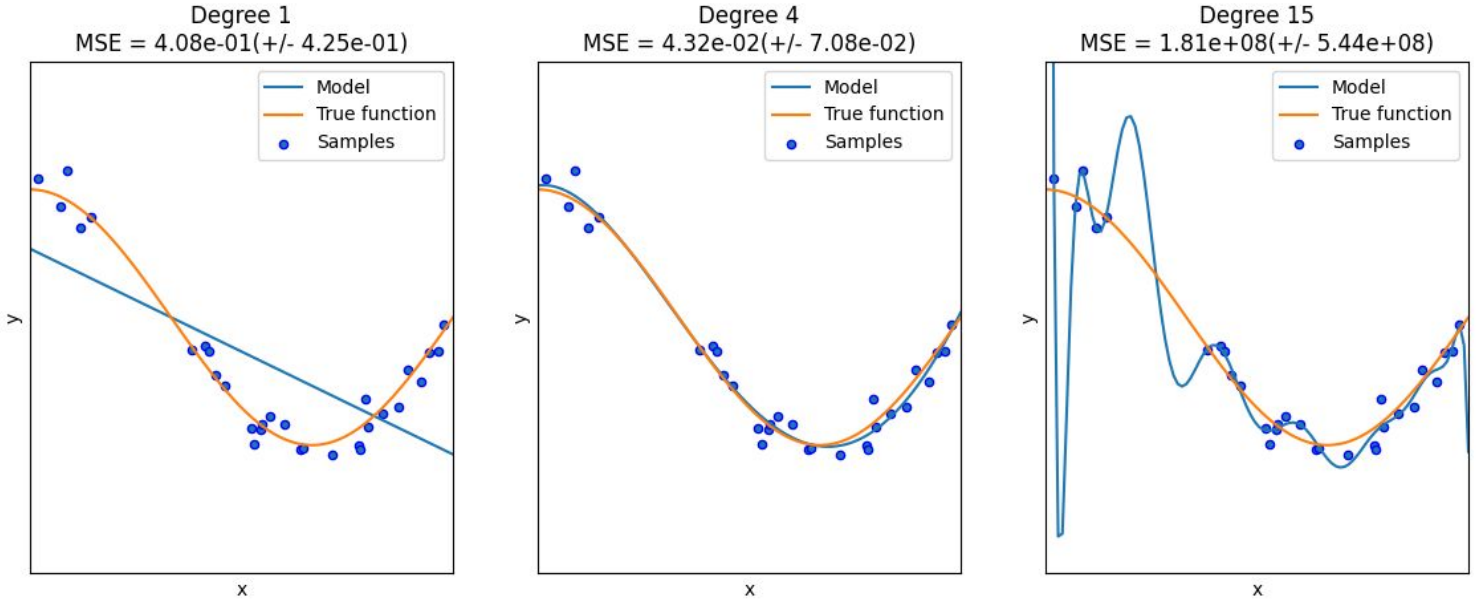
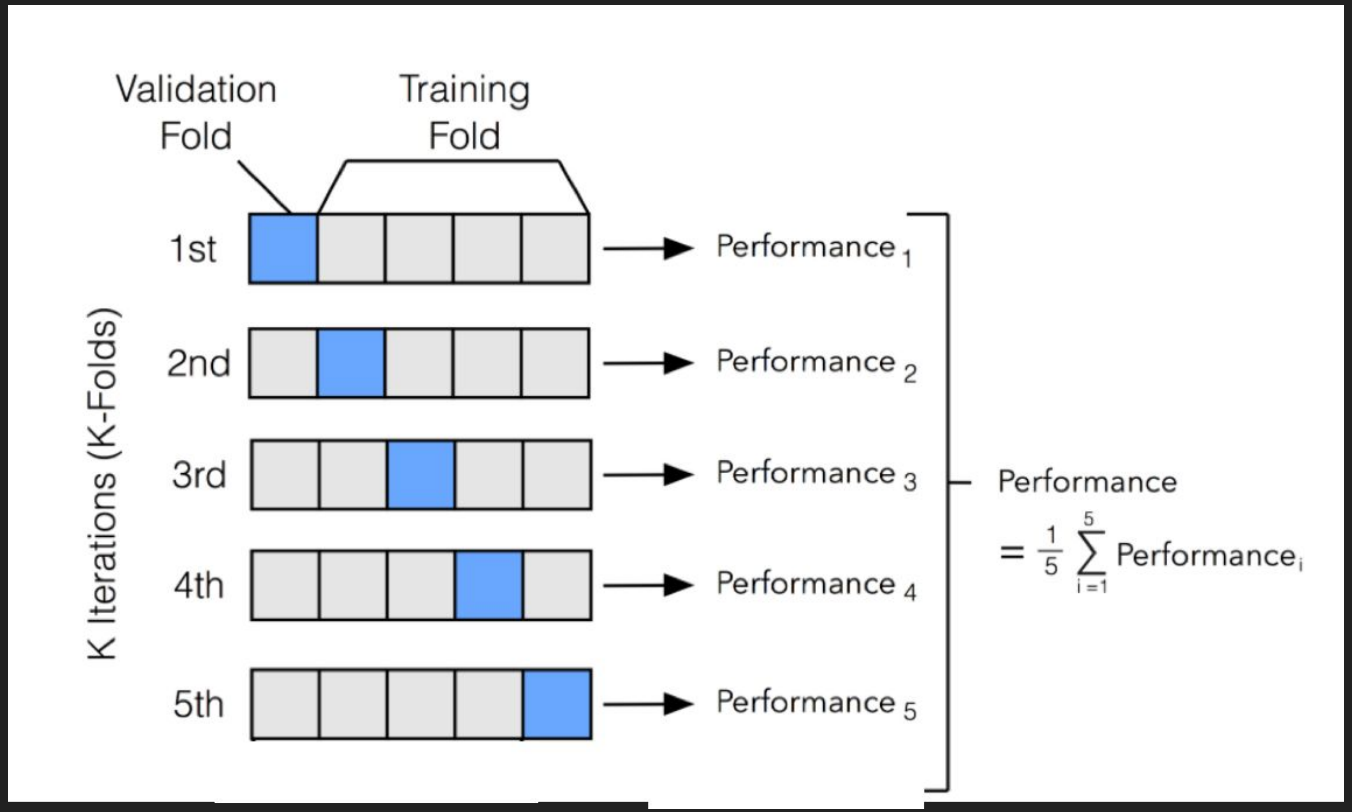
L6

