

# Cheat Sheet – Bias-Variance Tradeoff

## What is Bias?

$$bias = \mathbb{E}[f'(x)] - f(x)$$

- Error between average model prediction and ground truth
- The bias of the estimated function tells us the capacity of the underlying model to predict the values

## What is Variance?

$$variance = \mathbb{E}[(f'(x) - \mathbb{E}[f'(x)])^2]$$

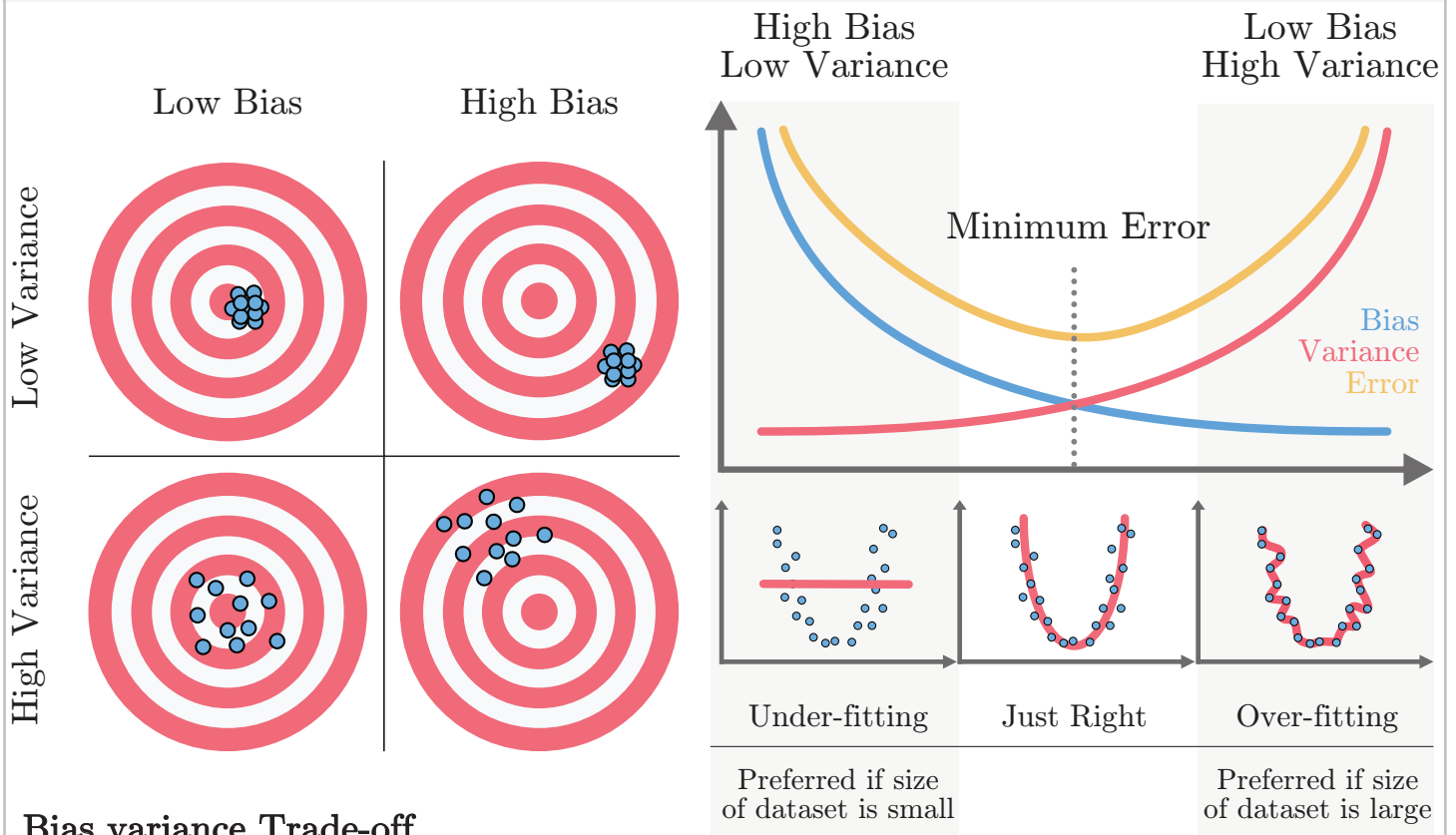
- Average variability in the model prediction for the given dataset
- The variance of the estimated function tells you how much the function can adjust to the change in the dataset

### High Bias

- Overly-simplified Model
- Under-fitting
- High error on both test and train data

### High Variance

- Overly-complex Model
- Over-fitting
- Low error on train data and high on test
- Starts modelling the noise in the input



## Bias variance Trade-off

- Increasing bias reduces variance and vice-versa
- $Error = bias^2 + variance + irreducible\ error$
- The best model is where the error is reduced.
- Compromise between bias and variance

# Cheat Sheet – Imbalanced Data in Classification



## Accuracy doesn't always give the correct insight about your trained model

<b>Accuracy:</b> %age correct prediction	Correct prediction over total predictions	One value for entire network
<b>Precision:</b> <u>Exactness</u> of model	From the detected cats, how many were actually cats	Each class/label has a value
<b>Recall:</b> <u>Completeness</u> of model	Correctly detected cats over total cats	Each class/label has a value
<b>F1 Score:</b> Combines Precision/Recall	Harmonic mean of Precision and Recall	Each class/label has a value

## Performance metrics associated with Class 1

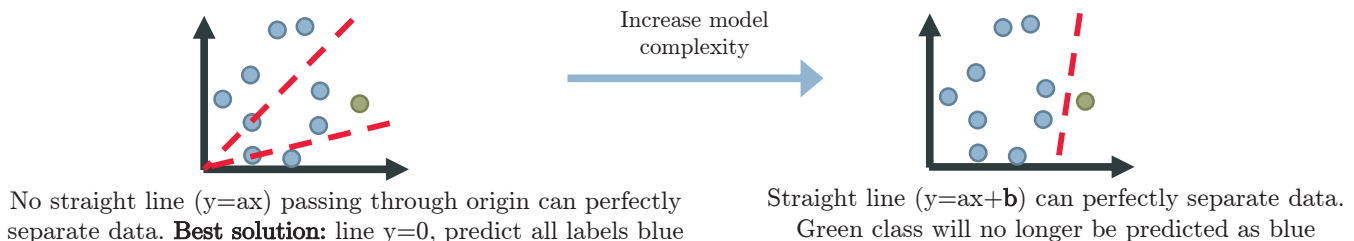
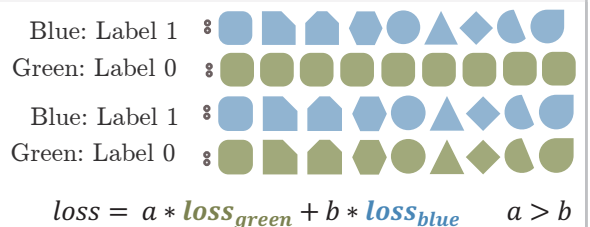
		Actual Labels		(Is your prediction correct?) (What did you predict)	
		1	0	True	Negative
Predicted Labels	1	True Positive	False Positive	(Your prediction is <u>correct</u> )	(You predicted <u>0</u> )
	0	False Negative	True Negative		

