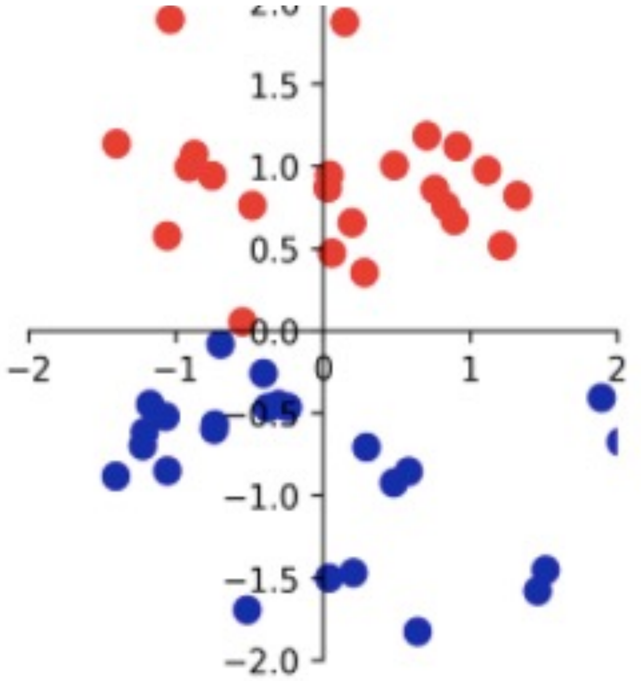
Recall: Basic Data Initialization Approach

* Simplify learning by standardizing input data so mean is 0 and standard deviation 1

Original data:



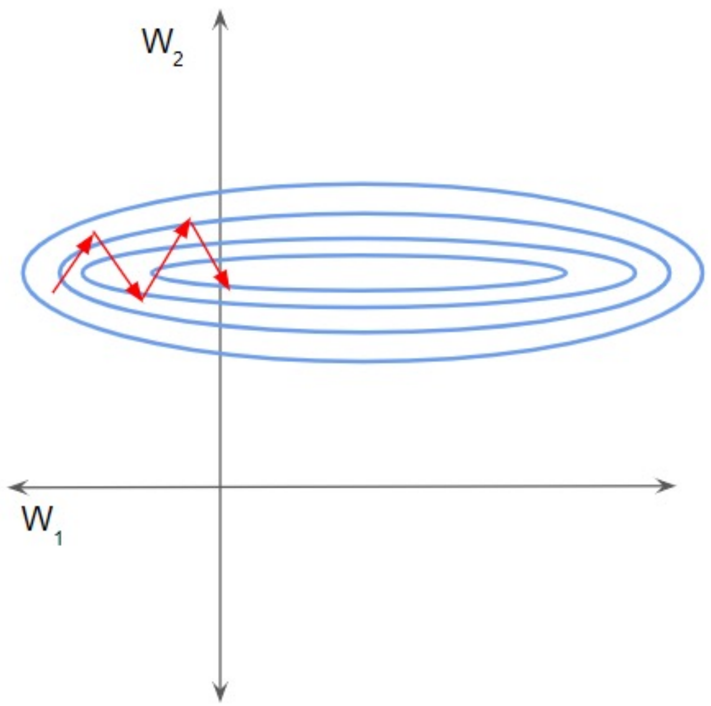
Standardized data:



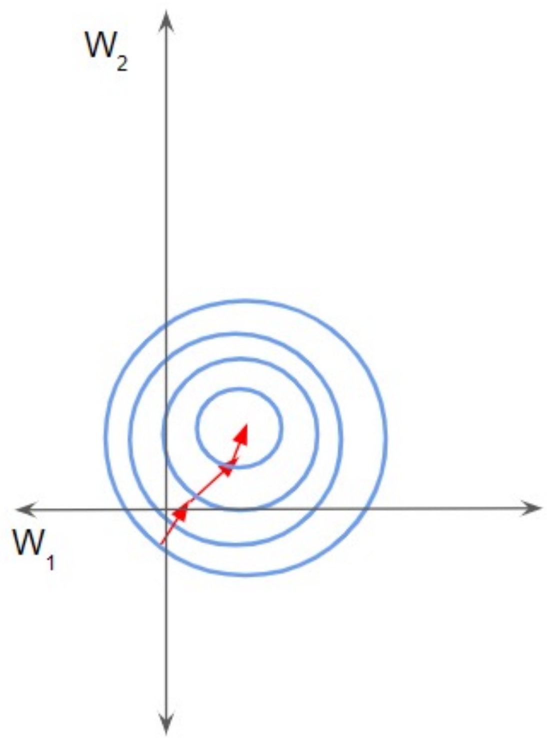
https://github.com/amueller/introduction_to_ml_with_python/blob/master/03-unsupervised-learning.ipynb