Overfit. Regularization (Batch Normalization, Dropout)

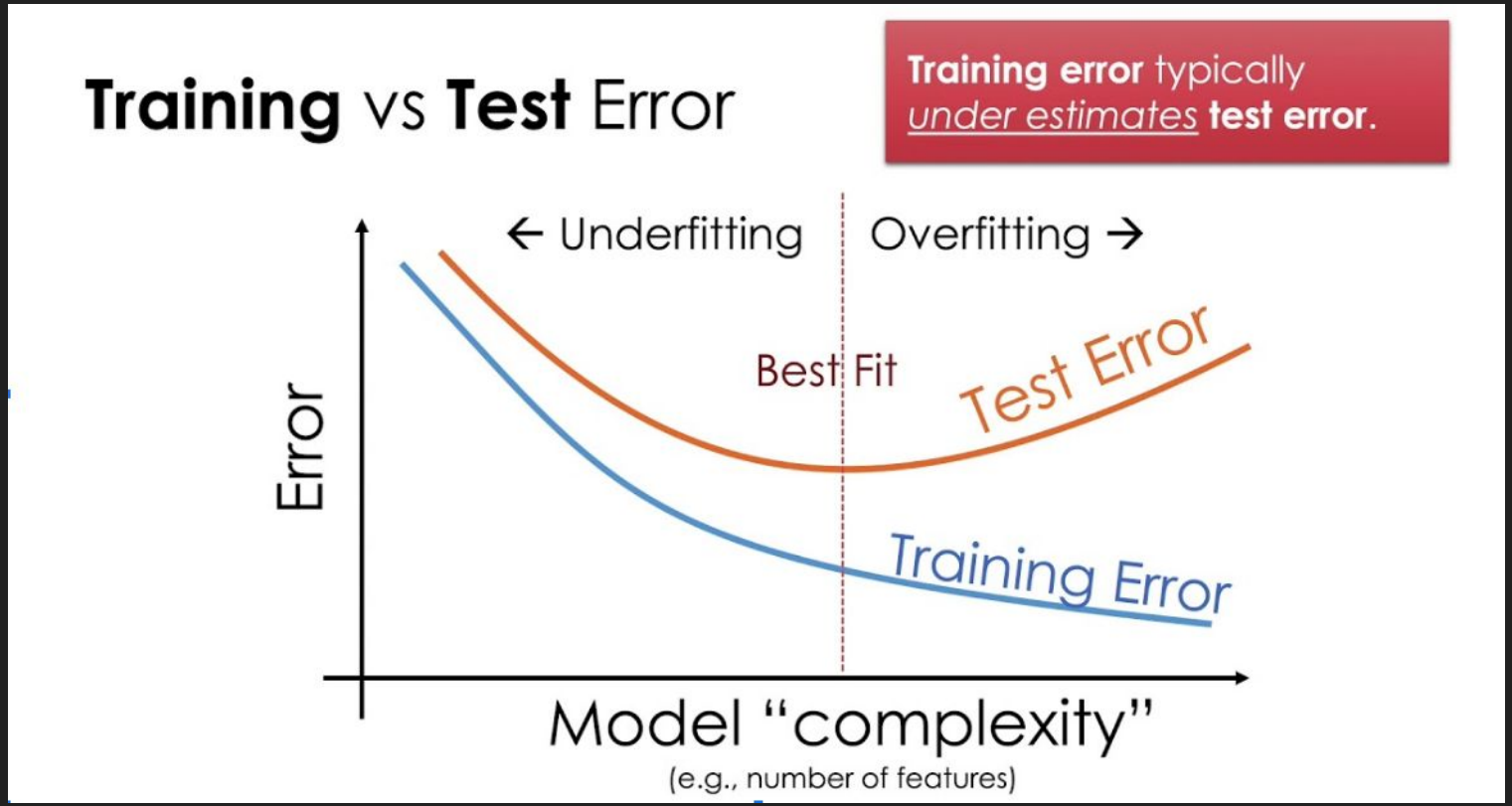
Overfit. Demo.



