Deep Learning - Regularization

Itthi Chatnuntawech

Fully connected layer
(Dense)

Convolutional layer

Conv1D, 2D, 3D, …
separable Conv

Optimizer

SGD
Adam
RMSprop

Pooling layer
max-pooling
average-pooling

Evaluation metric

accuracy
F1-score
AUC
confusion matrix

Activation function

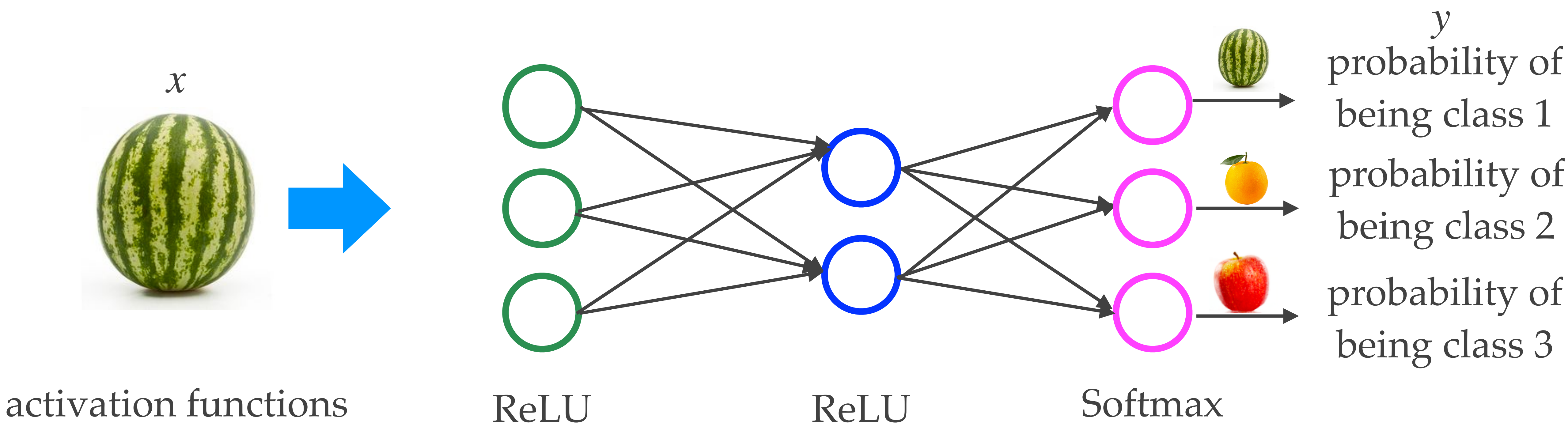
sigmoid
softmax
ESP (swish)
ReLU

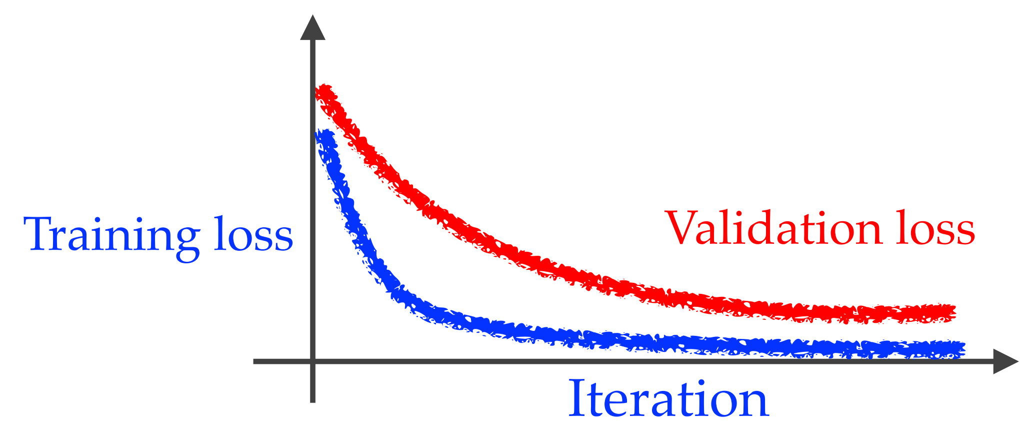
Loss function

categorical crossentropy
binary crossentropy
mean squared error
mean absolute error