Recall: Basic Data Initialization Approach

* Simplify learning by standardizing input data so mean is 0 and standard deviation 1

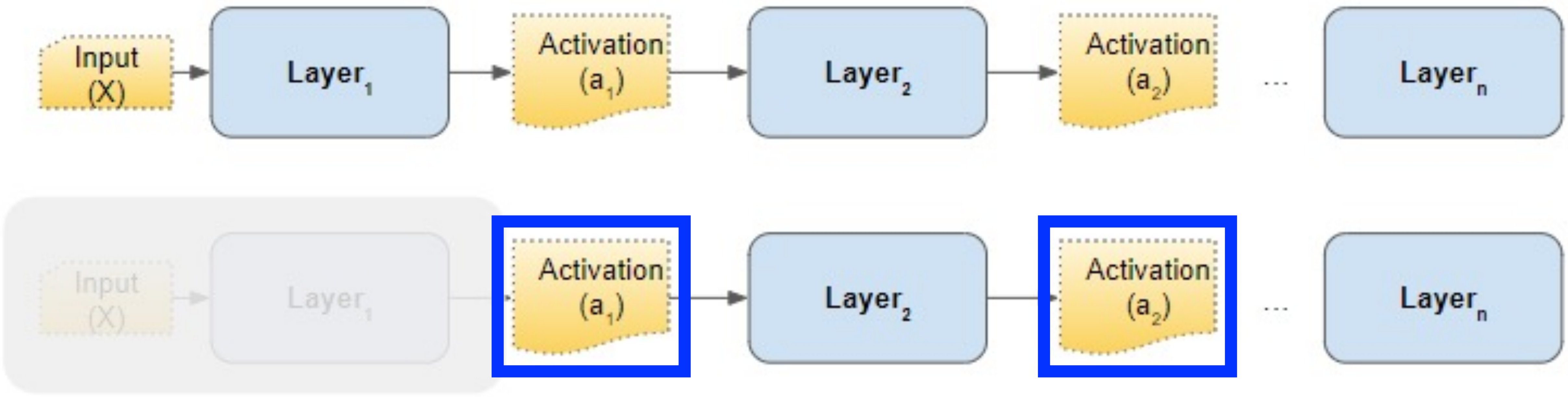


Standardization changes the loss function so gradient descent can more smoothly arrive at the minimum!



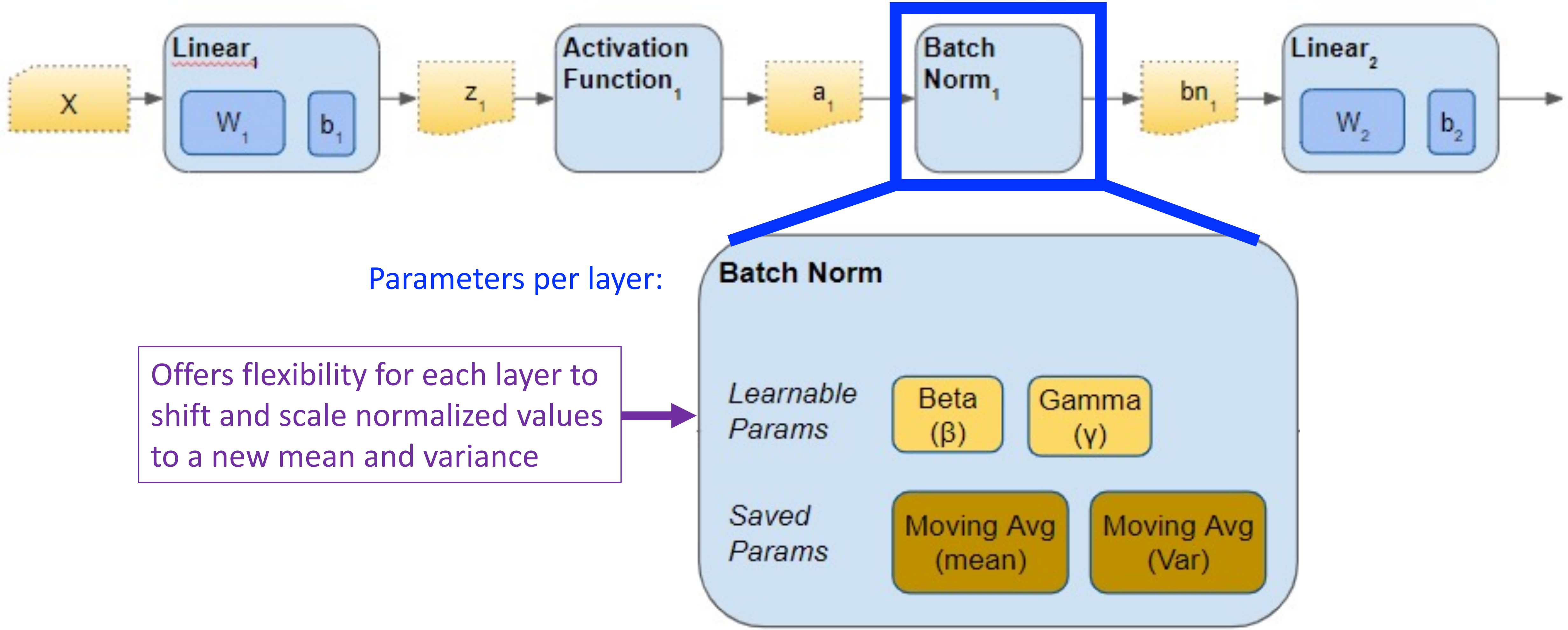
https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