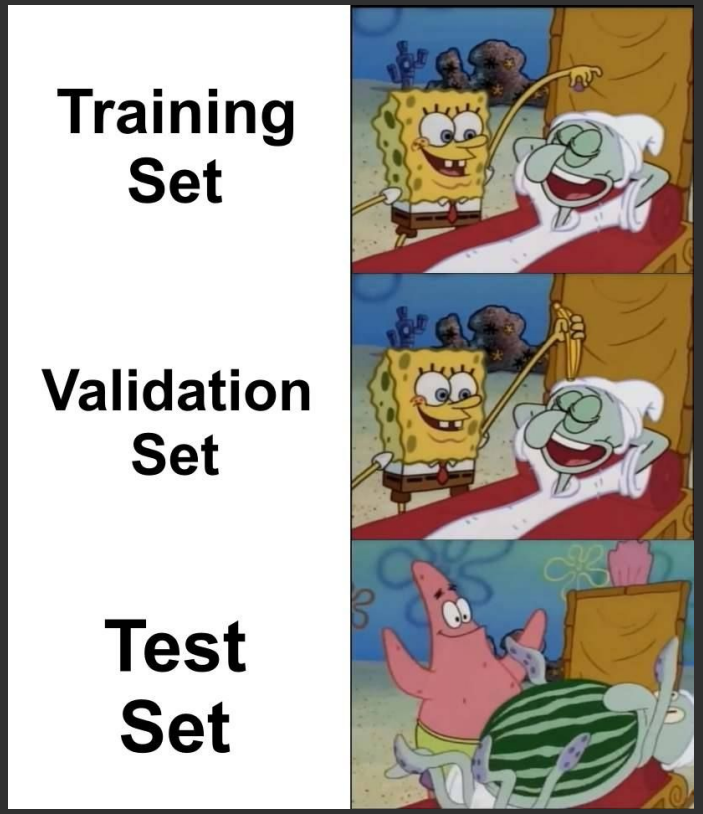
Cross Validation



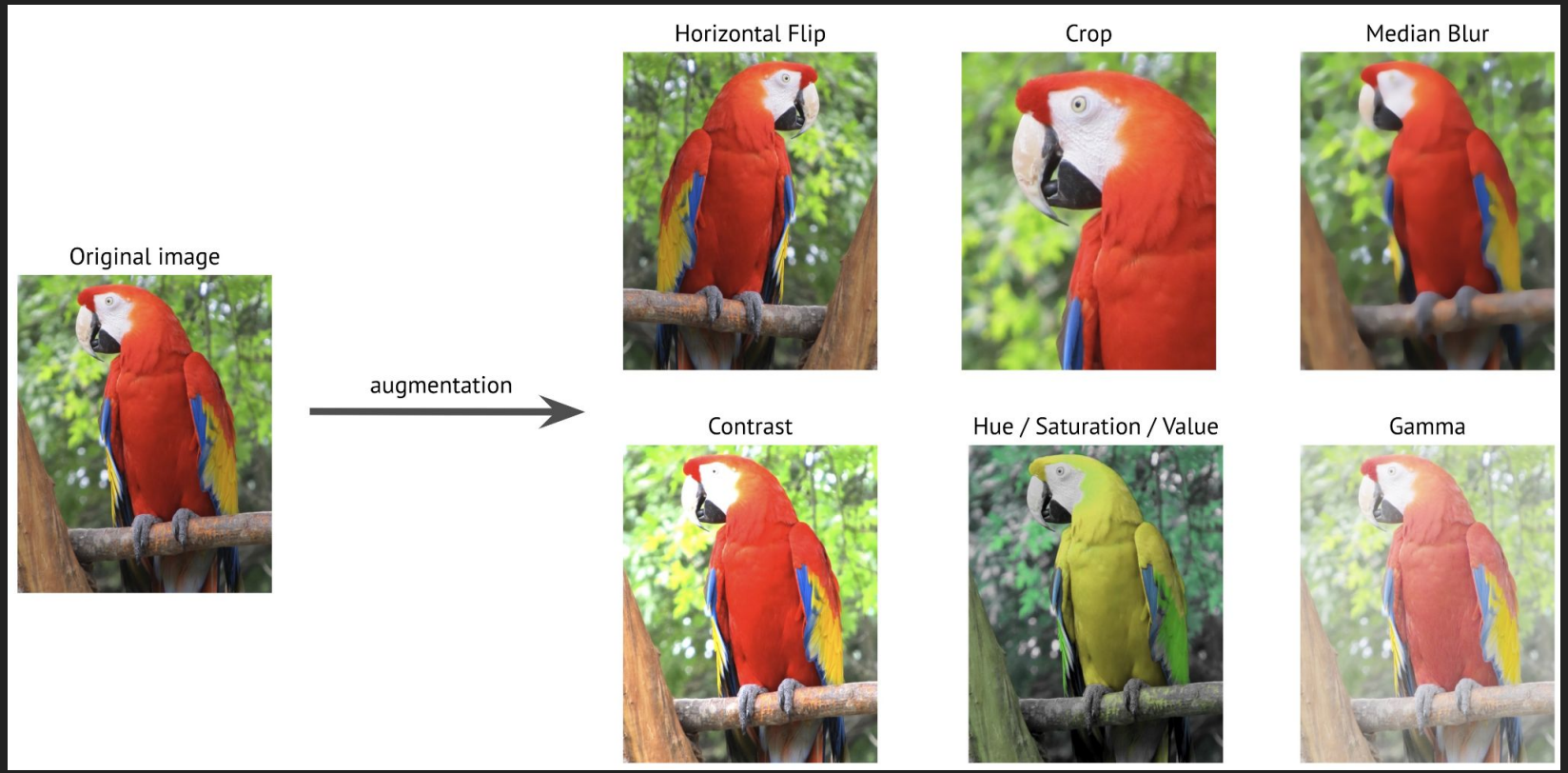
Overfit. Underfit.