Regularization
Dropout
Data augmentation
l_1, l_2 regularizations

- Combine basic components to build a neural network
  - More components → "More" representative power



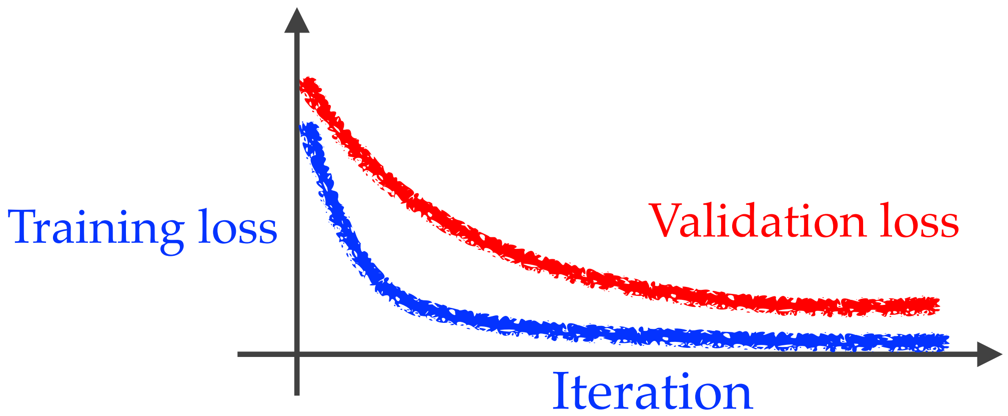


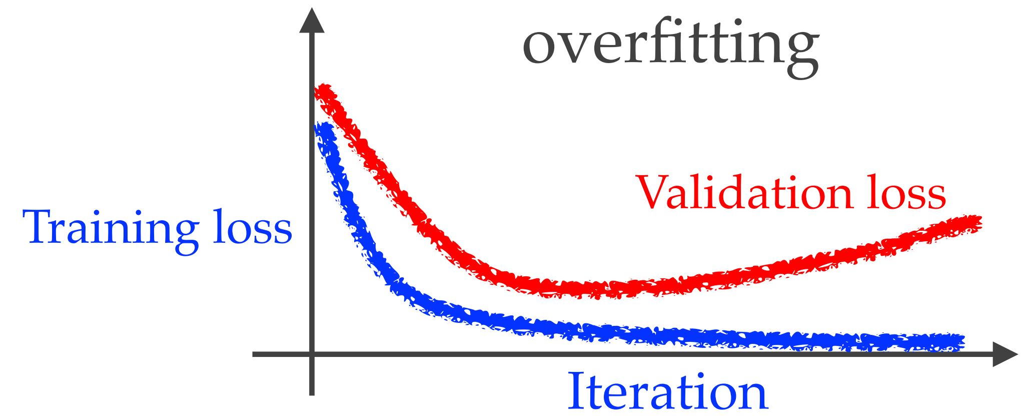

```python
# Import necessary modules
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# Create the model
model = keras.Sequential()
model.add(layers.Dense(3, activation="relu"))
model.add(layers.Dense(2, activation="relu"))
model.add(layers.Dense(3, activation="softmax"))

# Compile the model
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.CategoricalCrossentropy(),
)

# Train the model for 100 epochs with a batch size of 32
model.fit(x_train, y_train, batch_size=32, epochs=100, validation_data=(x_val,y_val))
```

Model

Loss function and optimizer

Regularization





- Regularization is frequently used to mitigate overfitting
  - Add a regularization term to the loss function
    - ℓ_2 regularization

$$min \sum_{i=1}^{N} L(y_i, f_W(x)) \quad \rightarrow \quad min \sum_{i=1}^{N} L(y_i, f_W(x)) + \lambda \|w\|_2^2$$

