Digital Image Processing ECE 566

Ahmad Ghasemi

Department of Electrical and Computer Engineering

University of Massachusetts Amherst

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

Batch Normalization

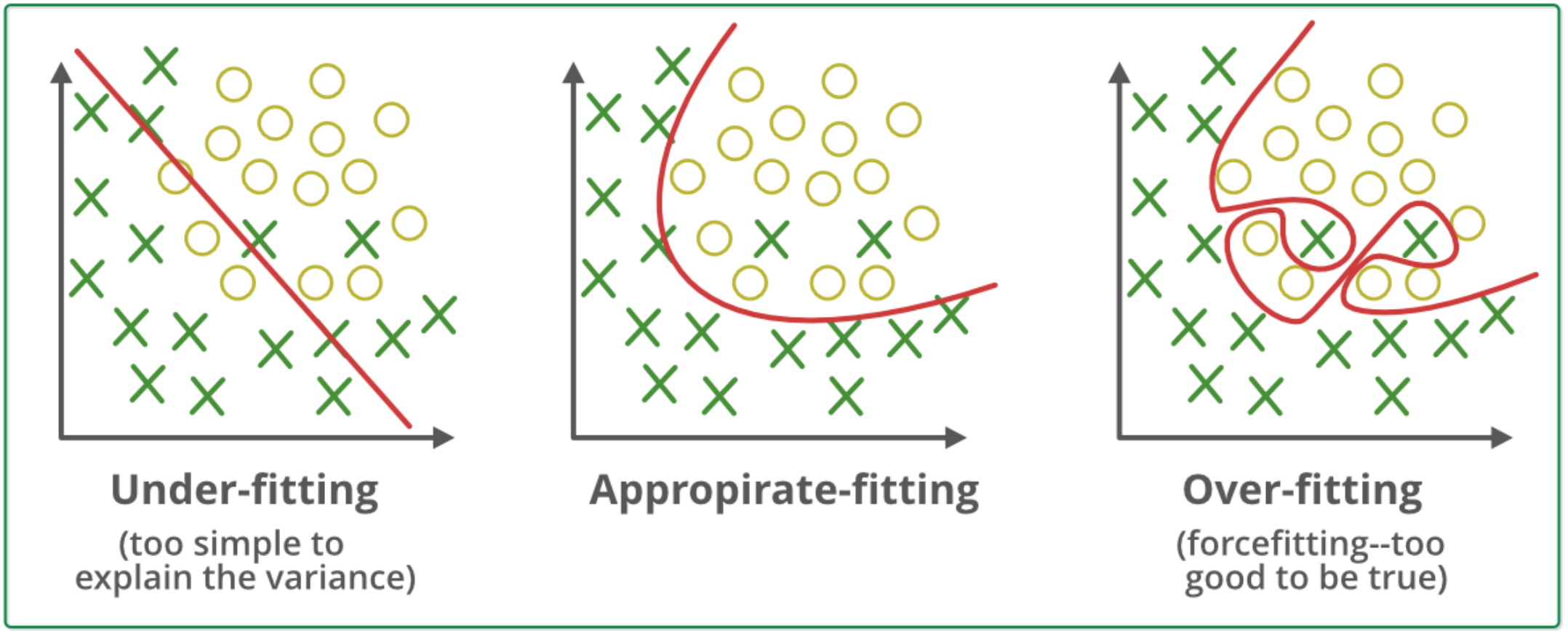
```
# example of batch normalization for an cnn
from keras.layers import Dense
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import BatchNormalization
...
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D())
model.add(Dense(1))
...
```

Overfitting and Underfitting

https://www.flickr.com/photos/iancarroll/5058330466/

University of Massachusetts Amherst

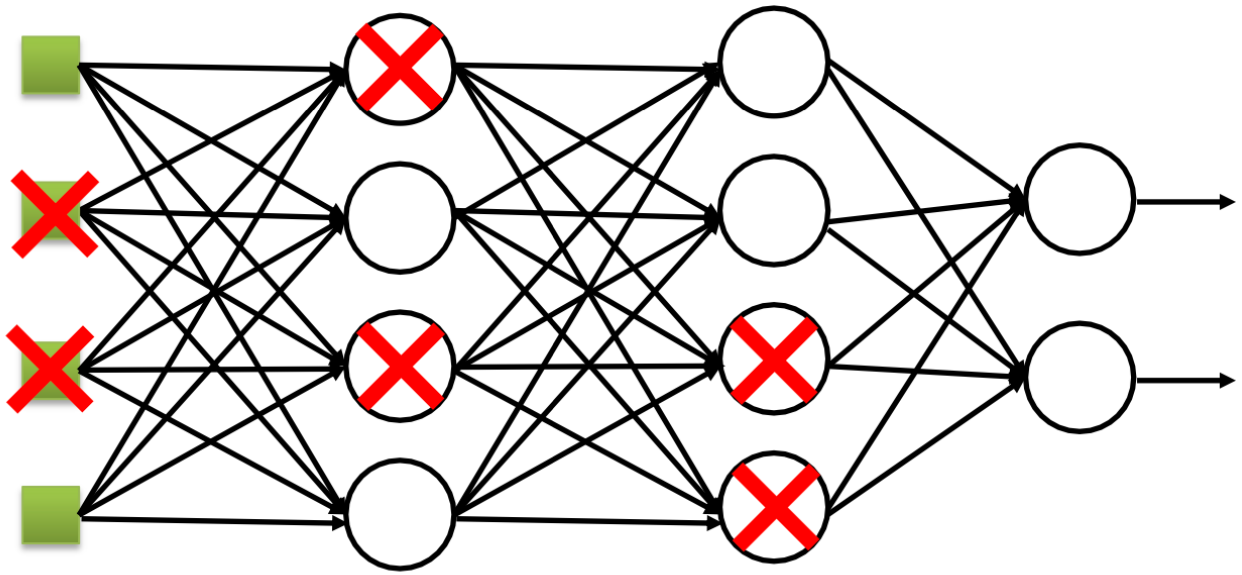
Overfitting and Underfitting



https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

Dropout

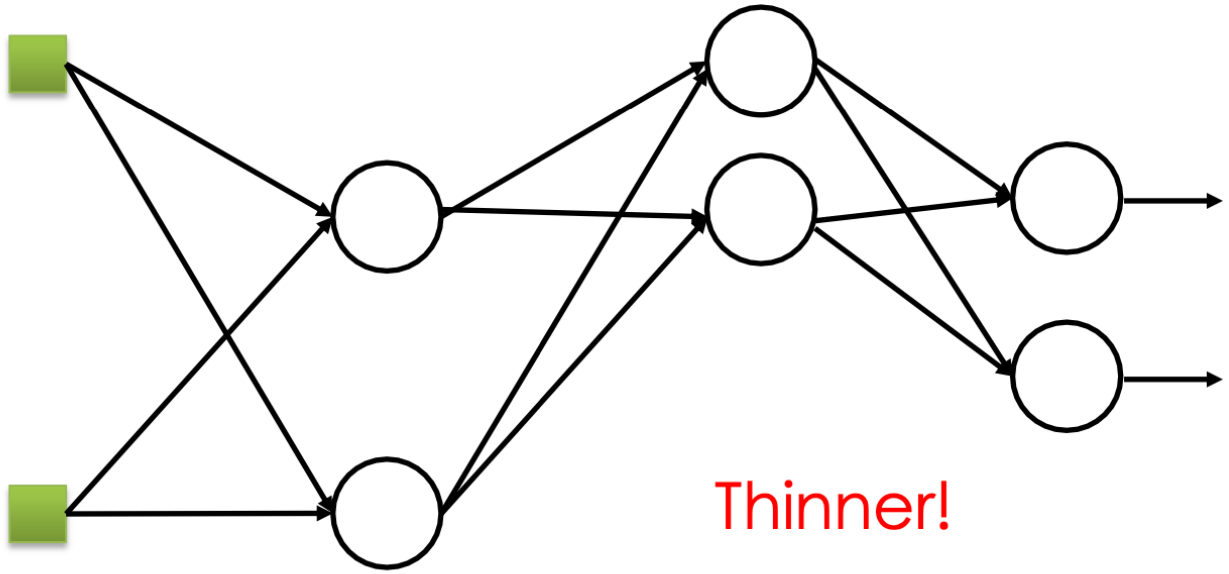
Training:



- **Each time before updating the parameters**
  - Each neuron has p% to dropout

Dropout

Training:

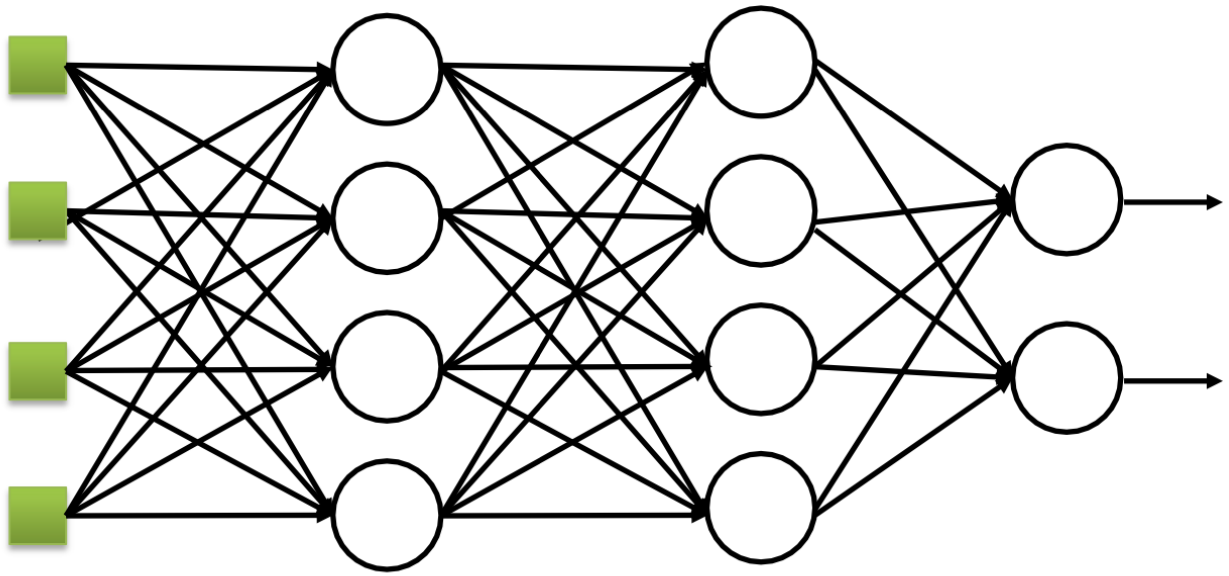


- **Each time before updating the parameters**
  - Each neuron has p% to dropout
  - Using the new network for training

The structure of the network is changed.

Dropout

Testing:

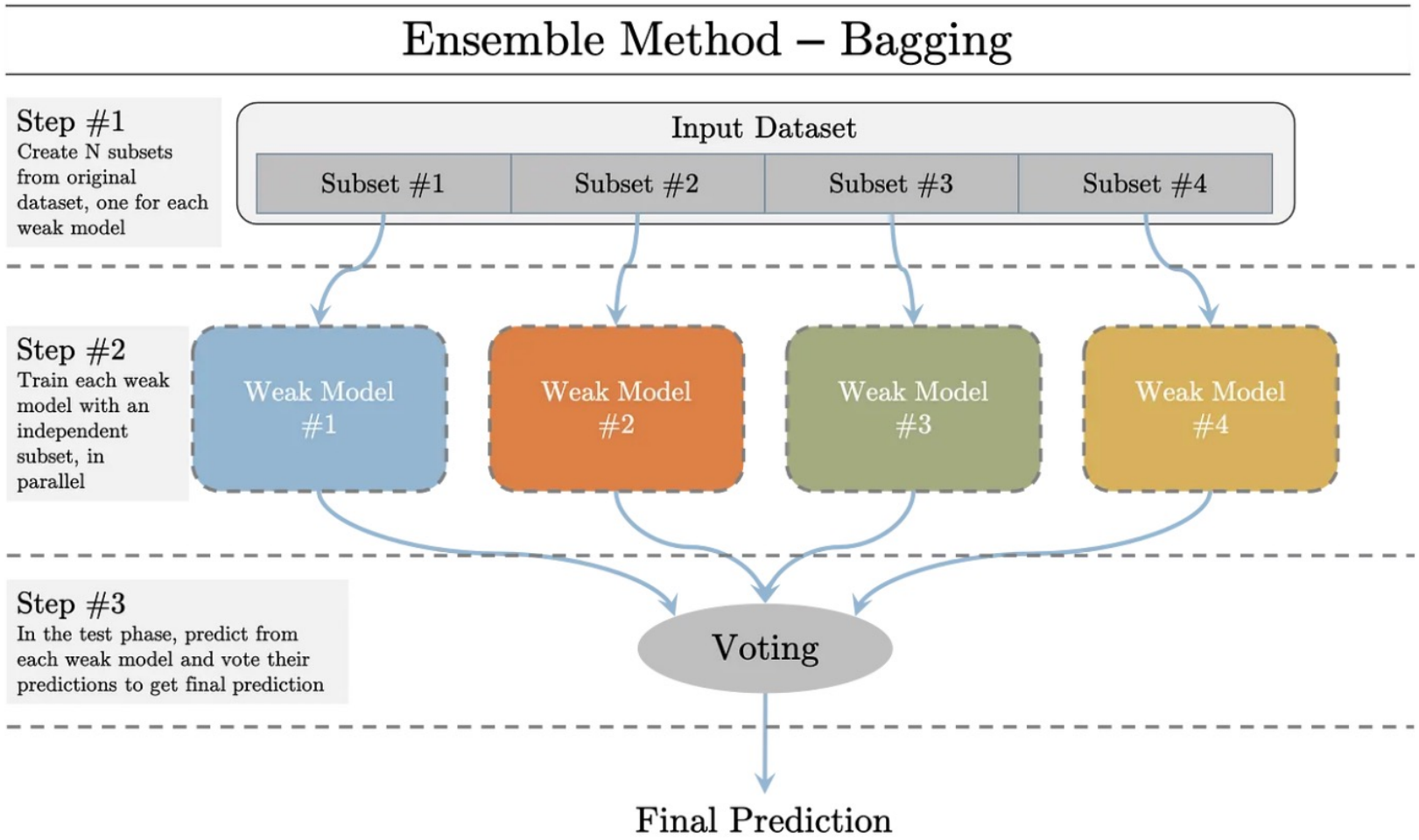


- No dropout

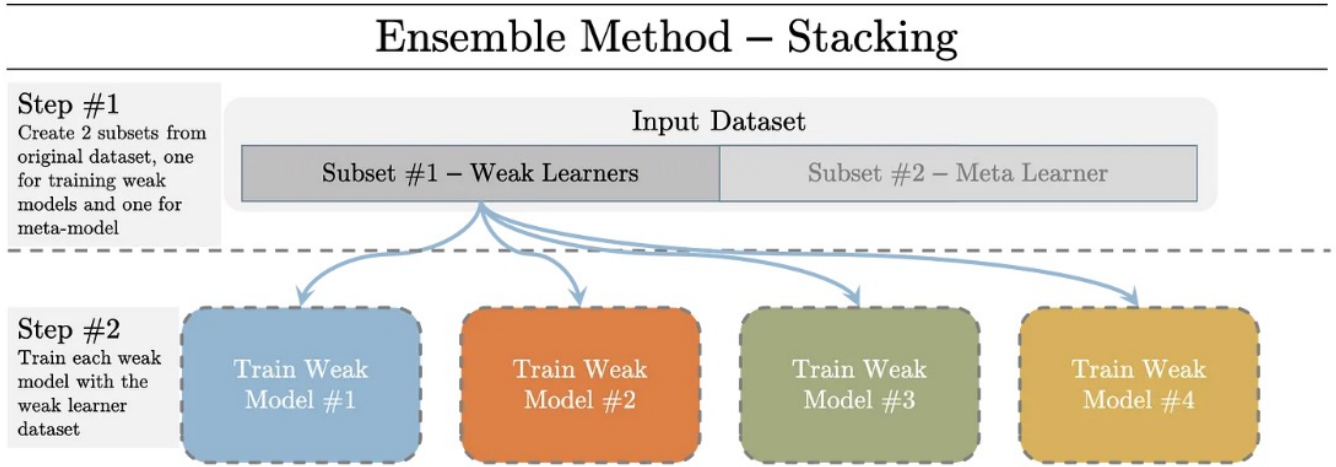
Ensemble Methods

University of Massachusetts Amherst

Ensemble Methods: Bagging



https://towardsdatascience.com/what-are-ensemble-methods-in-machine-learning-cac1d17ed349