Train / Test / Validation Split. Data Leakage.

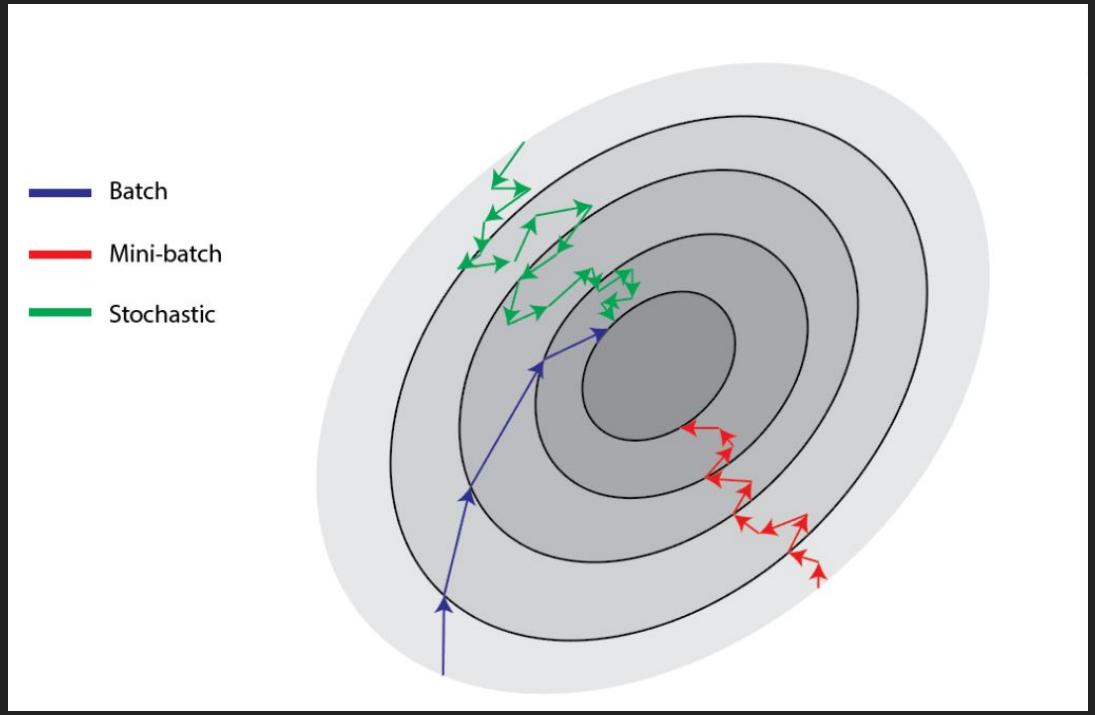


Data Augmentation

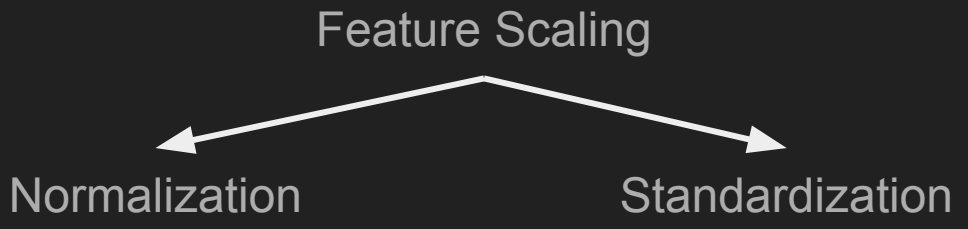


Batch Training

- **Batch Gradient Descent**.
  Batch Size = Size of Training Set
- **Stochastic Gradient Descent**.
  Batch Size = 1
- **Mini-Batch Gradient Descent**.
  1 < Batch Size < Size of Training Set