    - ℓ_1 regularization

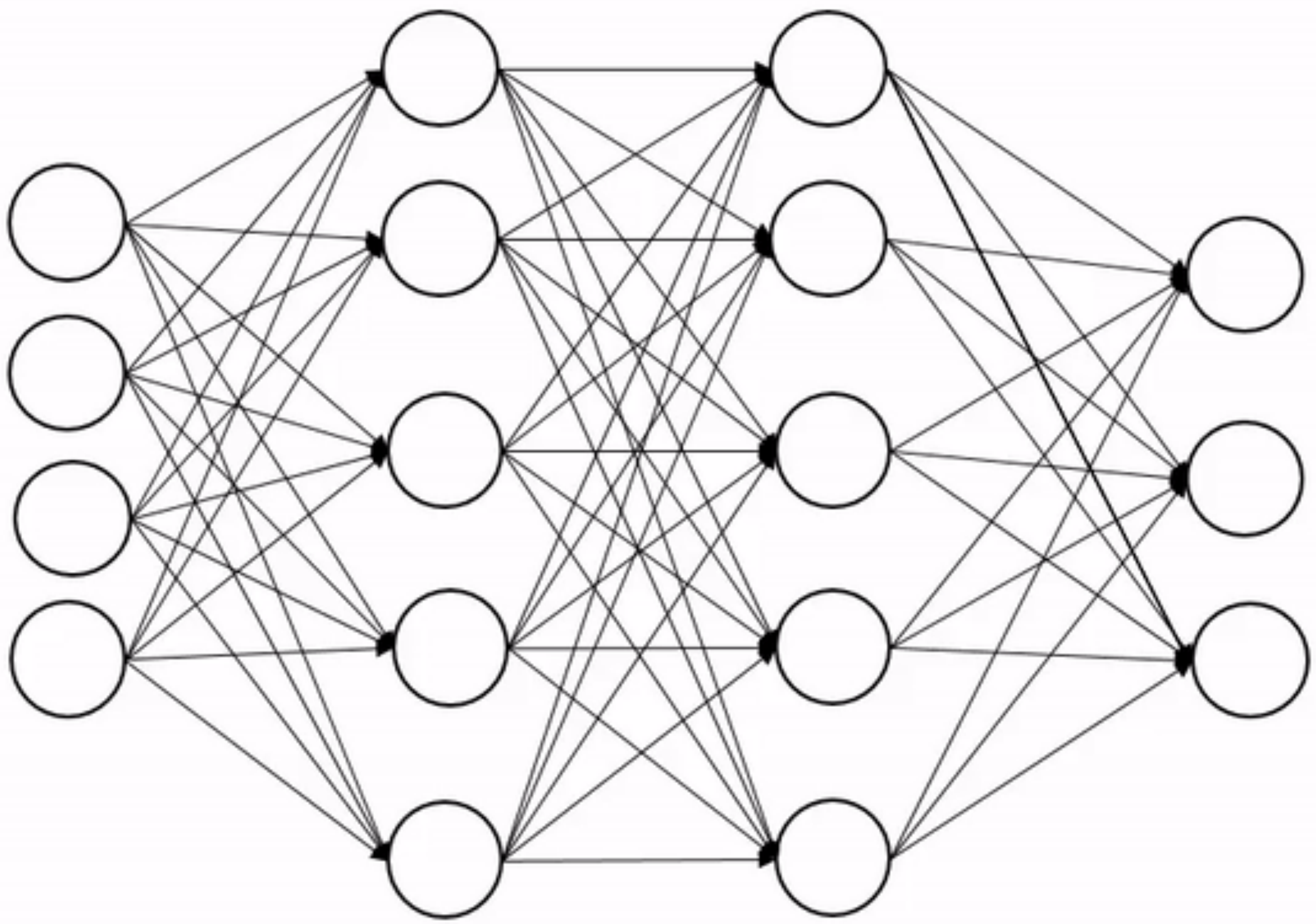
$$min \sum_{i=1}^{N} L(y_i, f_W(x)) \quad \rightarrow \quad min \sum_{i=1}^{N} L(y_i, f_W(x)) + \lambda \|w\|_1$$