$\text{Precision} = \frac{TP}{TP + FP}$ $\text{F1 score} = 2 \times \frac{(\text{Prec} \times \text{Rec})}{(\text{Prec} + \text{Rec})}$ $\text{Specificity} = \frac{TN}{TN + FP}$	$\text{False +ve rate} = \frac{FP}{TN + FP}$ $\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$ $\text{Recall, Sensitivity} = \frac{TP}{TP + FN}$ <p>True +ve rate</p>
---	---

## Possible solutions

- Data Replication:** Replicate the available data until the number of samples are comparable
- Synthetic Data:** Images: Rotate, dilate, crop, add noise to existing input images and create new data
- Modified Loss:** Modify the loss to reflect greater error when misclassifying smaller sample set
- Change the algorithm:** Increase the model/algorithm complexity so that the two classes are perfectly separable (Con: Overfitting)



# Cheat Sheet – PCA Dimensionality Reduction

## What is PCA?

- Based on the dataset find a new set of orthogonal feature vectors in such a way that the data spread is maximum in the direction of the feature vector (or dimension)
- Rates the feature vector in the decreasing order of data spread (or variance)
- The datapoints have maximum variance in the first feature vector, and minimum variance in the last feature vector
- The variance of the datapoints in the direction of feature vector can be termed as a measure of information in that direction.

## Steps

1. Standardize the datapoints
2. Find the covariance matrix from the given datapoints
3. Carry out eigen-value decomposition of the covariance matrix
4. Sort the eigenvalues and eigenvectors

$$X_{new} = \frac{X - \text{mean}(X)}{\text{std}(X)}$$

$$C[i, j] = \text{cov}(x_i, x_j)$$

$$C = V \Sigma V^{-1}$$

$$\Sigma_{\text{sort}} = \text{sort}(\Sigma) \quad V_{\text{sort}} = \text{sort}(V, \Sigma_{\text{sort}})$$

## Dimensionality Reduction with PCA

- Keep the first m out of n feature vectors rated by PCA. These m vectors will be the best m vectors preserving the maximum information that could have been preserved with m vectors on the given dataset

## Steps:

1. Carry out steps 1-4 from above
2. Keep first m feature vectors from the sorted eigenvector matrix
3. Transform the data for the new basis (feature vectors)
4. The importance of the feature vector is proportional to the magnitude of the eigen value

$$V_{\text{reduced}} = V[:, 0 : m]$$

$$X_{\text{reduced}} = X_{\text{new}} \times V_{\text{reduced}}$$

Figure 1

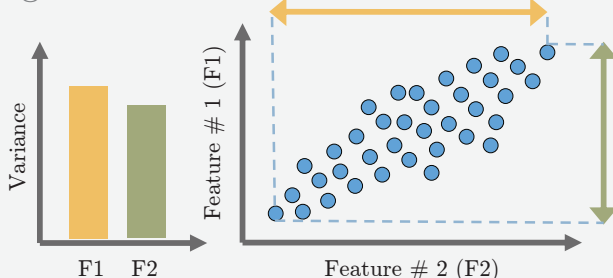


Figure 2

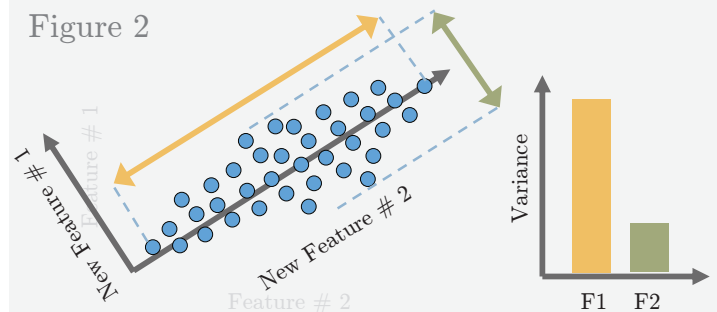
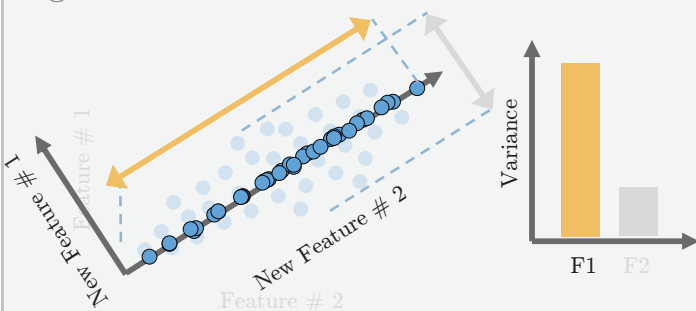


Figure 3



**Figure 1:** Datapoints with feature vectors as x and y-axis

**Figure 2:** The cartesian coordinate system is rotated to maximize the standard deviation along any one axis (new feature # 2)

**Figure 3:** Remove the feature vector with minimum standard deviation of datapoints (new feature # 1) and project the data on new feature # 2

# Cheat Sheet – Bayes Theorem and Classifier

## What is Bayes' Theorem?

- Describes the probability of an event, based on prior knowledge of conditions that might be related to the event.

$$P(A|B) = \frac{P(B|A)(\text{likelihood}) \times P(A)(\text{prior})}{P(B)(\text{prior})}$$

- How the probability of an event changes when we have knowledge of another event