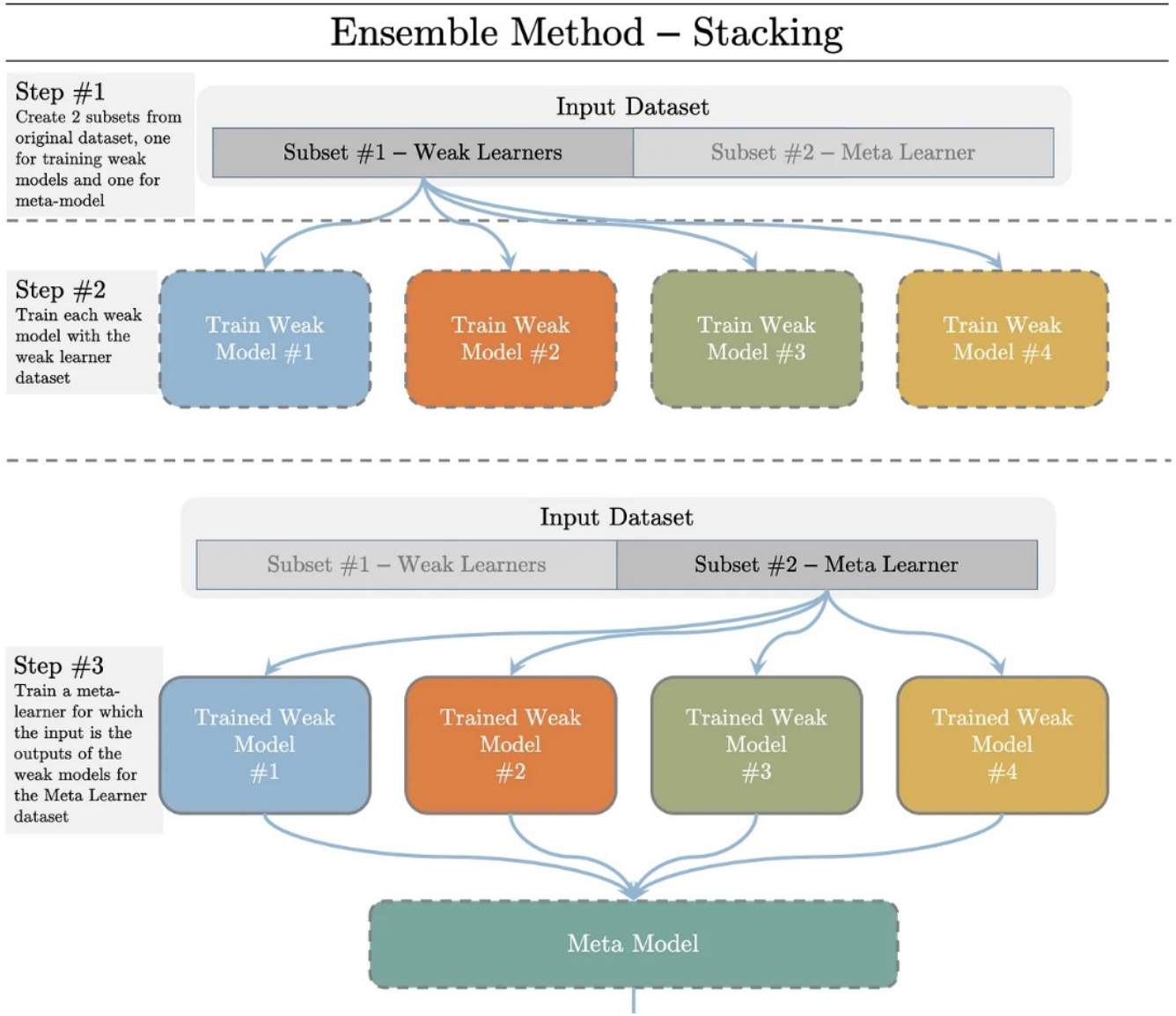


Ensemble Methods:

Stacking

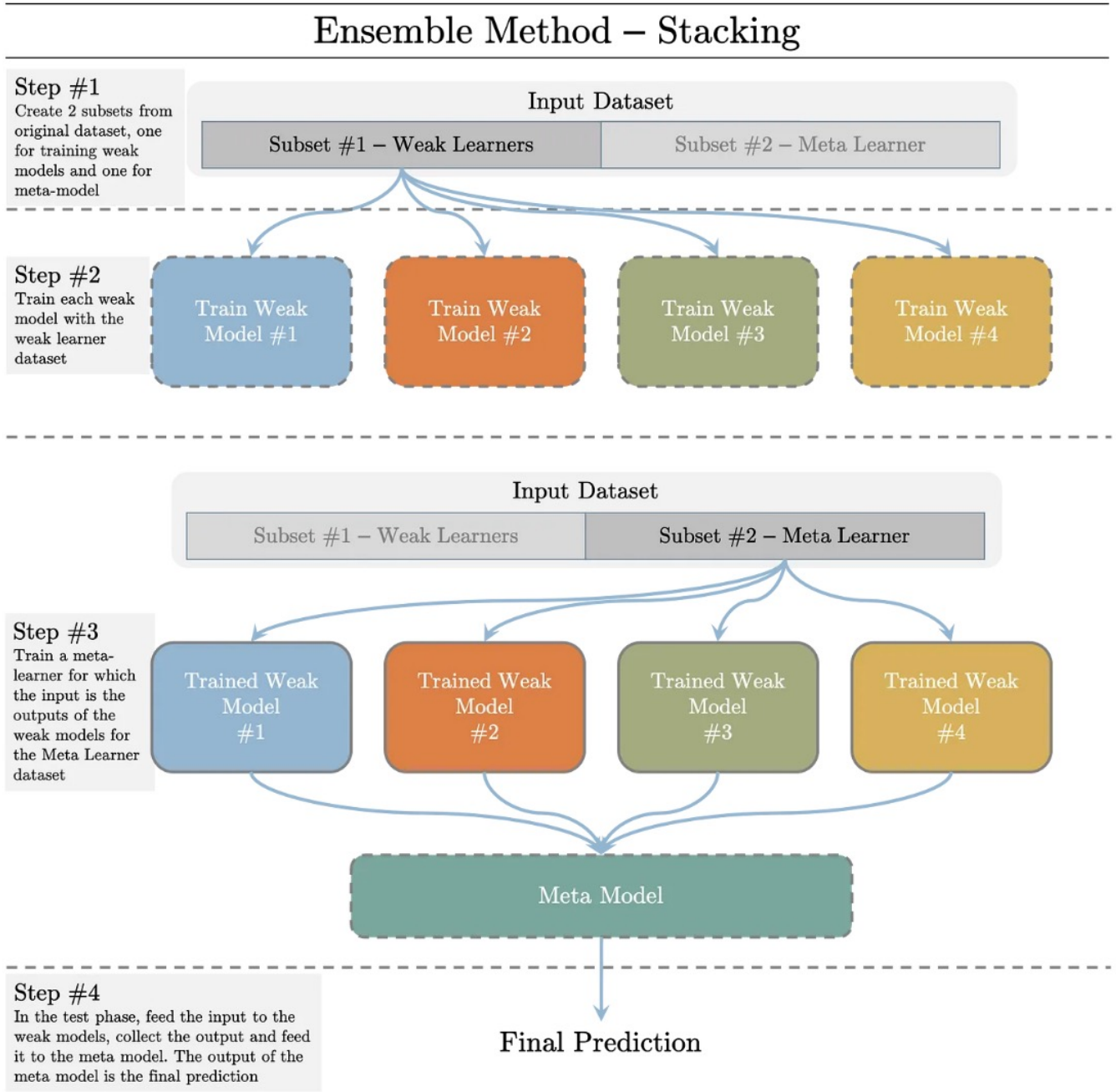
https://towardsdatascience.com/what-are-ensemble-methods-in-machine-learning-cac1d17ed349