  - Use dropout - much more popular

keyword: dropblock
for CNN

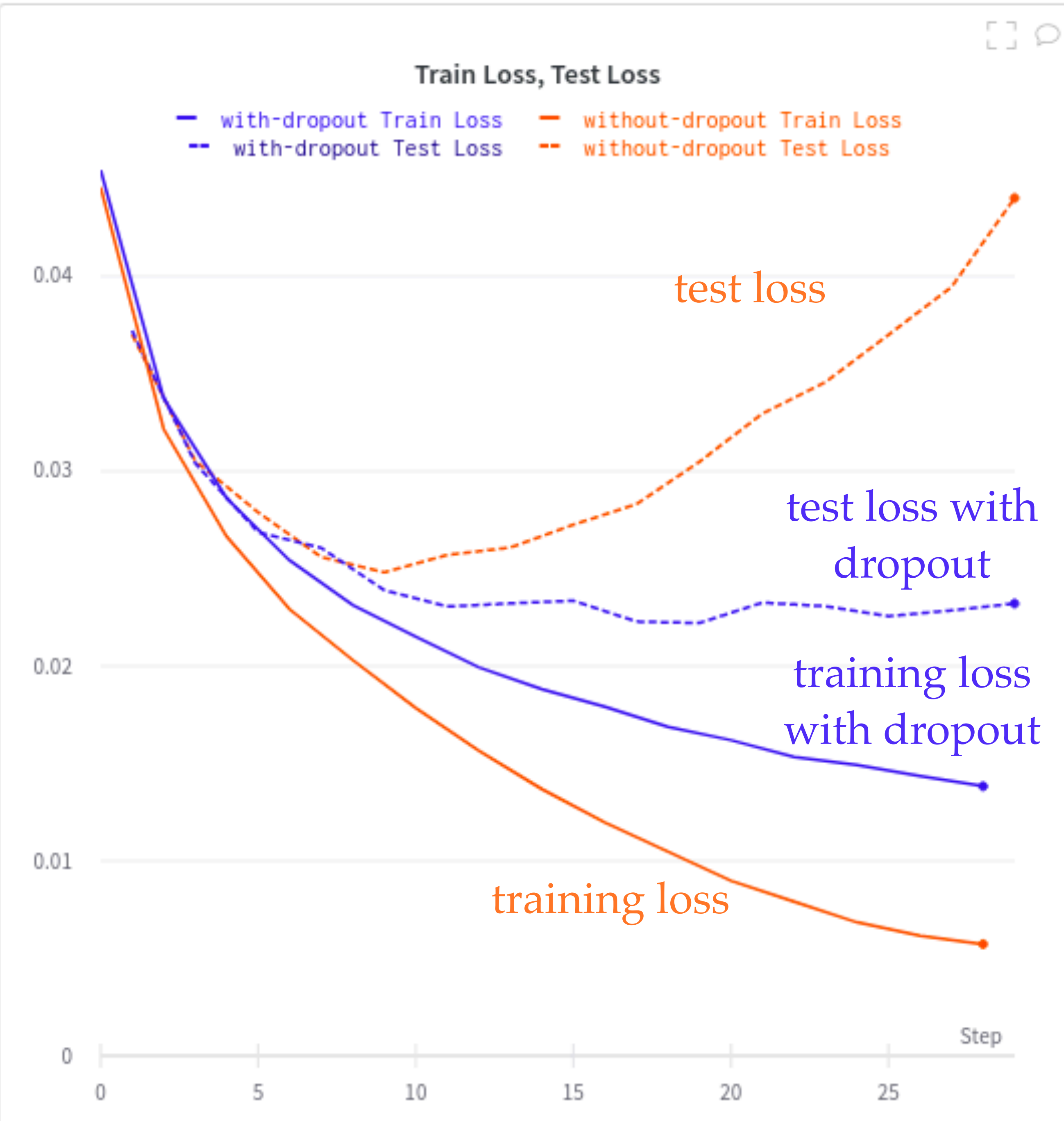
Dropout