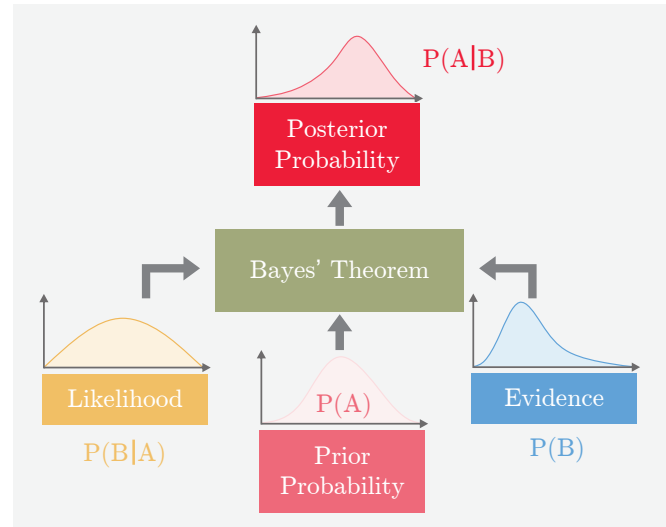
$$P(A) \longrightarrow P(A|B)$$

Usually a better estimate than  $P(A)$

## Example

- Probability of fire  $P(F) = 1\%$
- Probability of smoke  $P(S) = 10\%$
- Prob of smoke given there is a fire  $P(S|F) = 90\%$
- What is the probability that there is a fire given we see a smoke  $P(F|S)$ ?

$$P(F|S) = \frac{P(S|F) \times P(F)}{P(S)} = \frac{0.9 \times 0.01}{0.1} = 9\%$$



## Maximum A Posteriori Probability (MAP) Estimation

The MAP estimate of the random variable  $y$ , given that we have observed iid  $(x_1, x_2, x_3, \dots)$ , is given by. We try to accommodate our prior knowledge when estimating.

$$\hat{y}_{MAP} = \underset{y}{\operatorname{argmax}} P(y) \prod_i P(x_i|y)$$

$y$  that maximizes the product of **prior** and **likelihood**

## Maximum Likelihood Estimation (MLE)

The MLE estimate of the random variable  $y$ , given that we have observed iid  $(x_1, x_2, x_3, \dots)$ , is given by. We assume we don't have any prior knowledge of the quantity being estimated.

$$\hat{y}_{MLE} = \underset{y}{\operatorname{argmax}} \prod_i P(x_i|y)$$

$y$  that maximizes only the **likelihood**

MLE is a special case of MAP where our prior is uniform (all values are equally likely)

## Naïve Bayes' Classifier (Instantiation of MAP as classifier)

Suppose we have two classes,  $y=y_1$  and  $y=y_2$ . Say we have more than one evidence/features  $(x_1, x_2, x_3, \dots)$ , using Bayes' theorem

$$P(y|x_1, x_2, x_3, \dots) = \frac{P(x_1, x_2, x_3, \dots|y) \times P(y)}{P(x_1, x_2, x_3, \dots)}$$

Bayes' theorem assumes the features  $(x_1, x_2, x_3, \dots)$  are i.i.d. i.e.  $P(x_1, x_2, x_3, \dots|y) = \prod_i P(x_i|y)$

$$P(y|x_1, x_2, x_3, \dots) = \prod_i P(x_i|y) \frac{P(y)}{P(x_1, x_2, x_3, \dots)}$$

$$\hat{y} = y_1 \text{ if } \frac{P(y_1|x_1, x_2, x_3, \dots)}{P(y_2|x_1, x_2, x_3, \dots)} > 1 \text{ else } \hat{y} = y_2$$

# Cheat Sheet – Regression Analysis

## What is Regression Analysis?

Fitting a function  $f(\cdot)$  to datapoints  $y_i=f(x_i)$  under some error function. Based on the estimated function and error, we have the following types of regression

### 1. Linear Regression:

Fits a **line** minimizing the **sum of mean-squared error** for each datapoint.

$$\min_{\beta} \sum_i \|y_i - f_{\beta}^{linear}(x_i)\|^2$$

$$f_{\beta}^{linear}(x_i) = \beta_0 + \beta_1 x_i$$

### 2. Polynomial Regression:

Fits a **polynomial** of order  $k$  ( $k+1$  unknowns) minimizing the **sum of mean-squared error** for each datapoint.

$$\min_{\beta} \sum_{i=0}^m \|y_i - f_{\beta}^{poly}(x_i)\|^2$$

$$f_{\beta}^{poly}(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \beta_k x_i^k$$

### 3. Bayesian Regression:

For each datapoint, fits a **gaussian distribution** by minimizing the **mean-squared error**. As the number of data points  $x_i$  increases, it converges to point estimates i.e.  $n \rightarrow \infty, \sigma^2 \rightarrow 0$

$$\min_{\beta} \sum_i \|y_i - \mathcal{N}(f_{\beta}(x_i), \sigma^2)\|^2$$

$$f_{\beta}(x_i) = f_{\beta}^{poly}(x_i) \text{ or } f_{\beta}^{linear}(x_i)$$

$$\mathcal{N}(\mu, \sigma^2) \rightarrow \text{Gaussian with mean } \mu \text{ and variance } \sigma^2$$

### 4. Ridge Regression:

Can fit either a **line, or polynomial** minimizing the **sum of mean-squared error** for each datapoint and the **weighted L2 norm** of the function parameters beta.

$$\min_{\beta} \sum_{i=0}^m \|y_i - f_{\beta}(x_i)\|^2 + \sum_{j=0}^k \beta_j^2$$

$$f_{\beta}(x_i) = f_{\beta}^{poly}(x_i) \text{ or } f_{\beta}^{linear}(x_i)$$

### 5. LASSO Regression:

Can fit either a **line, or polynomial** minimizing the **sum of mean-squared error** for each datapoint and the **weighted L1 norm** of the function parameters beta.

$$\min_{\beta} \sum_{i=0}^m \|y_i - f_{\beta}(x_i)\|^2 + \sum_{j=0}^k |\beta_j|$$

$$f_{\beta}(x_i) = f_{\beta}^{poly}(x_i) \text{ or } f_{\beta}^{linear}(x_i)$$

### 6. Logistic Regression (NOT regression, but classification):

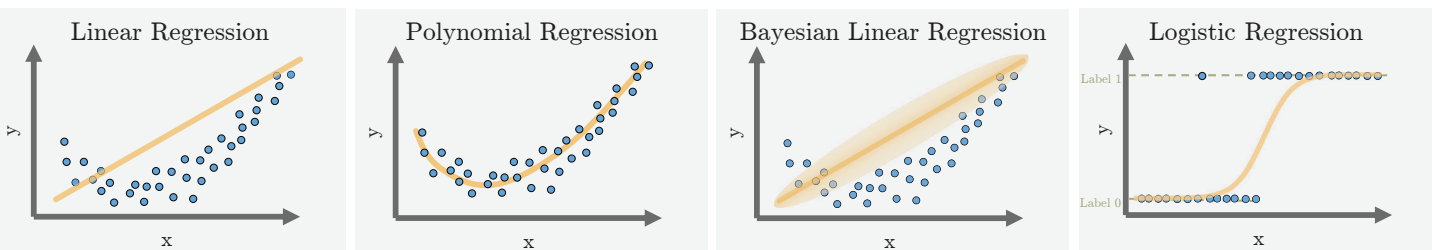
Can fit either a **line, or polynomial with sigmoid activation** minimizing the **sum of mean-squared error** for each datapoint. The labels  $y$  are binary class labels.