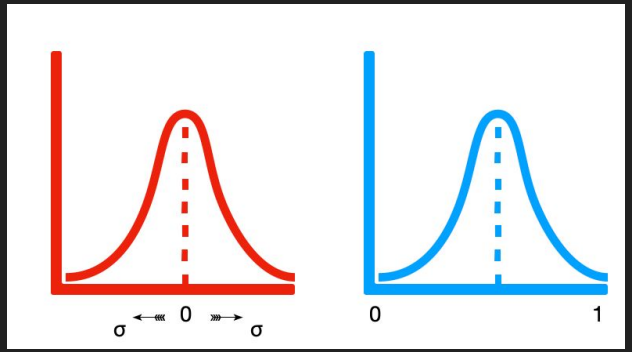
Normalization vs Standardization

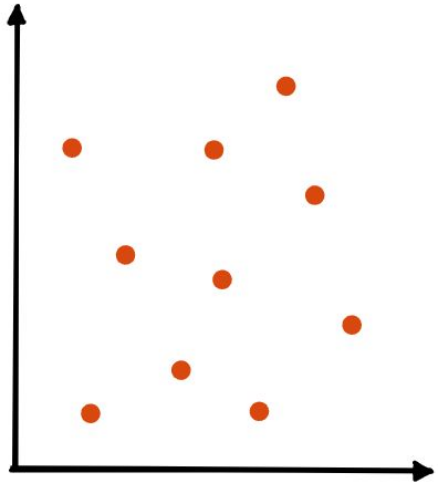


$$z_{norm} = \frac{z - z_{min}}{z_{max} - z_{min}}$$

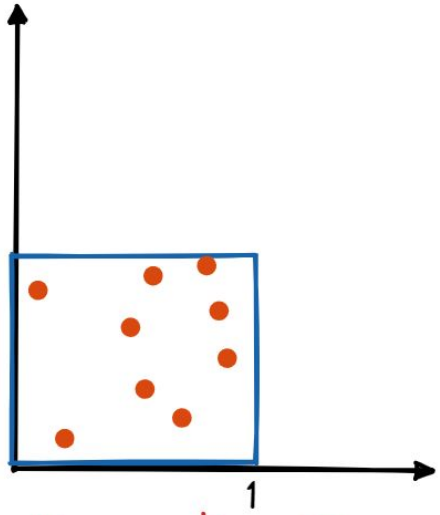
$$z_{std} = \frac{z - \mu}{\sigma}$$

Normalization vs Standardization

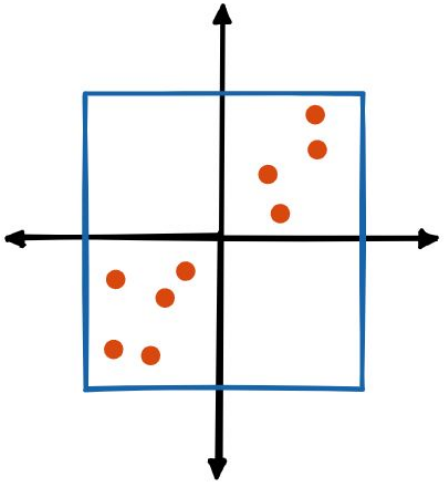




Actual Data

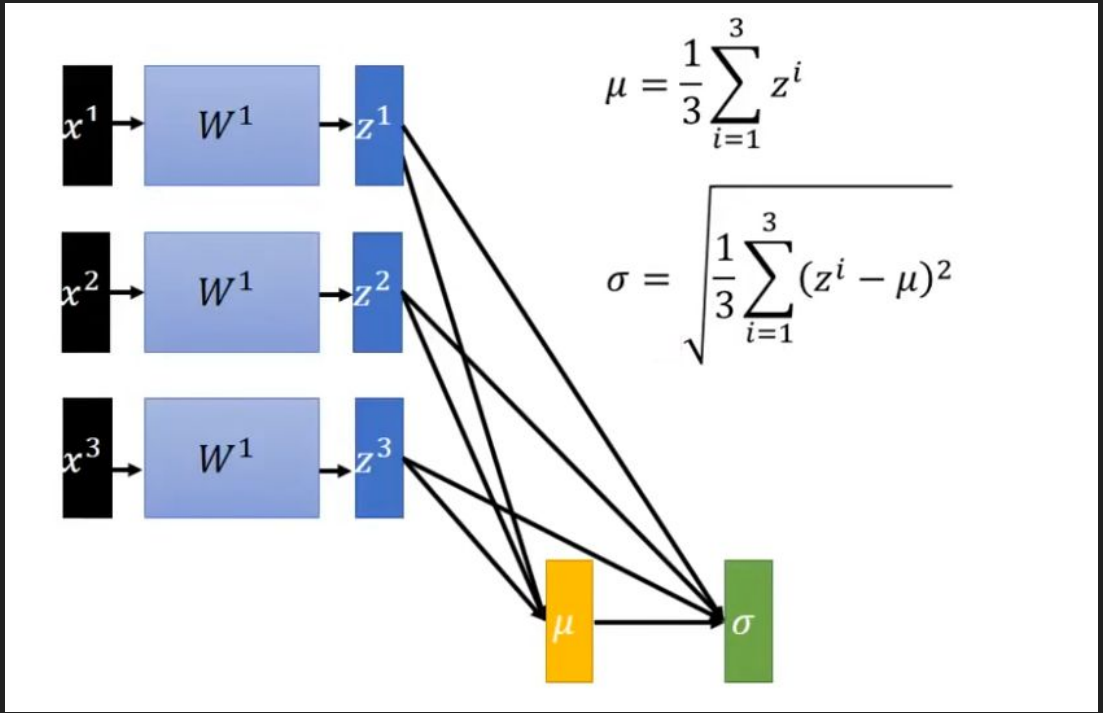


Normalization



Standardization

Batch Normalization. Training Stage.