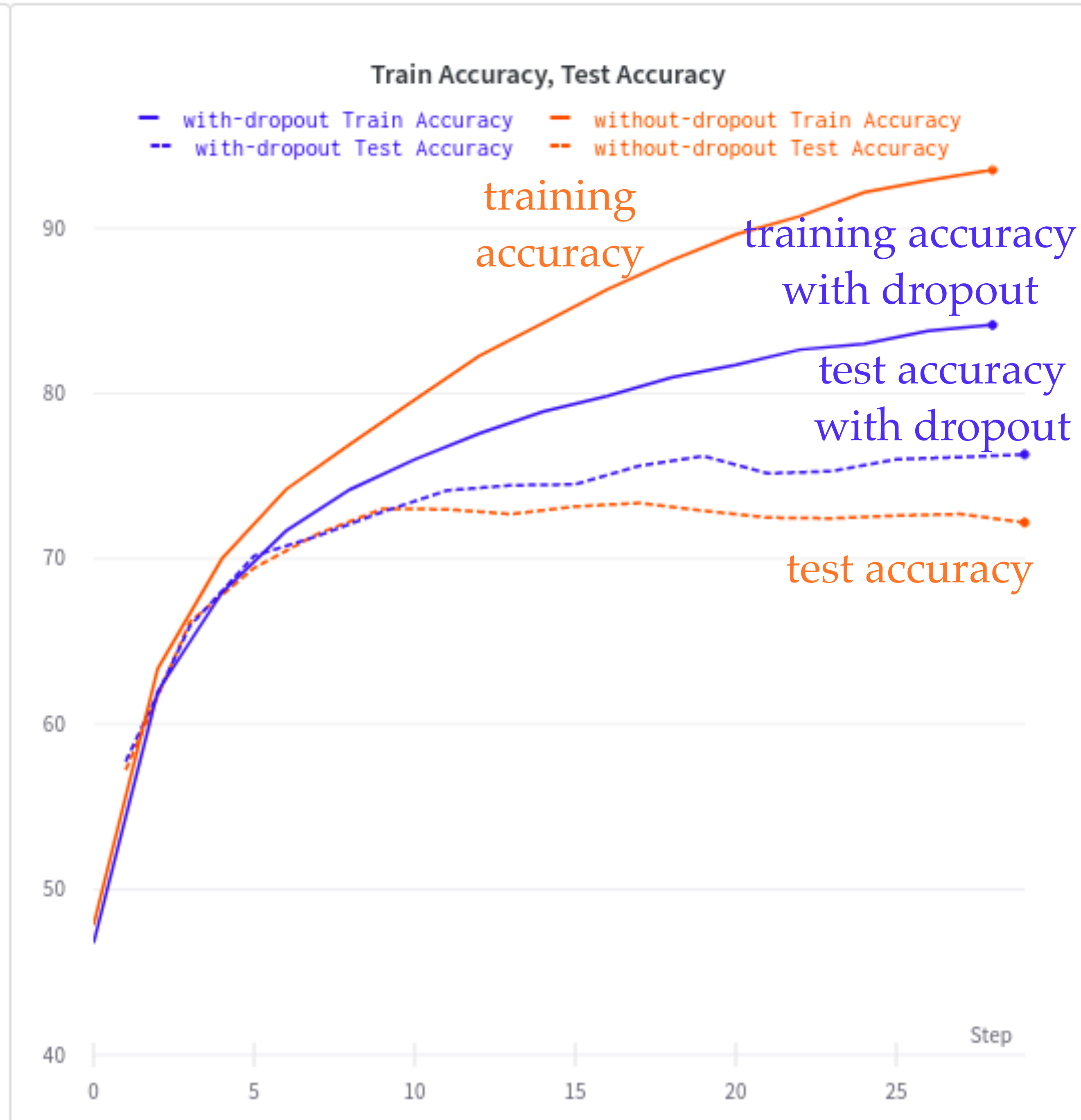
- Dropout can be used to reduce overfitting
  - Randomly omit some neurons / units