$$\min_{\beta} \sum_i \|y_i - \sigma(f_{\beta}(x_i))\|^2$$

$$f_{\beta}(x_i) = f_{\beta}^{poly}(x_i) \text{ or } f_{\beta}^{linear}(x_i)$$

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

## Visual Representation:



## Summary:

	What does it fit?	Estimated function	Error Function
Linear	A line in $n$ dimensions	$f_{\beta}^{linear}(x_i) = \beta_0 + \beta_1 x_i$	$\sum_{i=0}^m \ y_i - f_{\beta}(x_i)\ ^2$
Polynomial	A polynomial of order $k$	$f_{\beta}^{poly}(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots$	$\sum_{i=0}^m \ y_i - f_{\beta}(x_i)\ ^2$
Bayesian Linear	Gaussian distribution for each point	$\mathcal{N}(f_{\beta}(x_i), \sigma^2)$	$\sum_{i=0}^m \ y_i - \mathcal{N}(f_{\beta}(x_i), \sigma^2)\ ^2$
Ridge	Linear/polynomial	$f_{\beta}^{poly}(x_i) \text{ or } f_{\beta}^{linear}(x_i)$	$\sum_{i=0}^m \ y_i - f_{\beta}(x_i)\ ^2 + \sum_{j=0}^n \beta_j^2$
LASSO	Linear/polynomial	$f_{\beta}^{poly}(x_i) \text{ or } f_{\beta}^{linear}(x_i)$	$\sum_{i=0}^m \ y_i - f_{\beta}(x_i)\ ^2 + \sum_{j=0}^n  \beta_j $
Logistic	Linear/polynomial with sigmoid	$\sigma(f_{\beta}(x_i))$	$\sum_{i=0}^m \ y_i - f_{\beta}(x_i)\ ^2$

# Cheat Sheet – Regularization in ML

## What is Regularization in ML?

- Regularization is an approach to address over-fitting in ML.
- Overfitted model fails to generalize estimations on test data
- When the underlying model to be learned is low bias/high variance, or when we have small amount of data, the estimated model is prone to over-fitting.
- Regularization reduces the variance of the model

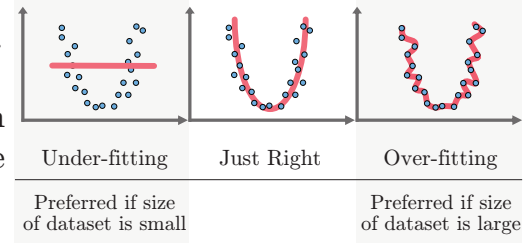


Figure 1. Overfitting

## Types of Regularization:

### 1. Modify the loss function:

- **L2 Regularization:** Prevents the weights from getting too large (defined by L2 norm). Larger the weights, more complex the model is, more chances of overfitting.

$$loss = error(y, \hat{y}) + \lambda \sum_j \beta_j^2 \quad \lambda \geq 0, \lambda \propto model\ bias, \quad \lambda \propto \frac{1}{model\ variance}$$

- **L1 Regularization:** Prevents the weights from getting too large (defined by L1 norm). Larger the weights, more complex the model is, more chances of overfitting. L1 regularization introduces sparsity in the weights. It forces more weights to be zero, than reducing the the average magnitude of all weights

$$loss = error(y, \hat{y}) + \lambda \sum_j |\beta_j| \quad \lambda \geq 0, \lambda \propto model\ bias, \quad \lambda \propto \frac{1}{model\ variance}$$

- **Entropy:** Used for the models that output probability. Forces the probability distribution towards uniform distribution.

$$loss = error(p, \hat{p}) - \lambda \sum_i \hat{p}_i \log(\hat{p}_i) \quad \lambda \geq 0, \lambda \propto model\ bias, \quad \lambda \propto \frac{1}{model\ variance}$$

### 2. Modify data sampling:

- **Data augmentation:** Create more data from available data by randomly cropping, dilating, rotating, adding small amount of noise etc.
- **K-fold Cross-validation:** Divide the data into k groups. Train on (k-1) groups and test on 1 group. Try all k possible combinations.

### 3. Change training approach:

- **Injecting noise:** Add random noise to the weights when they are being learned. It pushes the model to be relatively insensitive to small variations in the weights, hence regularization
- **Dropout:** Generally used for neural networks. Connections between consecutive layers are randomly dropped based on a dropout-ratio and the remaining network is trained in the current iteration. In the next iteration, another set of random connections are dropped.

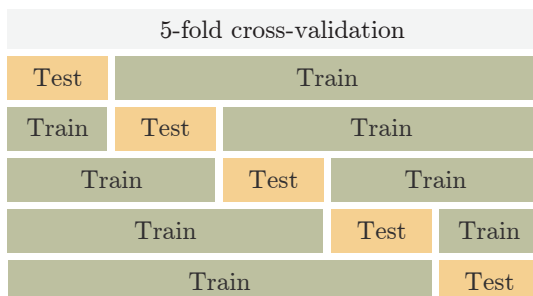


Figure 2. K-fold CV

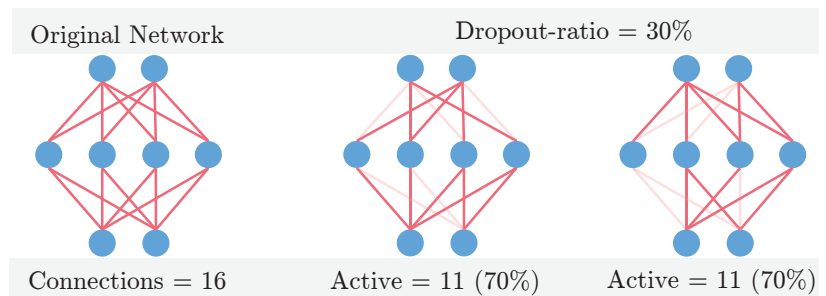


Figure 3. Drop-out



# Cheat Sheet – Famous CNNs

## AlexNet – 2012

**Why:** AlexNet was born out of the need to improve the results of the ImageNet challenge.

**What:** The network consists of 5 Convolutional (CONV) layers and 3 Fully Connected (FC) layers. The activation used is the Rectified Linear Unit (ReLU).