Ensemble Methods: Stacking

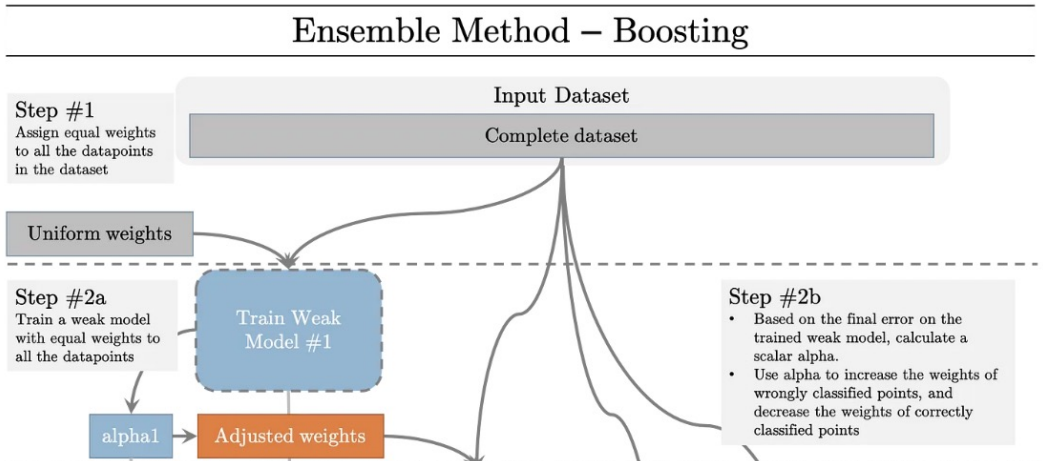


https://towardsdatascience.com/what-are-ensemble-methods-in-machine-learning-cac1d17ed349

Ensemble Methods: Stacking



https://towardsdatascience.com/what-are-ensemble-methods-in-machine-learning-cac1d17ed349



Ensemble Methods:

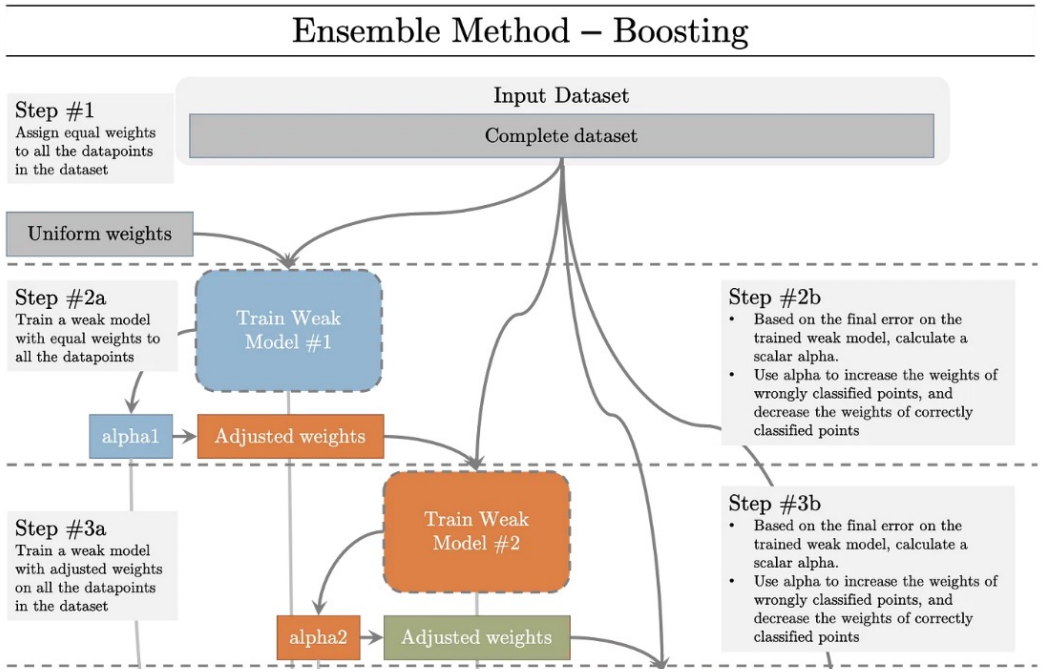
Boosting

https://towardsdatascience.com/what-are-ensemble-methods-in-machine-learning-cac1d17ed349

University of Massachusetts Amherst

Ensemble Methods:

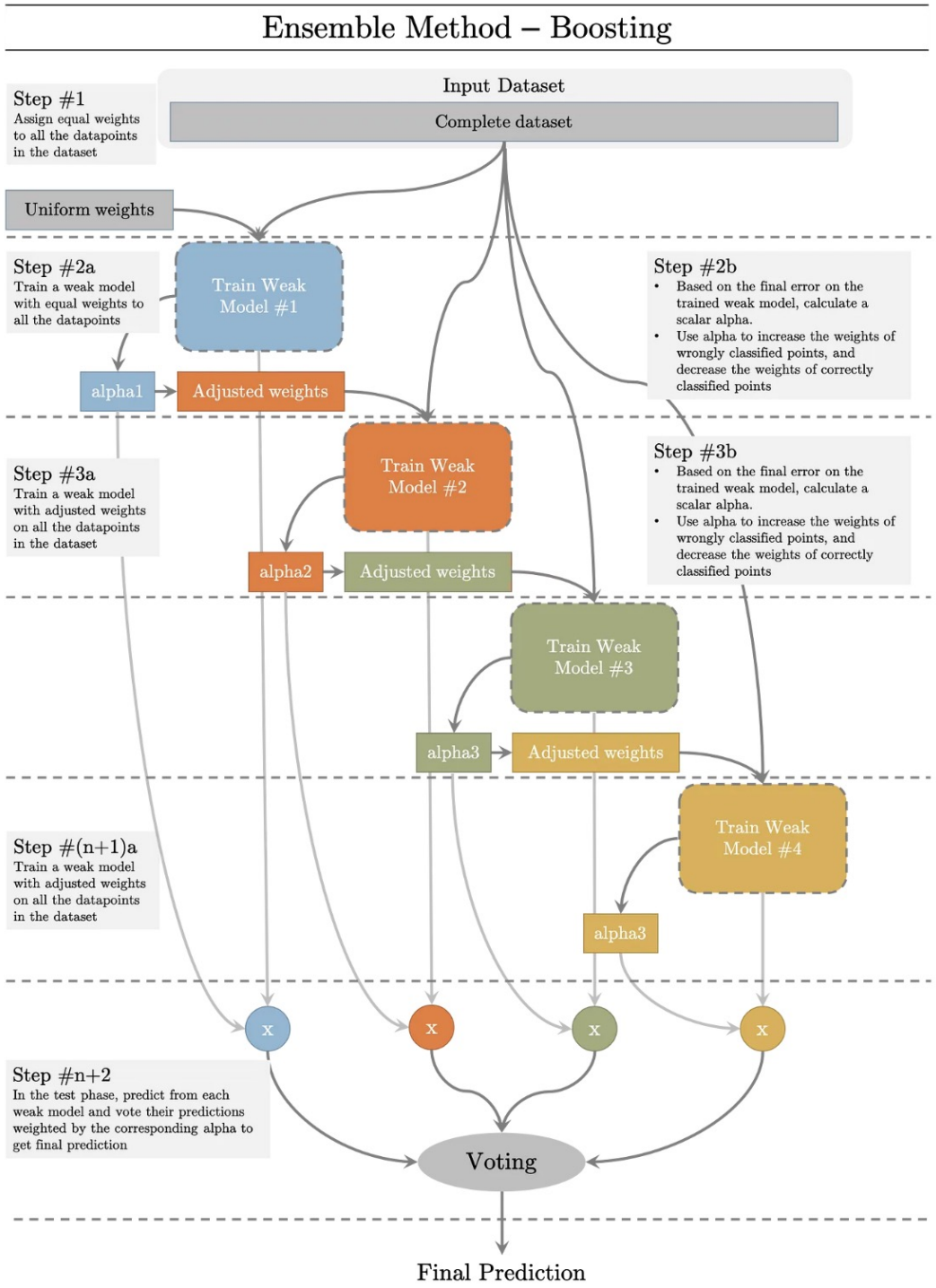
Boosting



https://towardsdatascience.com/what-are-ensemble-methods-in-machine-learning-cac1d17ed349