[Dropout for Deep Learning Regularization, explained with Examples]

Dropout

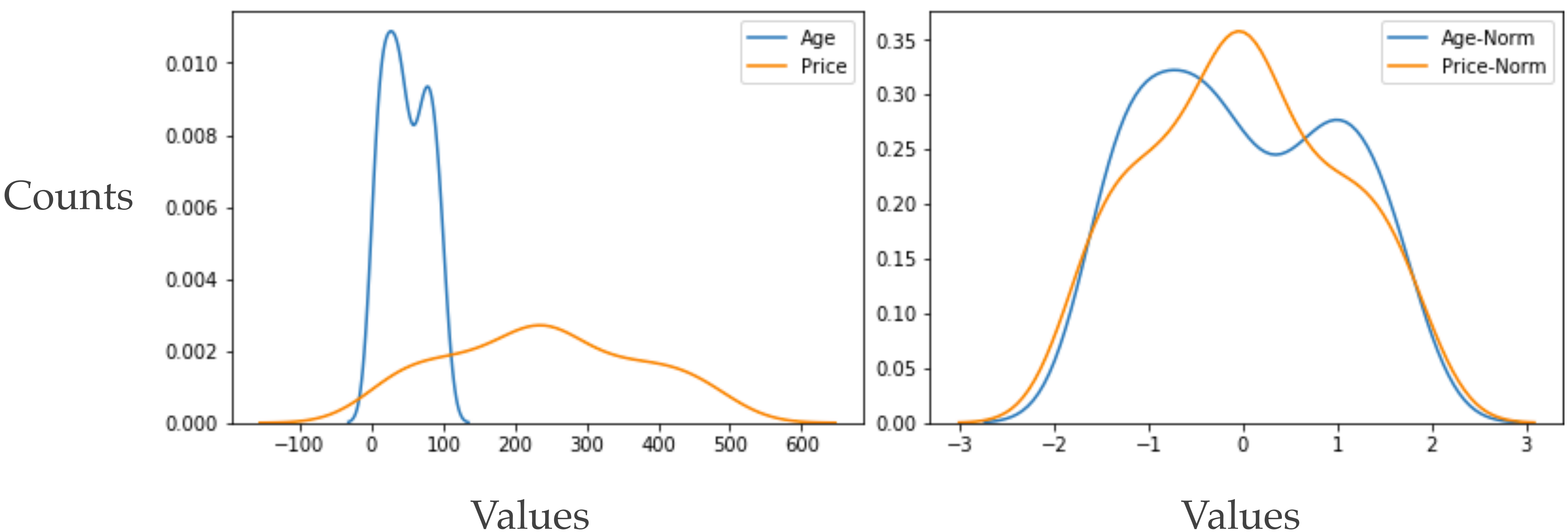




[Dropout for Deep Learning Regularization, explained with Examples]()

Batch Normalization

- Normalize every batch
  - reduce internal covariate shift problems*
- Can be thought of as a form of implicit regularization
  - can help reduce overfitting