Idea: Further Simplify Learning by Transforming Input to Hidden Layer(s)



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

Batch Normalization Layer



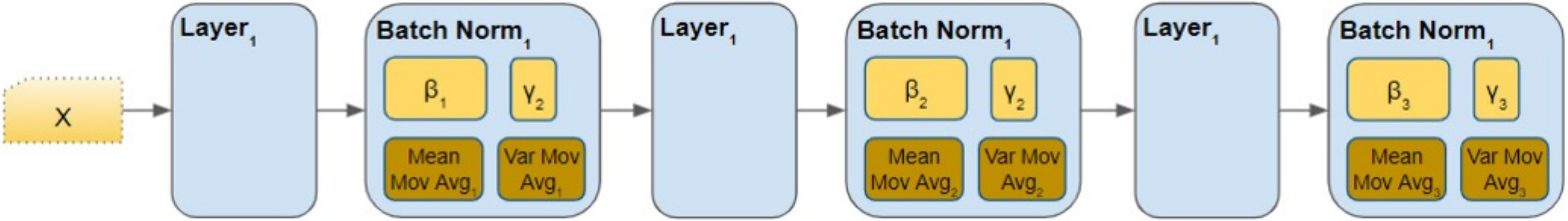
Parameters per layer:

Offers flexibility for each layer to shift and scale normalized values to a new mean and variance