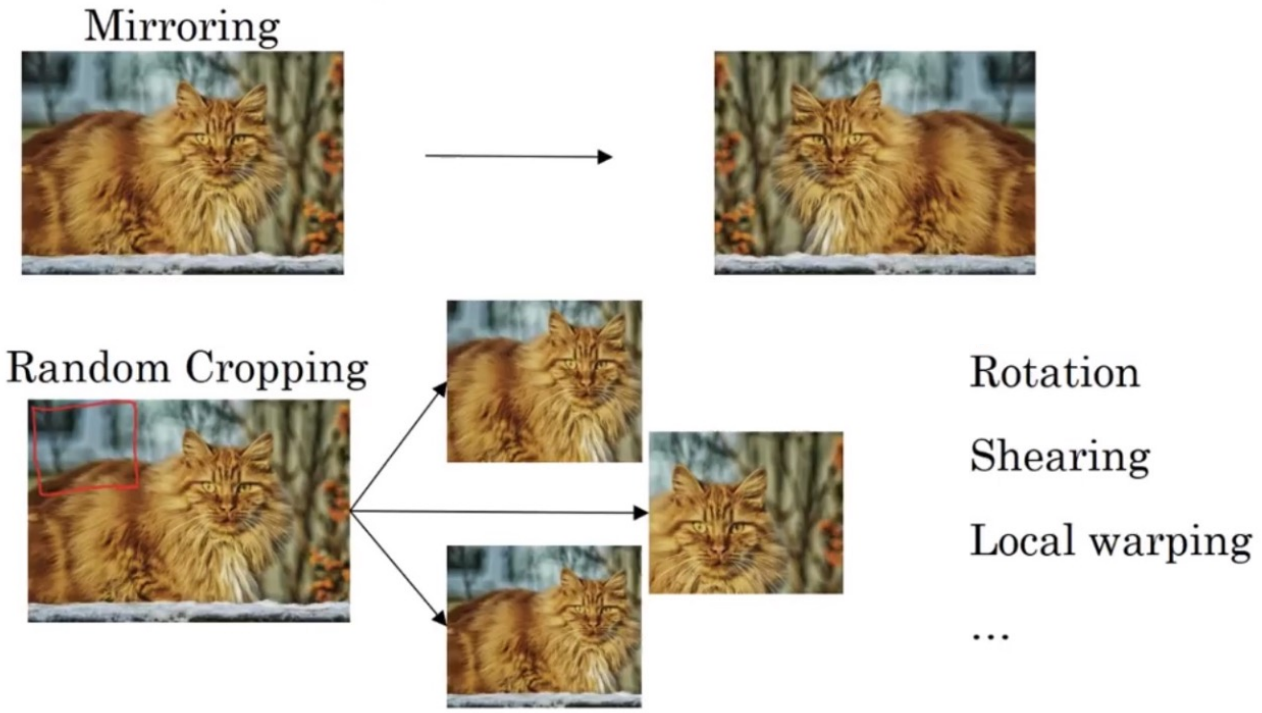
University of Massachusetts Amherst

Ensemble Methods: Boosting



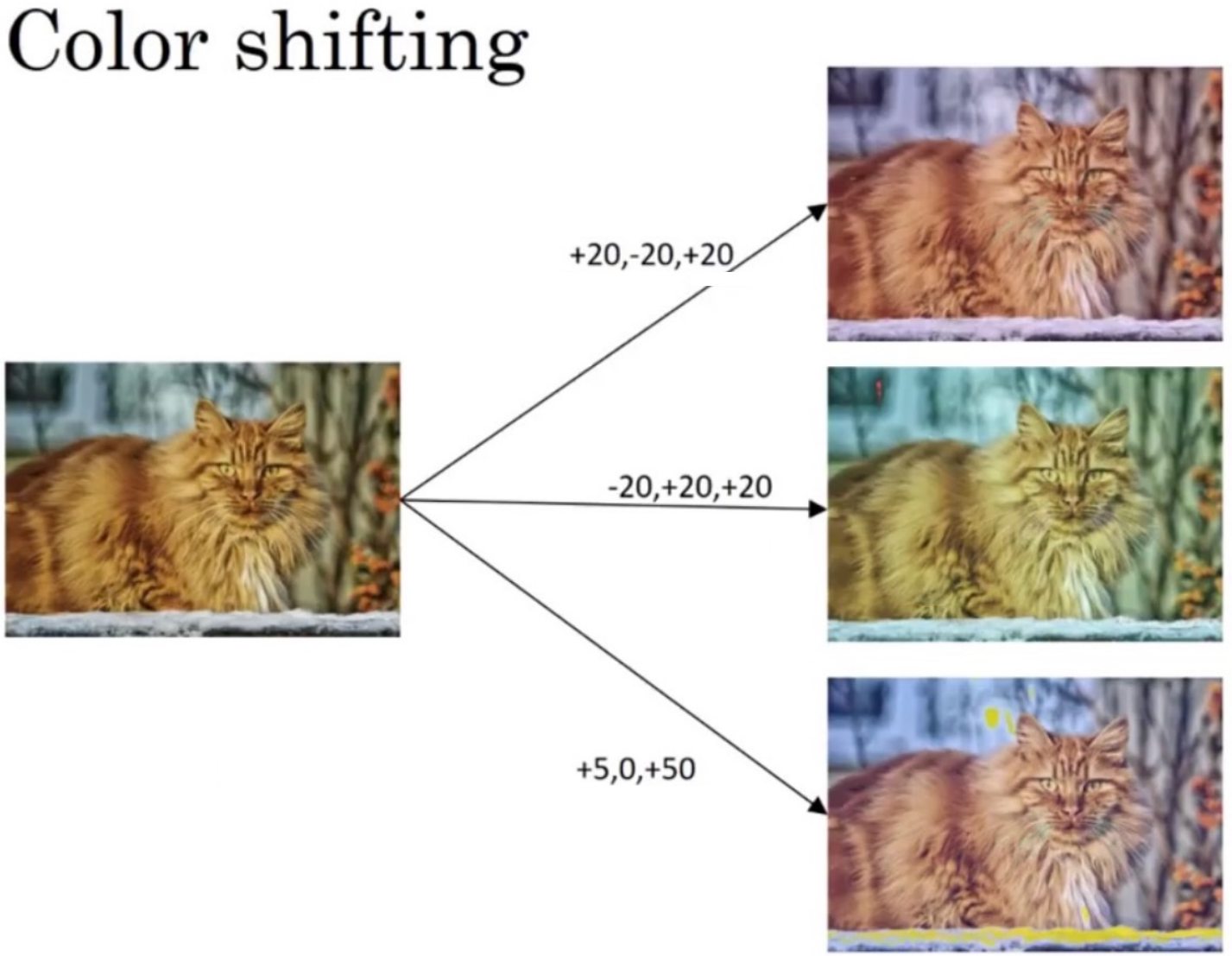
Data Augmentation

- Why data augmentation -> increase training data



Credit: Andrew Ng

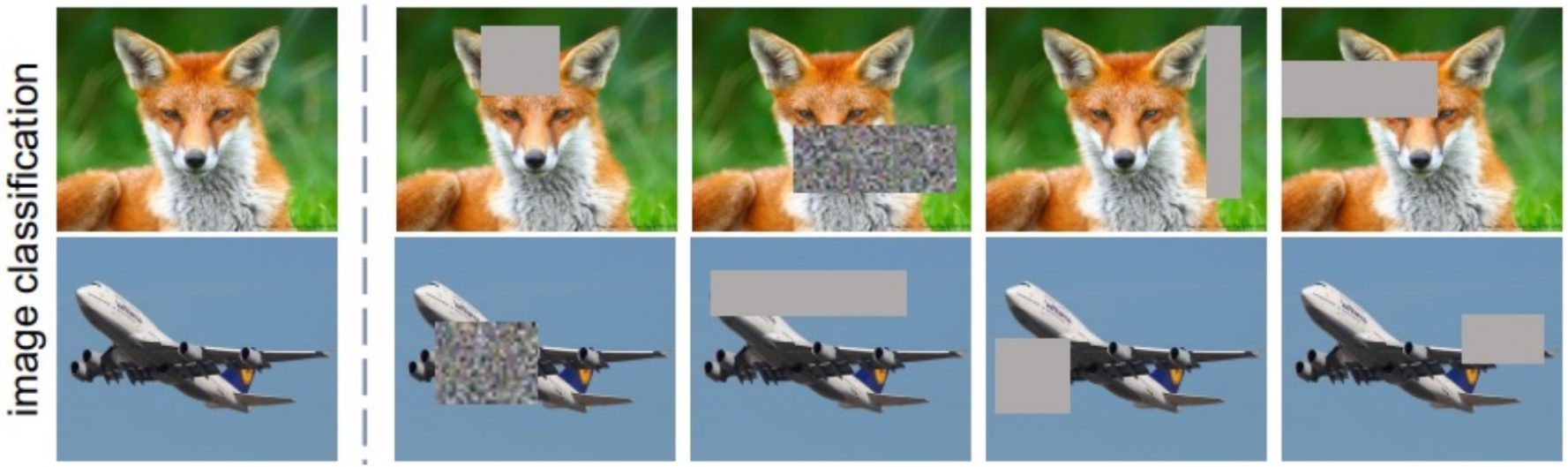