**How:** Data augmentation is carried out to reduce over-fitting, Uses Local response localization.

AlexNet Network - Structural Details											
Input	Output	Layer	Stride	Pad	Kernel	in	out	# of Param			
227 227 3	55 55 96	conv1	4	0	11 11	3	96	34944			
55 55 96	27 27 27	maxpool1	2	0	3 3	3	96	0			
27 27 96	27 27 256	conv2	1	2	5 5	96	256	614656			
27 27 256	13 13 256	maxpool2	2	0	3 3	3	256	256			
13 13 256	13 13 384	conv3	1	1	3 3	3	256	384			
13 13 384	13 13 384	conv4	1	1	3 3	3	384	384			
13 13 384	13 13 256	conv5	1	1	3 3	3	384	256			
13 13 256	6 6 256	maxpool5	2	0	3 3	3	256	256			
		fc6			1	1	9216	4096			
		fc7			1	1	4096	4096			
		fc8			1	1	4096	1000			
		Total							62,378,344		

## VGGNet – 2014

**Why:** VGGNet was born out of the need to reduce the # of parameters in the CONV layers and improve on training time

**What:** There are multiple variants of VGGNet (VGG16, VGG19, etc.)

**How:** The important point to note here is that all the conv kernels are of size 3x3 and maxpool kernels are of size 2x2 with a stride of two.

VGG16 - Structural Details											
#	Input Image	output	Layer	Stride	Kernel	in	out	Param			
1	224 224 64	224 224 64	conv3064	1	3 3	64	64	1792			
2	224 224 64	112 112 64	maxpool	2	2 2	64	64	36928			
3	112 112 64	112 112 128	conv3-128	1	3 3	64	128	73856			
4	112 112 128	56 56 128	maxpool	2	2 2	128	128	147584			
5	56 56 128	56 56 256	conv3-256	1	3 3	128	256	295168			
6	56 56 256	56 56 256	conv3-256	1	3 3	256	256	590080			
7	56 56 256	56 56 256	conv3-256	1	3 3	256	256	590080			
8	28 28 256	28 28 512	conv3-512	1	3 3	256	512	1180160			
9	28 28 512	28 28 512	conv3-512	1	3 3	512	512	2359808			
10	28 28 512	14 14 512	maxpool	2	2 2	512	512	0			
11	14 14 512	14 14 512	conv3-512	1	3 3	512	512	2359808			
12	14 14 512	14 14 512	conv3-512	1	3 3	512	512	2359808			
13	14 14 512	7 7 512	conv3-512	1	3 3	512	512	2359808			
14	14 14 512	7 7 512	maxpool	2	2 2	512	512	0			
15	1 1 25088	1 1 4096	fc			1	25088	4096	102764544		
16	1 1 4096	1 1 4096	fc			1	4096	4096	16781312		
17	1 1 4096	1 1 1000	fc			1	4096	1000	4097000		
		Total							138,423,208		

## ResNet – 2015

**Why:** Neural Networks are notorious for not being able to find a simpler mapping when it exists. ResNet solves that.

**What:** There are multiple versions of ResNetXX architectures where 'XX' denotes the number of layers. The most used ones are ResNet50 and ResNet101. Since the vanishing gradient problem was taken care of (more about it in the How part), CNN started to get deeper and deeper

**How:** ResNet architecture makes use of shortcut connections to solve the vanishing gradient problem. The basic building block of ResNet is a Residual block that is repeated throughout the network.

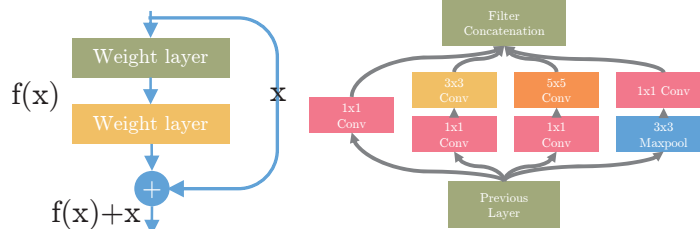


Figure 1 ResNet Block

Figure 2 Inception Block

## Inception – 2014

**Why:** Larger kernels are preferred for more global features, on the other hand, smaller kernels provide good results in detecting area-specific features. For effective recognition of such a variable-sized feature, we need kernels of different sizes. That is what Inception does.

**What:** The Inception network architecture consists of several inception modules of the following structure. Each inception module consists of four operations in parallel, 1x1 conv layer, 3x3 conv layer, 5x5 conv layer, max pooling

**How:** Inception increases the network space from which the best network is to be chosen via training. Each inception module can capture salient features at different levels.

Comparison				
Network	Year	Salient Feature	top5 accuracy	Parameters FLOP
AlexNet	2012	Deeper	84.70%	62M 1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M 19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M 2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M 11B

ResNet18 - Structural Details											
#	Input Image	output	Layer	Stride	Pad	Kernel	in	out	Param		
1	227 227 3	112 112 64	conv1	2	1	7 7 3	64	9472			
2	112 112 64	56 56 64	maxpool	1	1	3 3	64	0			
3	56 56 64	56 56 64	conv2-1	1	1	3 3	64	36928			
4	56 56 64	56 56 64	conv2-2	1	1	3 3	64	36928			
5	56 56 64	56 56 64	conv2-3	1	1	3 3	64	36928			
6	56 56 64	28 28 128	conv3-1	2	0.5	3 3	64	128	73856		
7	28 28 128	28 28 128	conv3-2	1	1	3 3	128	128	147584		
8	28 28 128	28 28 128	conv3-3	1	1	3 3	128	128	147584		
9	28 28 128	28 28 128	conv3-4	1	1	3 3	128	128	147584		
10	28 28 128	14 14 256	conv4-1	2	0.5	3 3	128	256	295168		
11	14 14 256	14 14 256	conv4-2	1	1	3 3	256	256	590080		
12	14 14 256	14 14 256	conv4-3	1	1	3 3	256	256	590080		
13	14 14 256	14 14 256	conv4-4	1	1	3 3	256	256	590080		
14	14 14 256	7 7 512	conv5-1	2	0.5	3 3	256	512	1180160		
15	7 7 512	7 7 512	conv5-2	1	1	3 3	512	512	2359808		
16	7 7 512	7 7 512	conv5-3	1	1	3 3	512	512	2359808		
17	7 7 512	7 7 512	conv5-4	1	1	3 3	512	512	2359808		
18	7 7 512	1 1 512	avg pool	7	0	7 7	512	512	0		
19	1 1 512	1 1 1000	fc				512	1000	513000		
Total									11,511,784		