Original paper: https://arxiv.org/abs/1502.03167

Batch Normalization. Training Stage.

$$\mu = \frac{1}{m}\sum_i z_i$$

$$\sigma = \frac{1}{m}\sqrt{\sum_i (z_i - \mu)^2}$$

$$z_i^{norm} = \frac{z_i - \mu}{\sigma + \epsilon}$$

$$\widetilde{z}_i = \gamma z_i^{norm} + \beta$$

Batch Normalization. Training Stage.

What are *mu* and *sigma* after training is done?

Batch Norm also keeps a running count of the **Exponential Moving Average (EMA)** of the mean and variance. During training, it simply calculates this EMA but does not do anything with it. At the end of training, it simply saves this value as part of the layer's state, for use during the Inference phase.

Batch Normalization. Training Stage. EMA

$$\mu = \frac{1}{m}\sum_i z_i$$

$$\overline{\mu_0} = 0$$

$$\overline{\mu_1} = 0.9\overline{\mu_0} + 0.1\mu$$

$$\overline{\mu_2} = 0.9\overline{\mu_1} + 0.1\mu$$

$$\overline{\mu_3} = 0.9\overline{\mu_2} + 0.1\mu$$

$$\dots$$

$$\overline{\mu_n} = 0.9\overline{\mu_{n-1}} + 0.1\mu$$

$$\sigma = \frac{1}{m}\sqrt{\sum_i (z_i - \mu)^2}$$

$$\overline{\sigma_0} = 0$$

$$\overline{\sigma_1} = 0.9\overline{\sigma_0} + 0.1\sigma$$

$$\overline{\sigma_2} = 0.9\overline{\sigma_1} + 0.1\sigma$$

$$\overline{\sigma_3} = 0.9\overline{\sigma_2} + 0.1\sigma$$

$$\dots$$

$$\overline{\sigma_n} = 0.9\overline{\sigma_{n-1}} + 0.1\sigma$$

Batch Normalization Benefits

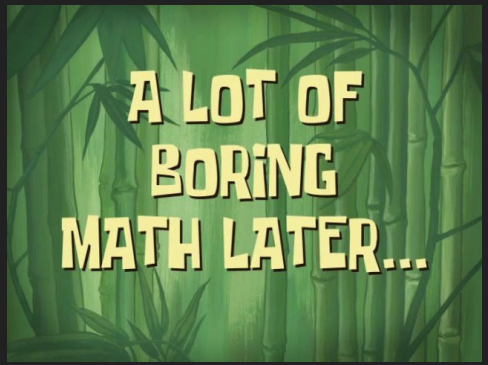
1. Adds some noise to NN (acts as regularization)
2. Training is much more smoother and stable (no big number multiplication)

Batch Normalization. Inference Stage.

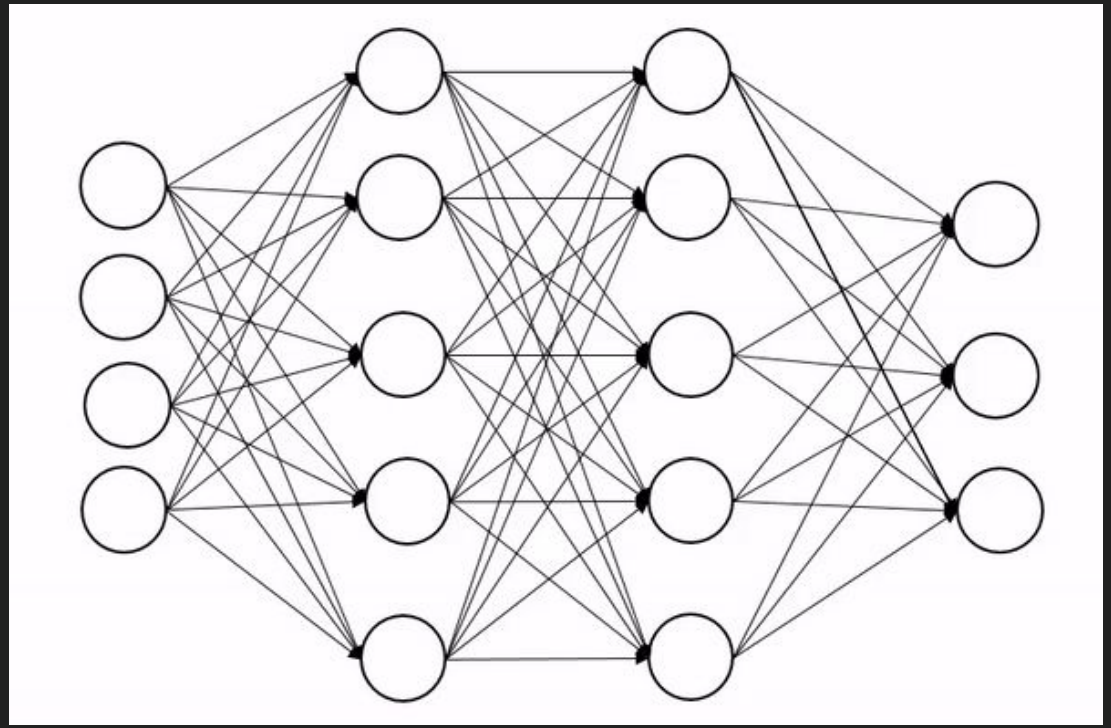
$$\overline{\mu_n} = 0.9\overline{\mu_{n-1}} + 0.1\mu$$
$$\overline{\sigma_n} = 0.9\overline{\sigma_{n-1}} + 0.1\sigma$$