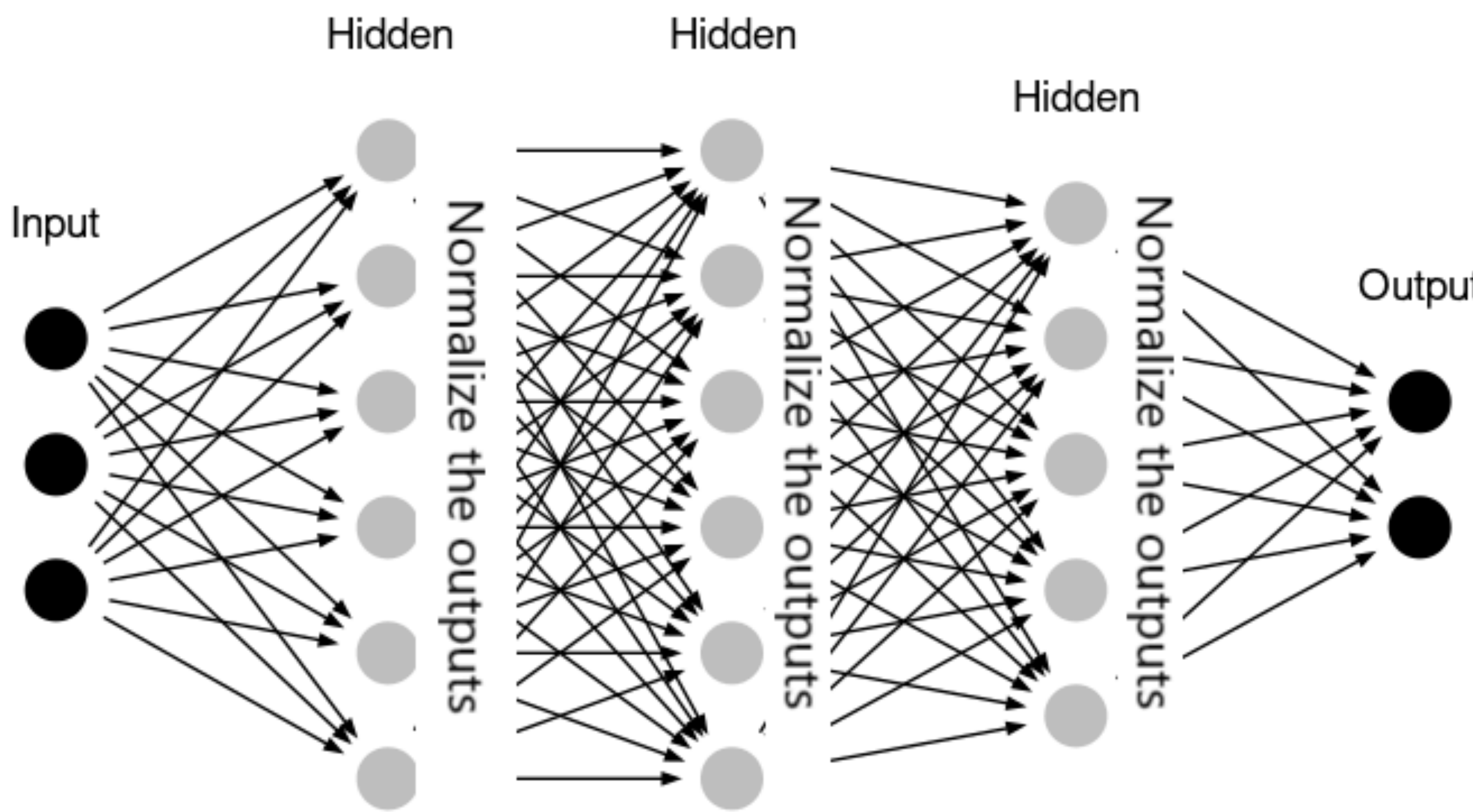


[Batch Normalization — Speed up Neural Network Training]

*This has been challenged by more recent work

Batch Normalization

- Normalize every batch
  - reduce internal covariate shift problems*
- Can be thought of as a form of implicit regularization
  - can help reduce overfitting



Batch Normalization — Speed up Neural Network Training

*This has been challenged by more recent work

keywords: layer normalization, instance normalization, group normalization, standardization

Batch Normalization

- Reduce internal covariate shift problems*

batch size # of features

Input: $x : N \times D$

Learnable params: $\gamma, \beta : D$

Intermediates: $\mu, \sigma : D$, $\hat{x} : N \times D$

Output: $y : N \times D$

$$\mu_j = \frac{1}{N} \sum_{i=1}^{N} x_{i,j}$$

$$\sigma_j^2 = \frac{1}{N} \sum_{i=1}^{N} (x_{i,j} - \mu_j)^2$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