GoogLeNet - Structural Details											
Input Image	output	Layer	Input Layer	Stride	Pad	Kernel	in	out	Param		
227 227 3	112 112 64	conv1	input	2	0.5	3 3	64	64	9472		
112 112 64	56 56 64	maxpool	conv1	1	0	1 1	64	64	0		
56 56 64	56 56 64	conv2-1	maxpool1	1	0	1 1	64	64	1180		
56 56 192	28 28 192	conv2-2	maxpool2	2	0.5	3 3	192	192	0		
Inception (3x)	28 28 192	28 28 96	conv3-1a	maxpool2	1	0	1 1	192	96	18528	
	28 28 96	28 28 16	conv3-1b	conv3-1a	1	0	1 1	192	16	3088	
	28 28 192	28 28 192	conv3-2a	conv3-1a	1	1	3 3	192	192	0	
	28 28 192	28 28 64	conv3-2b	conv3-1a	1	0	1 1	192	64	12832	
	28 28 96	28 28 128	conv3-2c	conv3-1a	1	0	1 1	192	128	110720	
	28 28 16	28 28 32	conv3-2d	conv3-1a	1	2	5 5	16	32	12832	
	28 28 192	28 28 256	conv3-2e	conv3-1a	1	0	1 1	192	32	6176	
		28 28 256	depth-concat	conv3-2a							
Inception (3x)	28 28 256	28 28 128	conv3-3a	depth-concat	1	0	1 1	256	128	32896	
	28 28 192	28 28 32	conv3-3b	depth-concat	1	0	1 1	256	32	8224	
	28 28 192	28 28 256	maxpool	depth-concat	1	1	3 3	256	256	0	
	28 28 192	28 28 128	conv3-4a	depth-concat	1	0	1 1	256	128	32896	
	28 28 96	28 28 192	conv3-4b	conv3-4a	1	1	3 3	128	192	221176	
	28 28 16	28 28 96	conv3-4c	conv3-4a	1	2	5 5	32	96	2896	
	28 28 192	28 28 64	conv3-4d	conv3-4a	1	0	1 1	256	64	16448	
		28 28 480	depth-concat	conv3-4a							
Inception (3x)	28 28 480	14 14 480	maxpool3	depth-concat	2	0.5	3 3	480	480	0	
	14 14 480	14 14 96	conv4-1a	maxpool3	1	0	1 1	480	96	46176	
	14 14 480	14 14 16	conv4-1b	conv4-1a	1	0	1 1	480	16	7056	
	14 14 480	14 14 480	maxpool	conv4-1a	1	1	3 3	480	480	0	
	14 14 96	14 14 192	conv5-1	maxpool3	1	1	3 3	96	192	92352	
	14 14 16	14 14 48	conv5-1b	conv5-1	1	2	5 5	16	48	19248	
	14 14 192	14 14 64	conv5-1c	conv5-1	1	0	1 1	480	64	39744	
		14 14 512	depth-concat	conv5-1c							
Inception (3x)	14 14 512	14 14 112	conv5-1d	depth-concat	1	0	1 1	512	112	57456	
	14 14 512	14 14 24	conv5-1e	depth-concat	1	0	1 1	64	24	1660	
	14 14 512	14 14 64	maxpool	depth-concat	1	1	3 3	64	64	0	
	14 14 512	14 14 128	conv5-2a	depth-concat	1	1	3 3	144	128	374336	
	14 14 16	14 14 48	conv5-2b	conv5-2a	1	2	5 5	24	48	38464	
	14 14 160	14 14 64	conv5-2c	conv5-2a	1	0	1 1	64	64	4160	
		14 14 512	depth-concat	conv5-2c							
		14 14 512	depth-concat	conv5-2c							
Inception (3x)	14 14 512	14 14 144	conv5-2d	depth-concat	1	0	1 1	512	144	73872	
	14 14 512	14 14 32	conv5-2e	depth-concat	1	0	1 1	64	32	2080	
	14 14 512	14 14 112	maxpool	depth-concat	1	1	3 3	64	112	78080	
	14 14 96	14 14 288	conv5-3	depth-concat	1	1	3 3	144	288	373536	
	14 14 16	14 14 48	conv5-3b	conv5-3	1	2	5 5	24	48	38464	
	14 14 112	14 14 64	conv5-3c	conv5-3	1	0	1 1	64	64	4160	
		14 14 512	depth-concat	conv5-3c							
		14 14 512	depth-concat	conv5-3c							
Inception (3x)	14 14 528	14 14 160	conv5-4a	depth-concat	1	0	1 1	528	160	84480	
	14 14 528	14 14 32	conv5-4b	depth-concat	1	0	1 1	64	32	2080	
	14 14 528	14 14 64	maxpool	depth-concat	1	1	3 3	64	64	0	
	14 14 528	14 14 256	conv5-4c	depth-concat	1	1	3 3	64	256	16640	
	14 14 96	14 14 288	conv5-4d	depth-concat	1	1	3 3	120	288	404400	
	14 14 16	14 14 48	conv5-4e	conv5-4d	1	2	5 5	32	48	19248	
		14 14 528	depth-concat	conv5-4e							
		14 14 892	depth-concat	conv5-4e							
Inception (3x)	14 14 892	7 7 832	maxpool4	depth-concat	2	0.5	3 3	832	832	0	
	7 7 832	7 7 160	conv5-5a	maxpool4	1	0	1 1	832	160	133280	
	7 7 832	7 7 32	conv5-5b	maxpool4	1	0	1 1	832	32	2656	
	7 7 832	7 7 832	maxpool	maxpool4	1	0	3 3	832	832	0	
	7 7 832	7 7 256	conv5-5c	maxpool4	1	0	1 1	832	256	212384	
	7 7 96	7 7 320	conv5-5d	maxpool4	1	1	3 3	160	320	461120	
	7 7 16	7 7 128	conv5-5e	conv5-5d	1	2	5 5	32	128	19248	
	7 7 256	7 7 128	conv5-5f	conv5-5d	1	0	1 1	832	128	106624	
Inception (3x)		7 7 832	depth-concat	conv5-5f							
	7 7 832	7 7 192	conv5-5g	depth-concat	1	0	1 1	832	192	159396	
	7 7 832	7 7 48	conv5-5h	depth-concat	1	0	1 1	832	48	3994	
	7 7 832	7 7 832	maxpool	depth-concat	1	1	3 3	832	832	0	
	7 7 832	7 7 256	conv5-5i	depth-concat	1	1	3 3	160	256	16640	
	7 7 96	7 7 384	conv5-5j	conv5-5i	1	1	3 3	192	384	663396	
	7 7 16	7 7 128	conv5-5k	conv5-5i	1	2	5 5	48	128	138728	
	7 7 256	7 7 128	conv5-5l	conv5-5i	1	0	1 1	832	128	106624	
Inception (3x)	7 7 1024	1 1 1024	avgpool	depth-concat	0	0	7 7	1024	1024	0	
	1 1 1024	1 1 1000	fc	depth-concat	1	0	1 1	1024	1000	1025600	
	Total						0	1	7 7	1024	1024