Data Augmentation



Credit: Andrew Ng

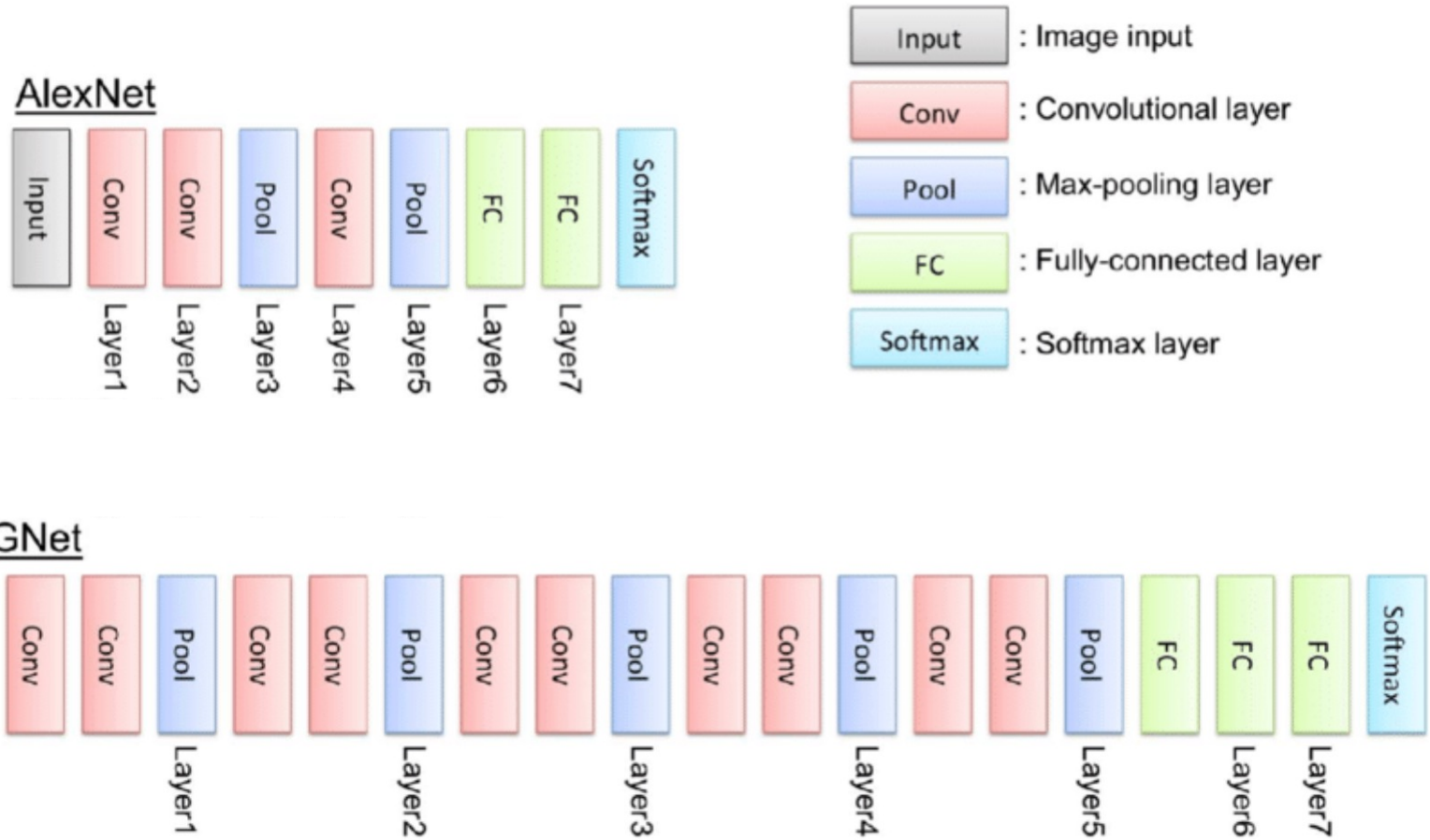
Data Augmentation

- Random erasing

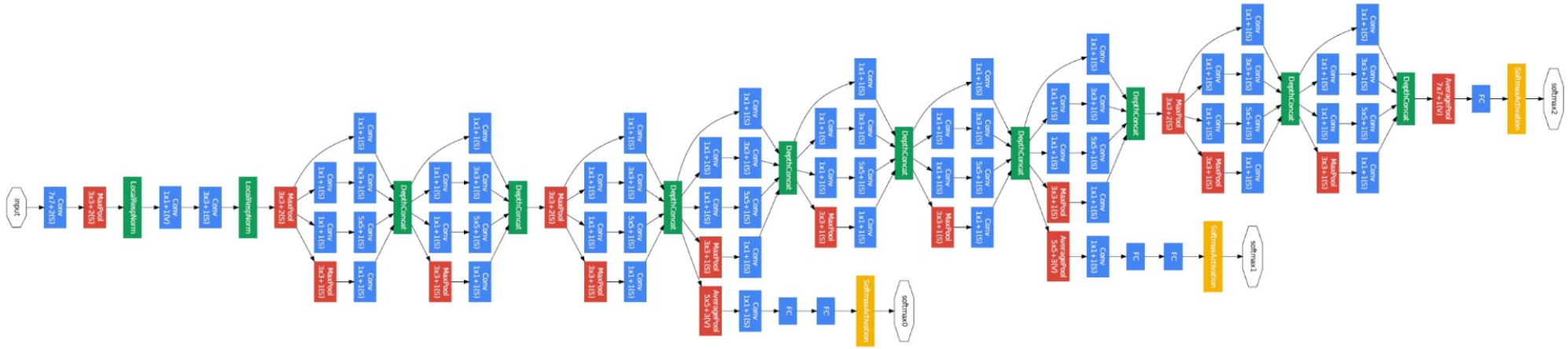


Reduces the risk of over-fitting and makes the model robust to occlusion
Improve the generalization ability of CNNs