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

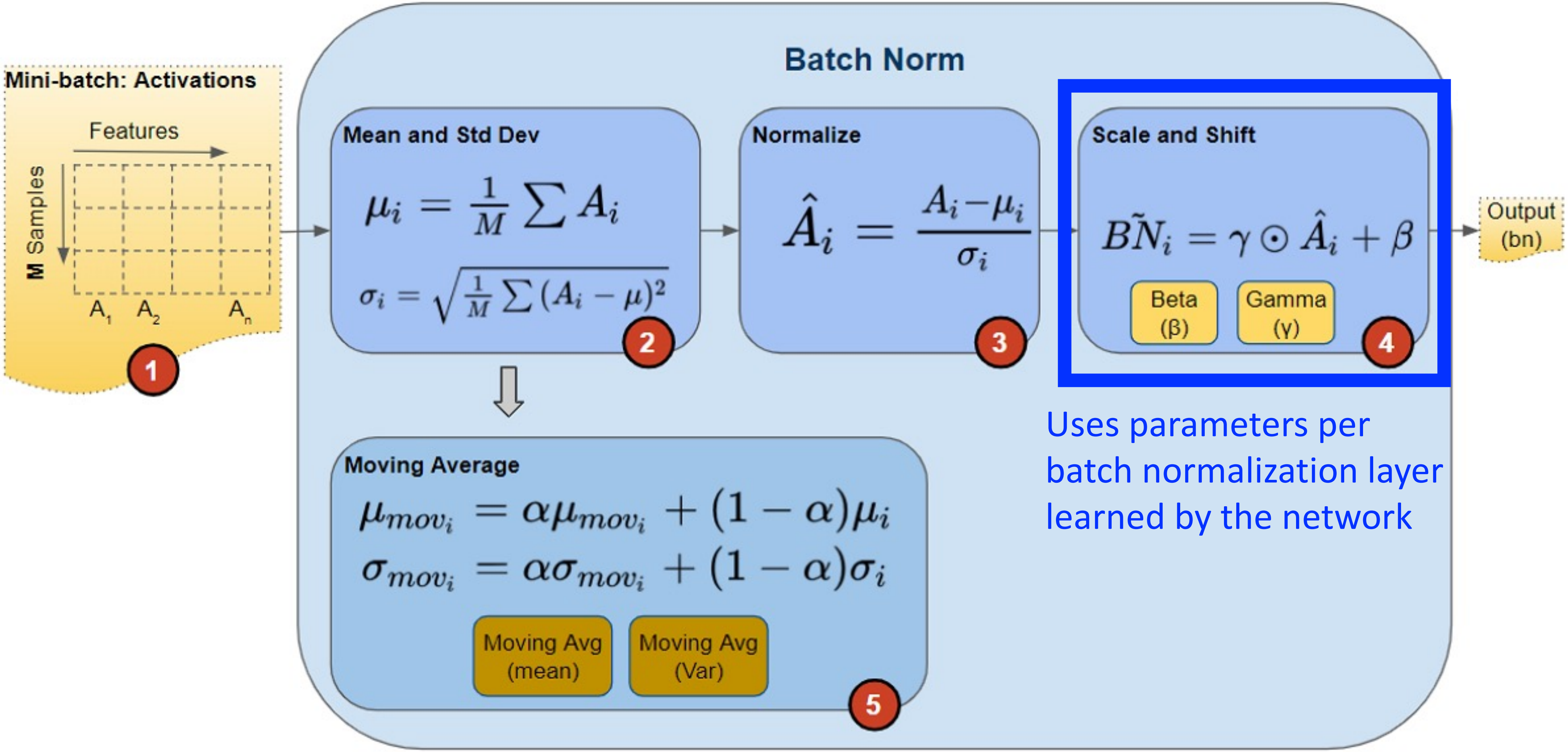
Batch Normalization Layer

How many trainable parameters must be learned during training for this subnetwork?



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

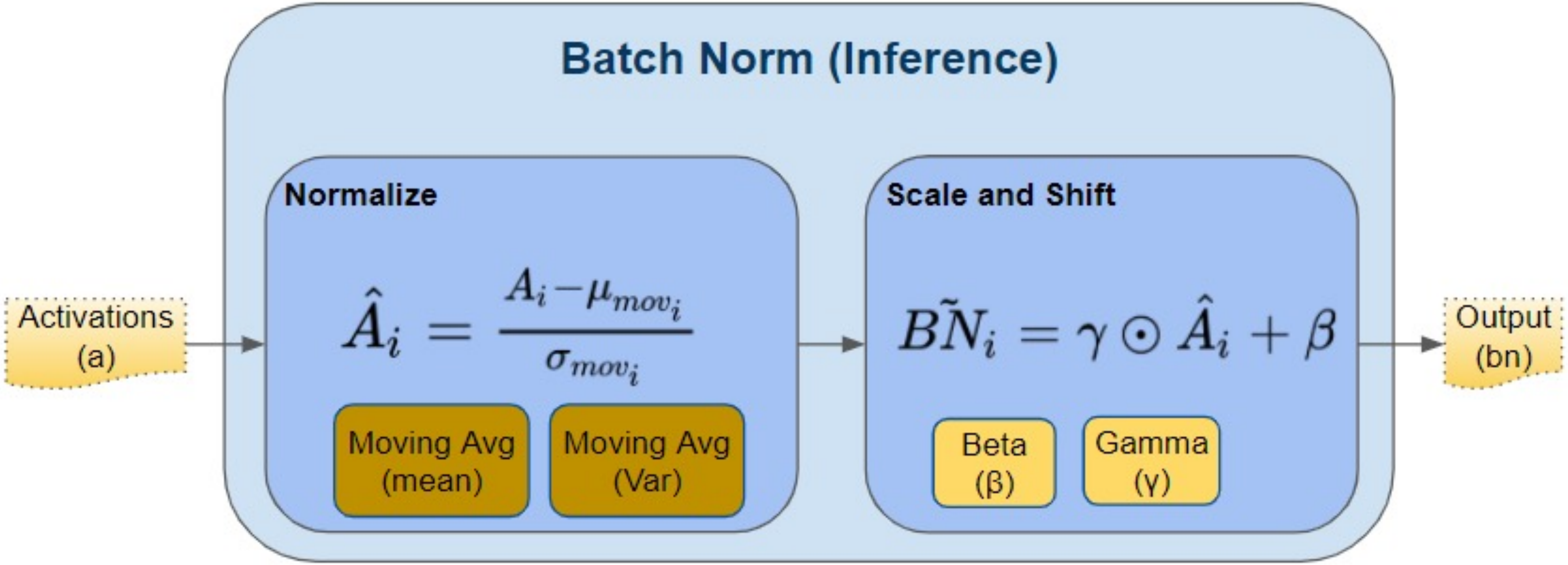
Batch Normalization: Training Operation



Uses parameters per batch normalization layer learned by the network

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

Batch Normalization: Test-Time Operation



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

Benefits and Limitations

- Pros - smooths the optimization function leading to:
  - Faster training convergence
  - More stable learning when paired with different hyperparameters and initializations
  - Better generalization performance

- Cons - extra layer(s) introduce more training and testing time

Santurkar et al. How Does Batch Normalization Help Optimization? Neurips 2018.

Today’s Topics

- Regularization
- Parameter norm penalty
- Early stopping
- Dataset augmentation
- Dropout
- Batch Normalization


The End