# Cheat Sheet – Convolutional Neural Network

## Convolutional Neural Network:

The data gets into the CNN through the input layer and passes through various hidden layers before getting to the output layer. The output of the network is compared to the actual labels in terms of loss or error. The partial derivatives of this loss w.r.t the trainable weights are calculated, and the weights are updated through one of the various methods using backpropagation.

## CNN Template:

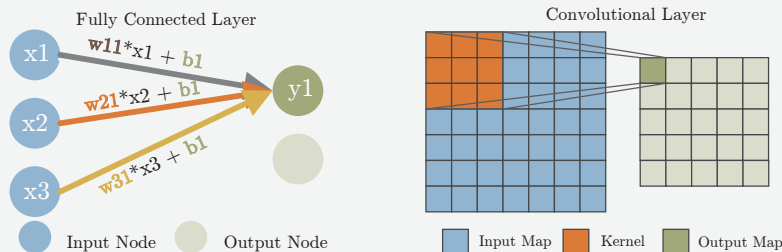
Most of the commonly used hidden layers (not all) follow a pattern

**1. Layer function:** Basic transforming function such as convolutional or fully connected layer.

**a. Fully Connected:** Linear functions between the input and the

**a. Convolutional Layers:** These layers are applied to 2D (3D) input feature maps. The trainable weights are a 2D (3D) kernel/filter that moves across the input feature map, generating dot products with the overlapping region of the input feature map.

**b. Transposed Convolutional (DeConvolutional) Layer:** Usually used to increase the size of the output feature map (Upsampling) The idea behind the transposed convolutional layer is to undo (not exactly) the convolutional layer



**2. Pooling:** Non-trainable layer to change the size of the feature map

**a. Max/Average Pooling:** Decrease the spatial size of the input layer based on selecting the maximum/average value in receptive field defined by the kernel

**b. UnPooling:** A non-trainable layer used to increase the spatial size of the input layer based on placing the input pixel at a certain index in the receptive field of the output defined by the kernel.

**3. Normalization:** Usually used just before the activation functions to limit the unbounded activation from increasing the output layer values too high

**a. Local Response Normalization LRN:** A non-trainable layer that square-normalizes the pixel values in a feature map within a local neighborhood.

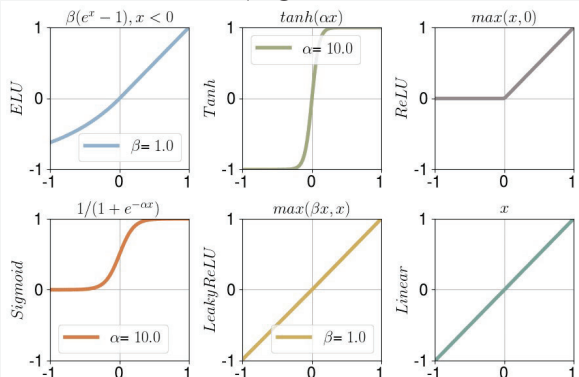
**b. Batch Normalization:** A trainable approach to normalizing the data by learning scale and shift variable during training.

**3. Activation:** Introduce non-linearity so CNN can efficiently map non-linear complex mapping.

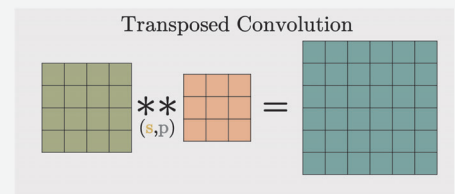
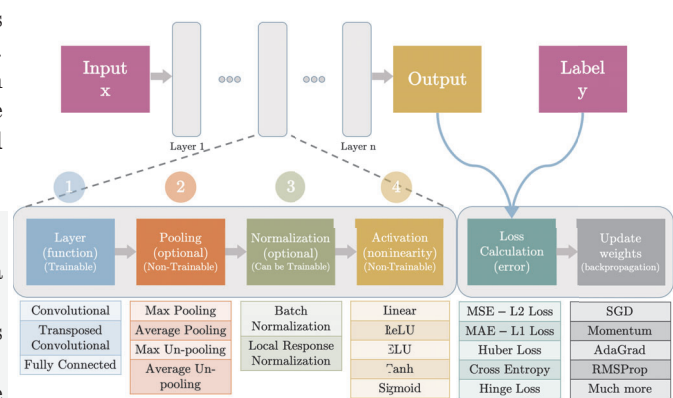
**a. Non-parametric/Static functions:** Linear, ReLU

**b. Parametric functions:** ELU, tanh, sigmoid, Leaky ReLU

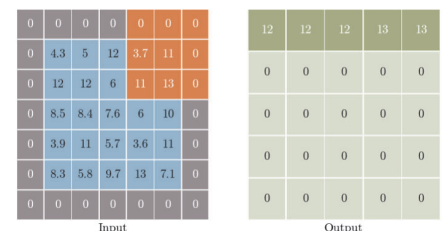
**c. Bounded functions:** tanh, sigmoid



Dataset: (x, y)



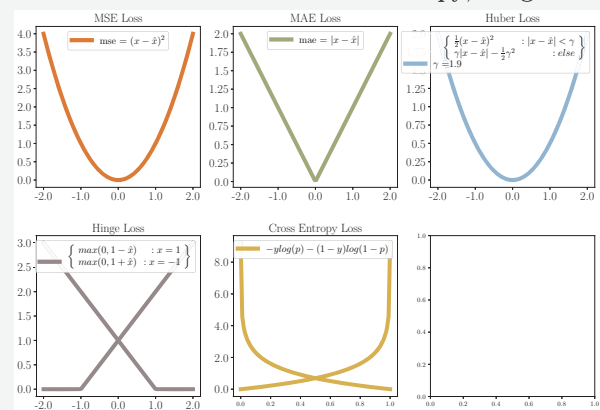
Type: max pool - Stride: 1 Padding: 1



**5. Loss function:** Quantifies how far off the CNN prediction is from the actual labels.

**a. Regression Loss Functions:** MAE, MSE, Huber loss

**b. Classification Loss Functions:** Cross entropy, Hinge loss





# Cheat Sheet – Ensemble Learning in ML

## What is Ensemble Learning? Wisdom of the crowd

Combine multiple weak models/learners into one predictive model to reduce bias, variance and/or improve accuracy.