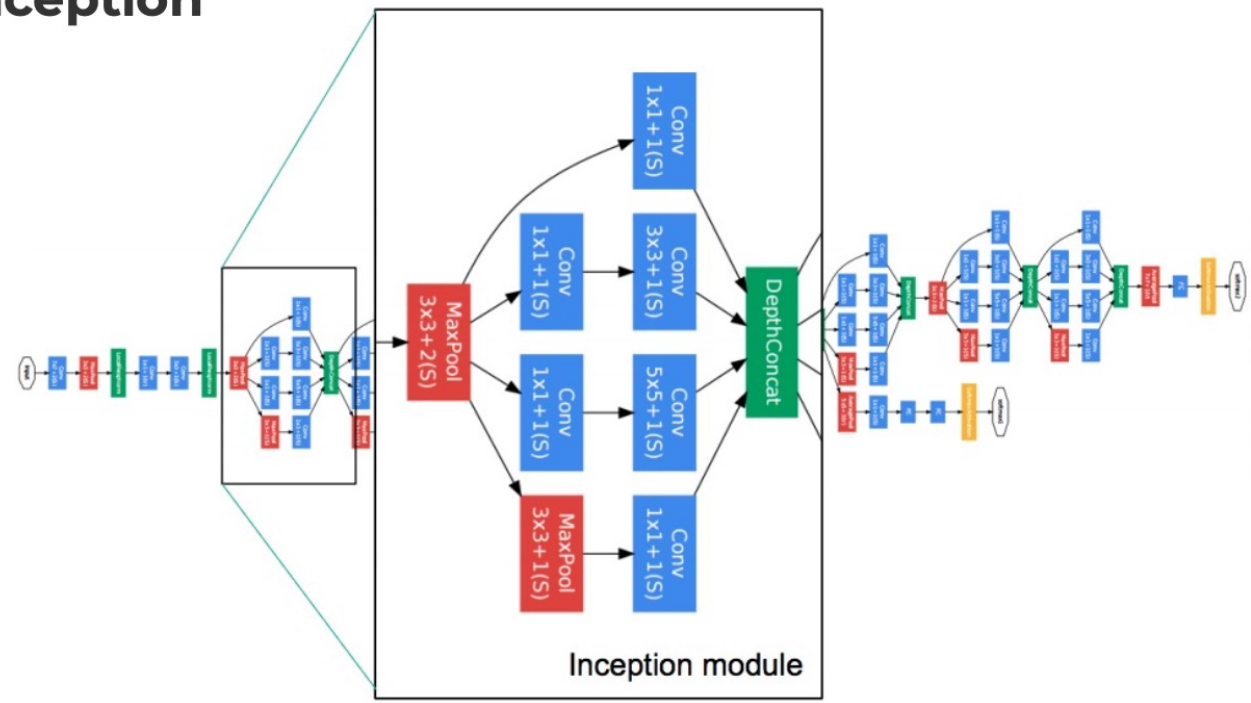
Neural Nets


AlexNet
Input
Conv
Conv
Pool
Conv
Pool
FC
FC
Softmax
Layer1
Layer2
Layer3
Layer4
Layer5
Layer6
Layer7
Input : Image input
Conv : Convolutional layer
Pool : Max-pooling layer
FC : Fully-connected layer
Softmax : Softmax layer
VGGNet
Input
Conv
Conv
Pool
Conv
Conv
Pool
Conv
Conv
Pool
Conv
Conv
Pool
Conv
Conv
Pool
FC
FC
FC
Softmax
Layer1
Layer2
Layer3
Layer4
Layer5
Layer6
Layer7

Neural Nets

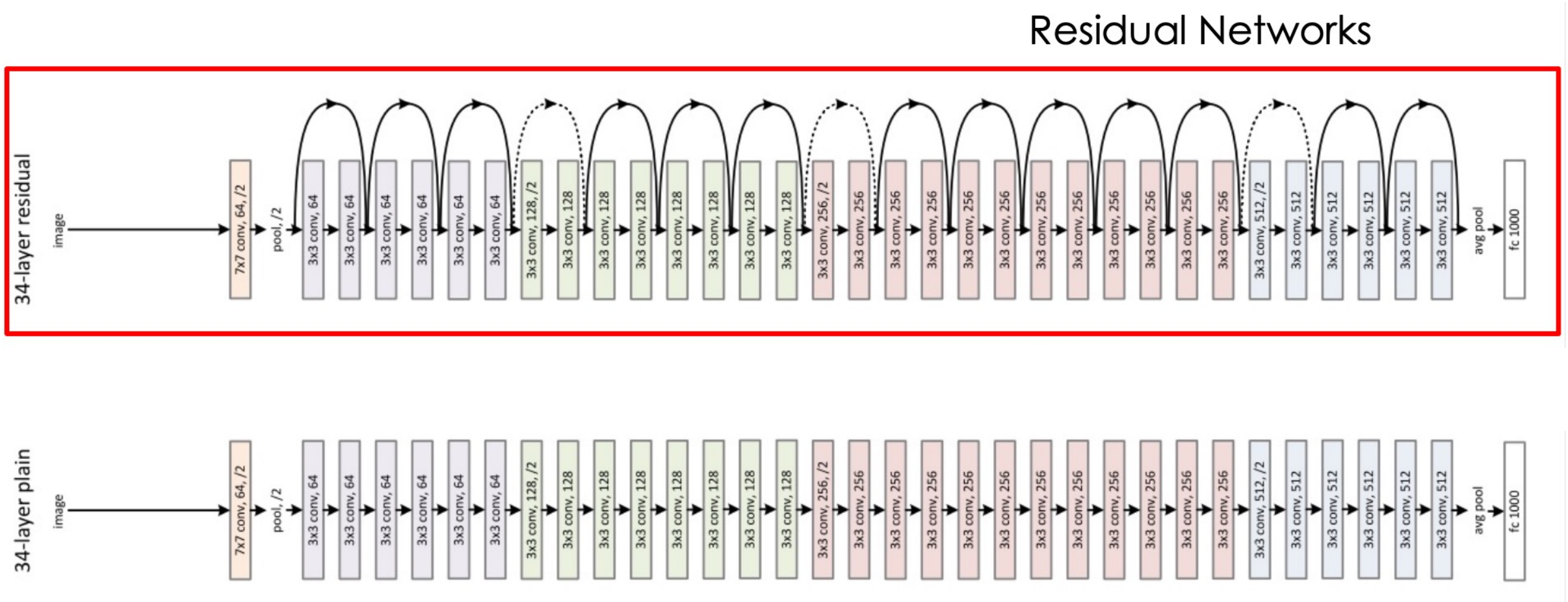


GoogLeNet/Inception



University of Massachusetts Amherst

Neural Nets



Residual Nets

- Deep networks performs worse
  - As we add more layers
- Problem
  - Vanishing gradients
- It models
  - H(x) = F(x) + x
- Skip connections
  - Help in backpropagation

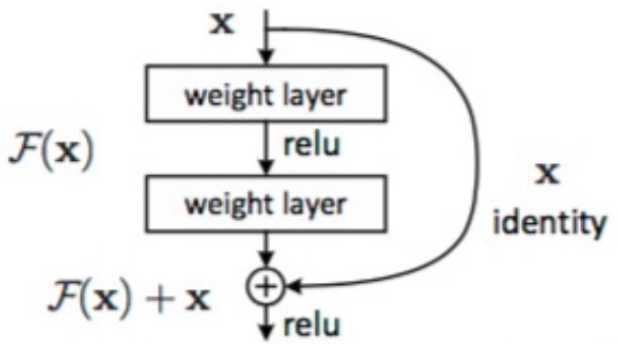


Figure 2. Residual learning: a building block.