$$z_i^{norm} = \frac{z_i - \overline{\mu_n}}{\overline{\sigma_n} + \epsilon}$$

$$\widetilde{z}_i = \gamma z_i^{norm} + \beta$$

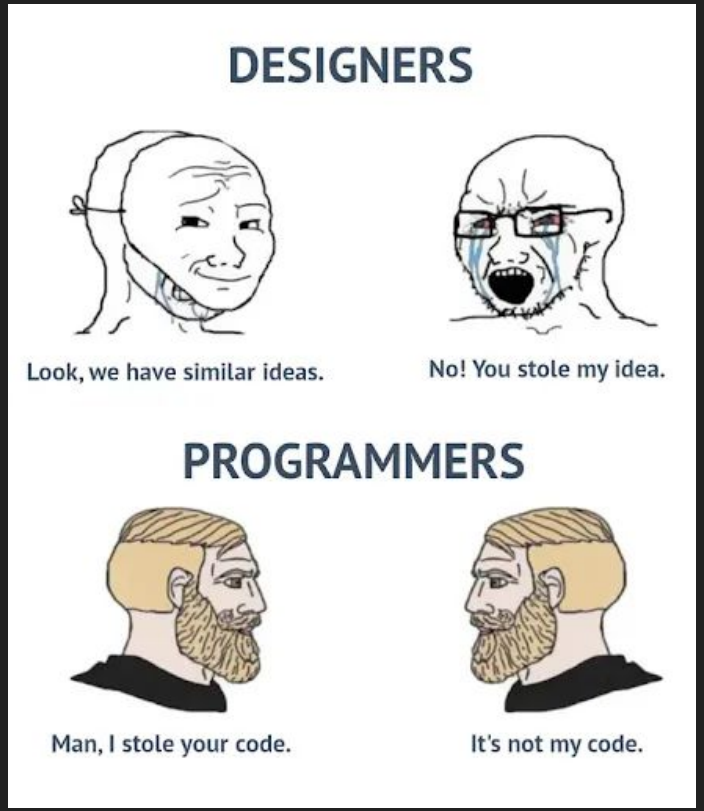


Dropout

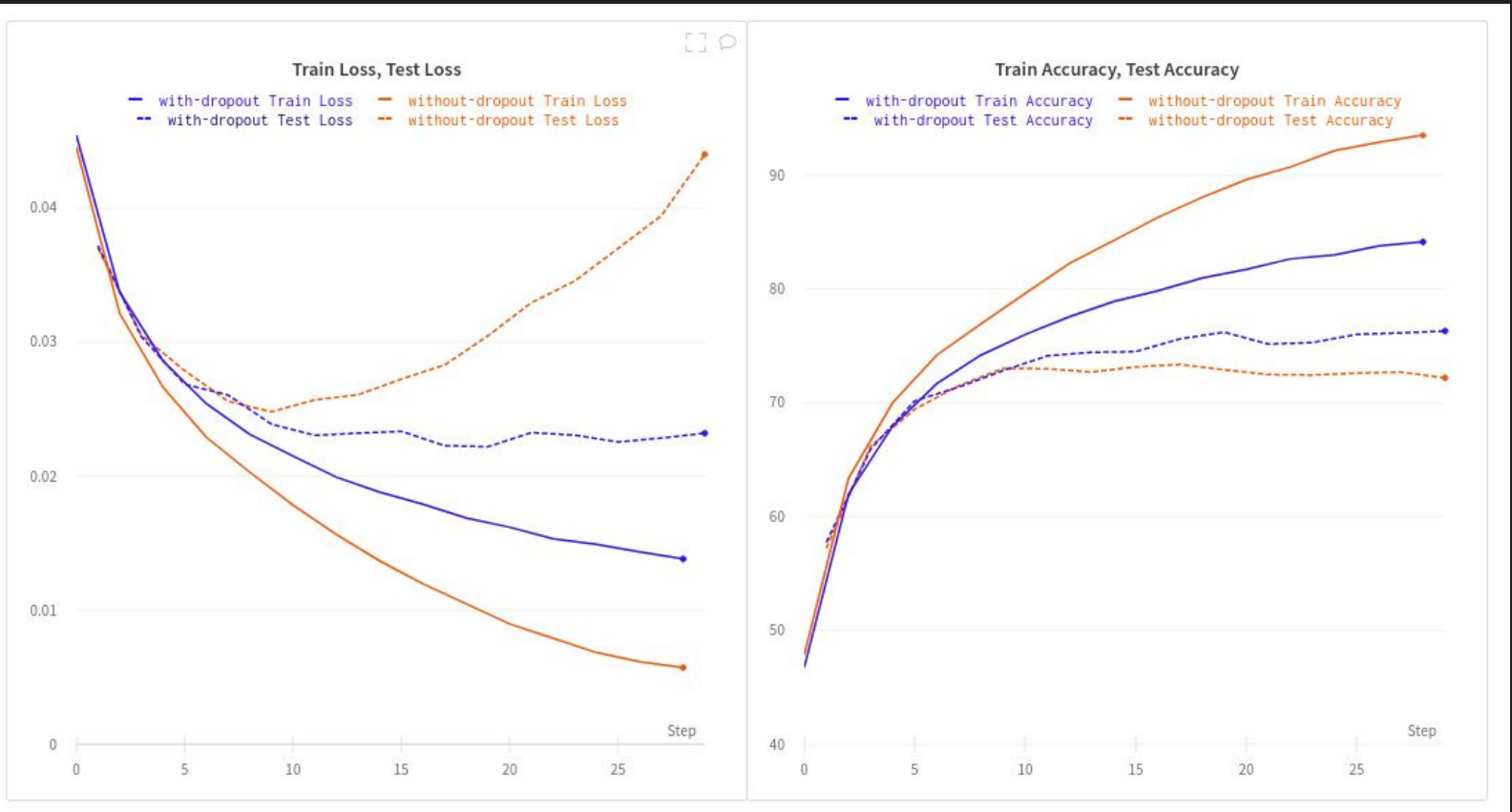


Original Paper: https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

Inverted Dropout. Demo Code



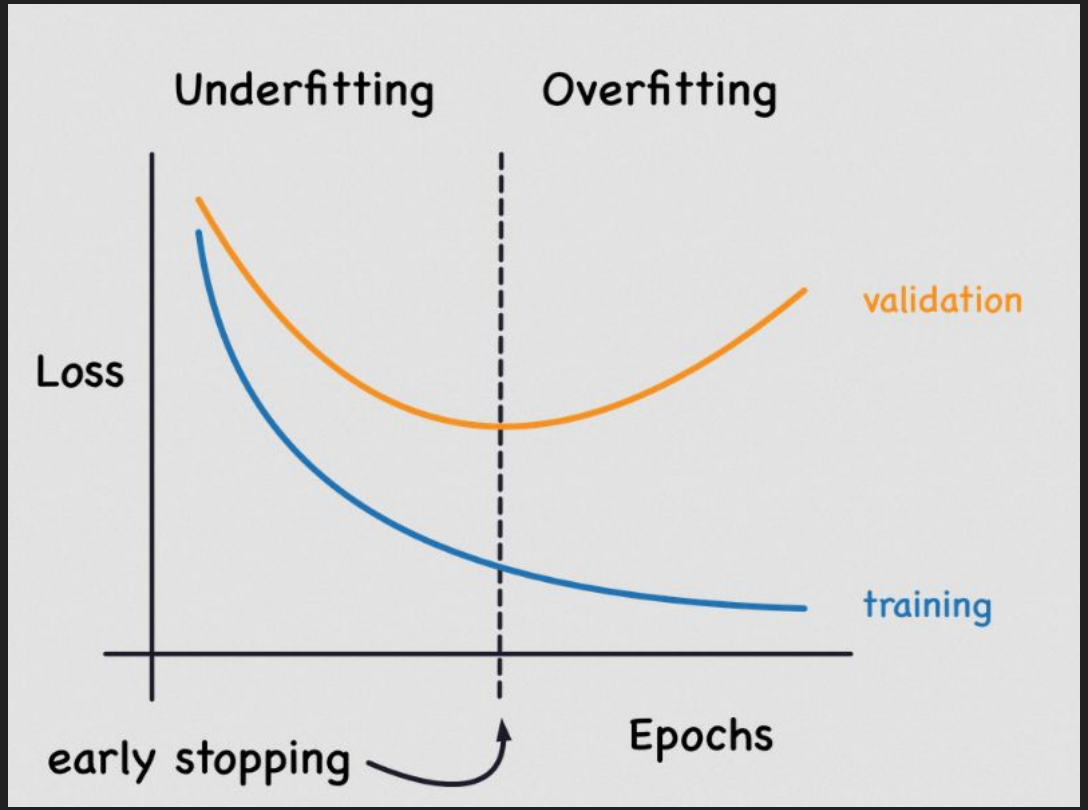
Dropout


Train Loss, Test Loss
with-dropout Train Loss
with-dropout Test Loss
without-dropout Train Loss
without-dropout Test Loss
0.04
0.03
0.02
0.01
0
Step
0
5
10
15
20
25
Train Accuracy, Test Accuracy
with-dropout Train Accuracy
with-dropout Test Accuracy
without-dropout Train Accuracy
without-dropout Test Accuracy
90
80
70
60
50
40
Step
0
5
10
15
20
25

Dropout Features

1. Dropout forces a neural network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.
2. Dropout roughly doubles the number of iterations required to converge. However, training time for each epoch is less.
3. With H hidden units, each of which can be dropped, we have
4. 2^H possible models. In testing phase, the entire network is considered and each activation is reduced by a factor *p*.

Early Stopping



HW

Experiment with multiclass classification (MLP) for MNIST data set:

- Include dropout layers
- Include batch normalization layers
- Include more layers
- Experiment with activation functions
- Experiment presence / absence of dropout and batch normalization
- Experiment with batch size
- Plot **loss = f(epochs)** for each experiment

References

1. https://towardsdatascience.com/time-series-from-scratch-exponentially-weighted-moving-averages-ewma-theory-and-implementation-607661d574fe
2. https://medium.com/codex/simple-moving-average-and-exponentially-weighted-moving-average-with-pandas-57d4a457d363
3. https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739
4. https://deepnotes.io/dropout