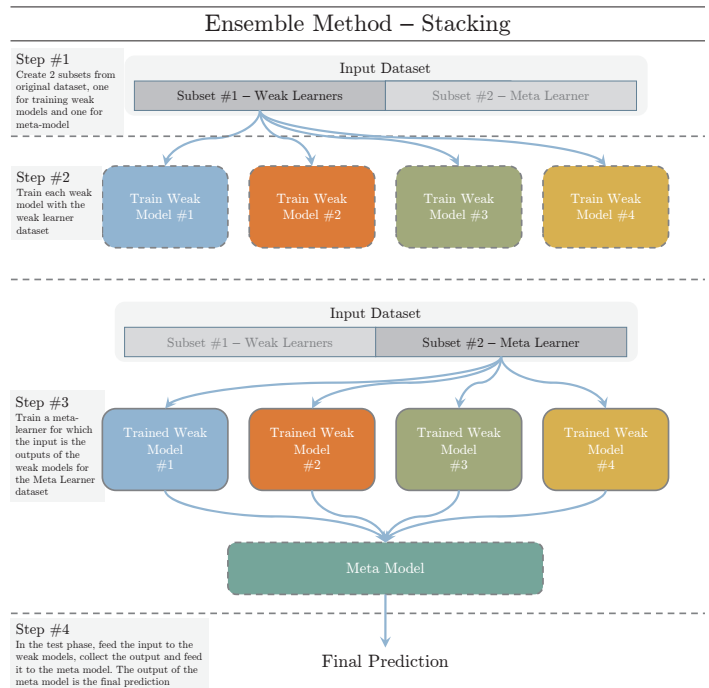
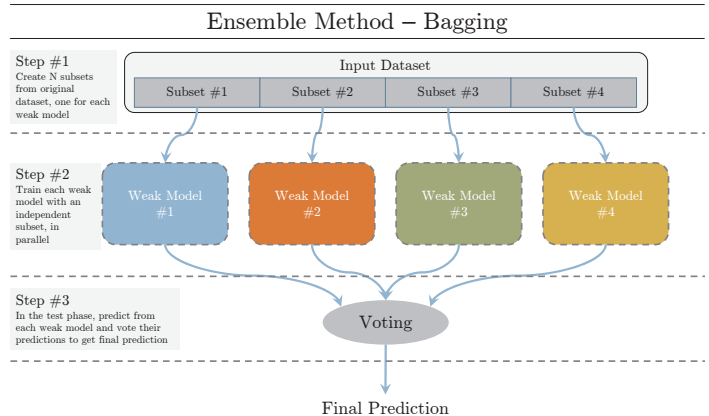
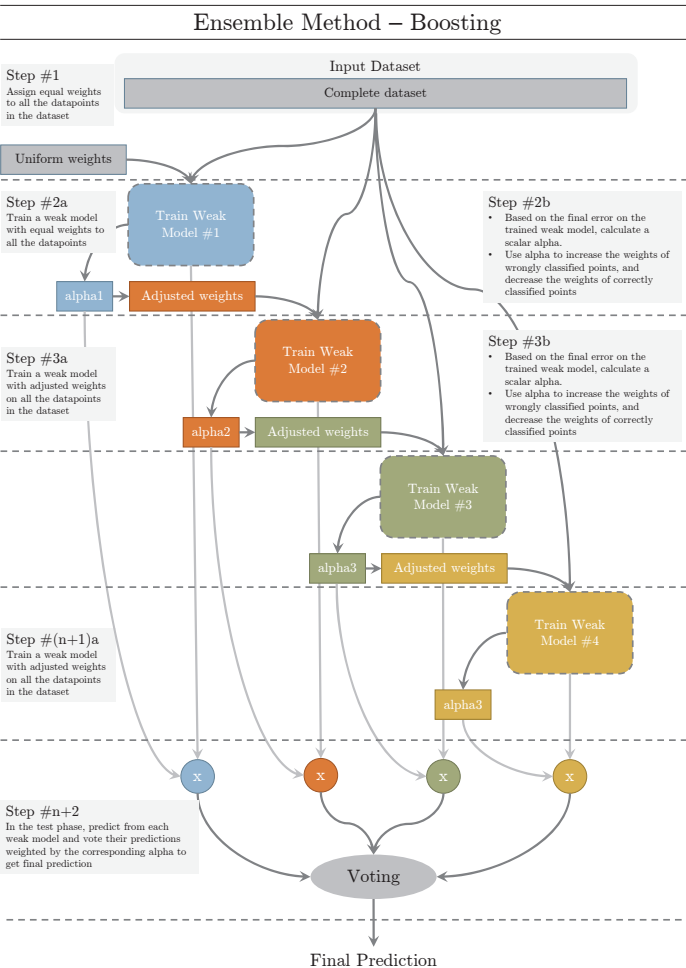
## Types of Ensemble Learning: N number of weak learners

**1. Bagging:** Trains N different weak models (usually of same types – homogenous) with N non-overlapping subset of the input dataset in parallel. In the test phase, each model is evaluated. The label with the greatest number of predictions is selected as the prediction. Bagging methods reduces variance of the prediction

**2. Boosting:** Trains N different weak models (usually of same types – homogenous) with the complete dataset in a sequential order. The datapoints wrongly classified with previous weak model is provided more weights to that they can be classified by the next weak learner properly. In the test phase, each model is evaluated and based on the test error of each weak model, the prediction is weighted for voting. Boosting methods decreases the bias of the prediction.

**3. Stacking:** Trains N different weak models (usually of different types – heterogenous) with one of the two subsets of the dataset in parallel. Once the weak learners are trained, they are used to trained a meta learner to combine their predictions and carry out final prediction using the other subset. In test phase, each model predicts its label, these set of labels are fed to the meta learner which generates the final prediction.

The block diagrams, and comparison table for each of these three methods can be seen below.



Parameter	Bagging	Boosting	Stacking
Focuses on	Reducing variance	Reducing bias	Improving accuracy
Nature of weak learners is	Homogenous	Homogenous	Heterogenous
Weak learners are aggregated by	Simple voting	Weighted voting	Learned voting (meta-learner)

Source: <https://www.cheatsheets.aqeel-anwar.com>