He et. al. Deep Residual Learning for Image Recognition, 2015

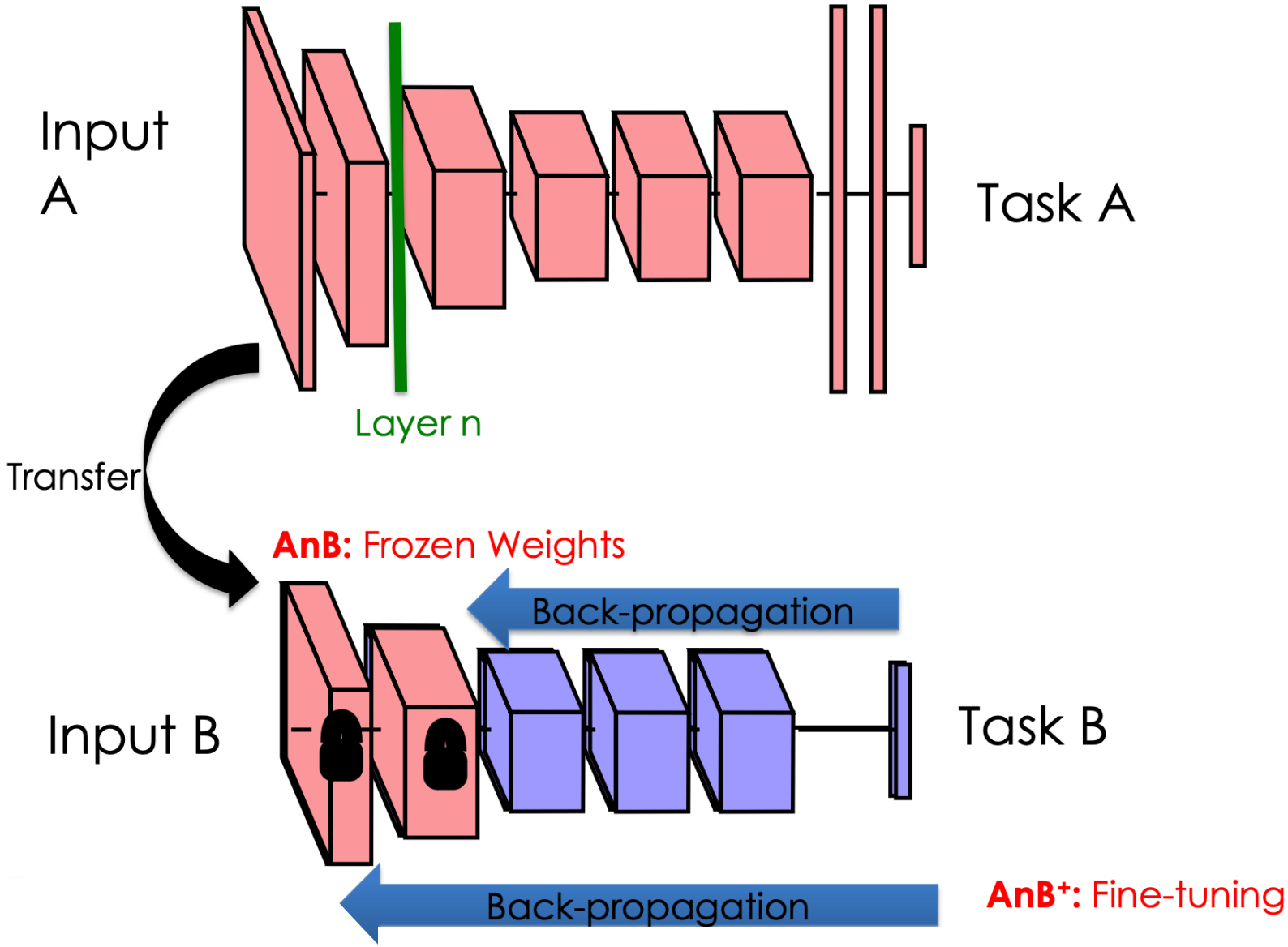
Transfer Learning

- Improvement of learning in a **new** task through the *transfer of knowledge* from a **related** task that has already been learned.
- Weight initialization for CNN

- Two major strategies
  - ConvNet as fixed feature extractor
  - Fine-tuning the ConvNet

University of Massachusetts Amherst

Transfer Learning



(image: Aghamirzaie & Salomon)

When & how to fine tune?

- Suppose we have model A, trained on dataset A
- Q: How do we apply transfer learning to dataset B to create model B?

https://cs231n.github.io/transfer-learning/

When & how to fine tune?

- New dataset is small and similar to original dataset.
  - train a linear classifier on the CNN codes

- Suppose we have model A, trained on dataset A

- Q: How do we apply transfer learning to dataset B to create model B?

https://cs231n.github.io/transfer-learning/

When & how to fine tune?

- Suppose we have model A, trained on dataset A
- Q: How do we apply transfer learning to dataset B to create model B?

- New dataset is small and similar to original dataset.
  - train a linear classifier on the CNN codes
- New dataset is large and similar to the original dataset
  - fine-tune through the full network

https://cs231n.github.io/transfer-learning/

When & how to fine tune?

- Suppose we have model A, trained on dataset A
- Q: How do we apply transfer learning to dataset B to create model B?

- New dataset is small and similar to original dataset.
  - train a linear classifier on the CNN codes
- New dataset is large and similar to the original dataset
  - fine-tune through the full network
- New dataset is small but very different from the original dataset
  - SVM classifier from activations somewhere earlier in the network

https://cs231n.github.io/transfer-learning/

When & how to fine tune?

- Suppose we have model A, trained on dataset A
- Q: How do we apply transfer learning to dataset B to create model B?

- New dataset is small and similar to original dataset.
  - train a linear classifier on the CNN codes
- New dataset is large and similar to the original dataset
  - fine-tune through the full network
- New dataset is small but very different from the original dataset
  - SVM classifier from activations somewhere earlier in the network
- New dataset is large and very different from the original dataset
  - fine-tune through the entire network

https://cs231n.github.io/transfer-learning/

When & how to fine tune?

- Suppose we have model A, trained on dataset A
- Q: How do we apply transfer learning to dataset B to create model B?

- New dataset is small and similar to original dataset.
  - train a linear classifier on the CNN codes
- New dataset is large and similar to the original dataset
  - fine-tune through the full network
- New dataset is small but very different from the original dataset
  - SVM classifier from activations somewhere earlier in the network
- New dataset is large and very different from the original dataset
  - fine-tune through the entire network

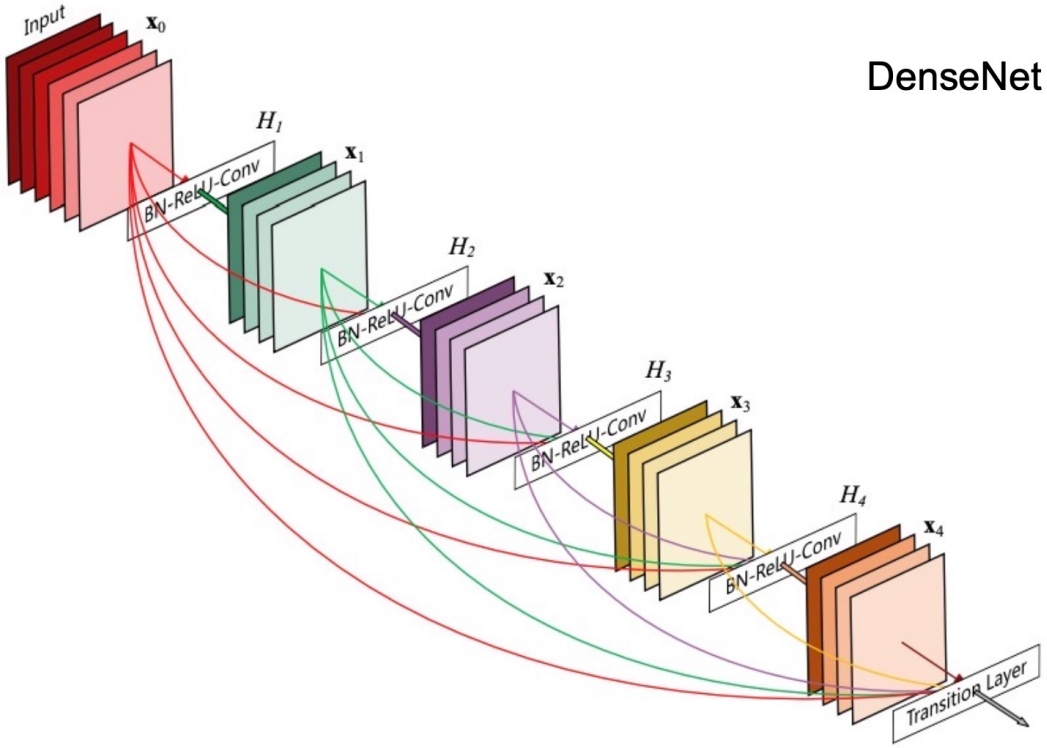
Dataset size	Dataset similarity	Recommendation
Large	Very different	Train model B from scratch, initialize weights from model A
Large	Similar	OK to fine-tune (less likely to overfit)
Small	Very different	Train classifier using the earlier layers (later layers won't help much)
Small	Similar	Don't fine-tune (overfitting). Train a linear classifier

https://cs231n.github.io/transfer-learning/

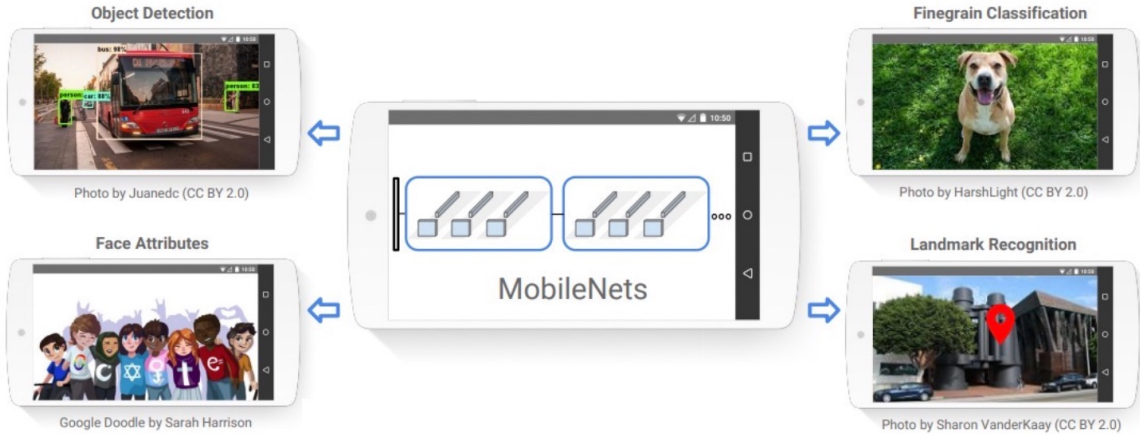
University of Massachusetts Amherst

Common CNNs

DenseNet



MobileNet



Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.v