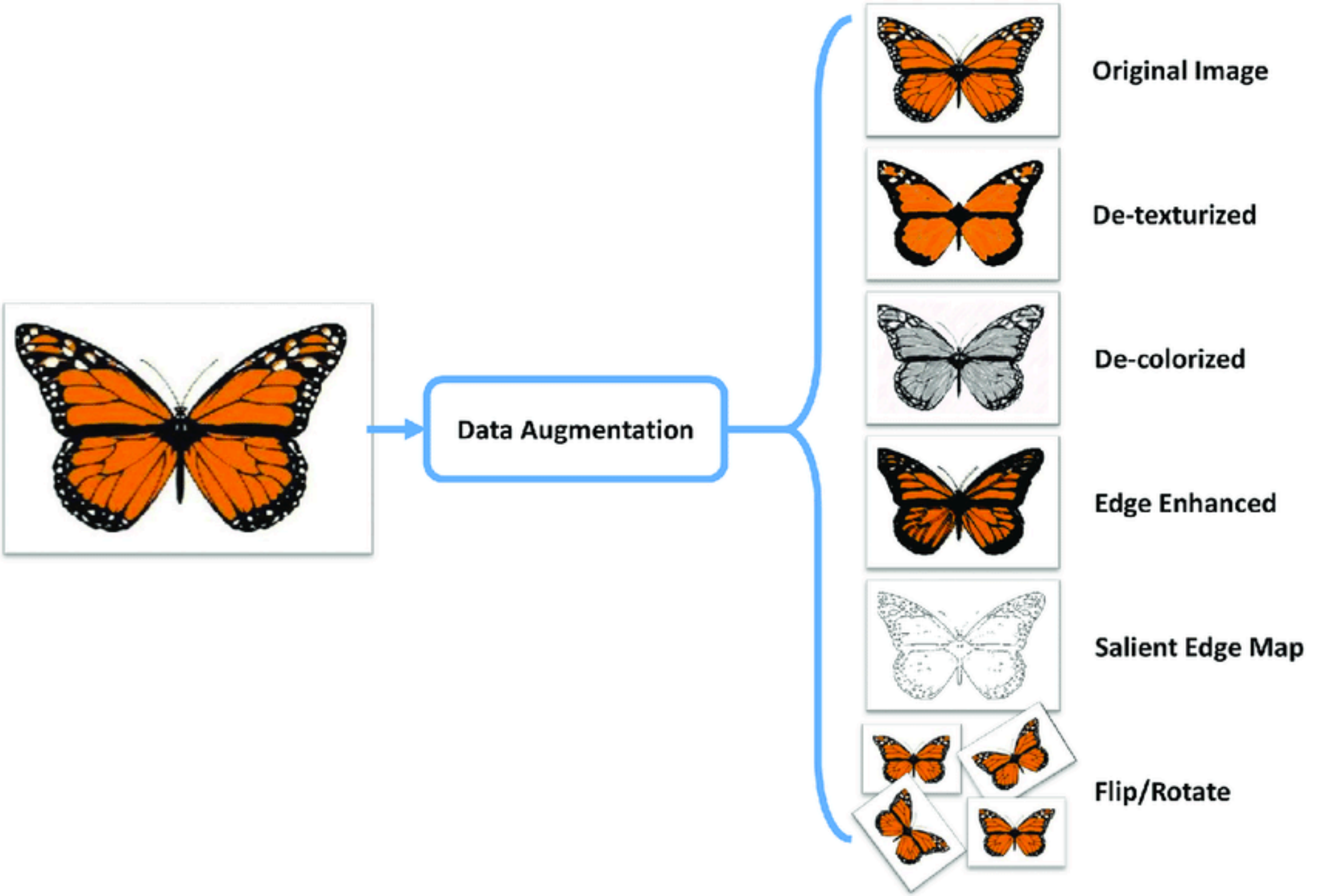
$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

- More robust to bad initialization

Lecture 7 (Stanford, CS231n, 2018)

*This has been challenged by more recent work

Data Augmentation



Data augmentation-assisted deep learning of hand-drawn partially colored sketches for visual search