Cheat Sheets

Machine Learning Interview Cheat sheets

Last Updated: March 2021

This document contains cheat sheets on various topics asked during a Machine Learning/Data science interview. This document is constantly updated to include more topics.

Table of Contents

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

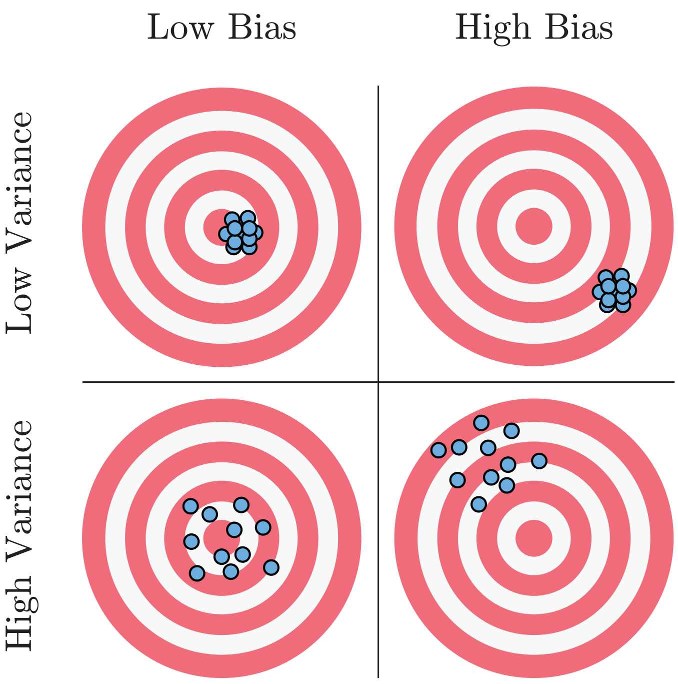
What is Variance?

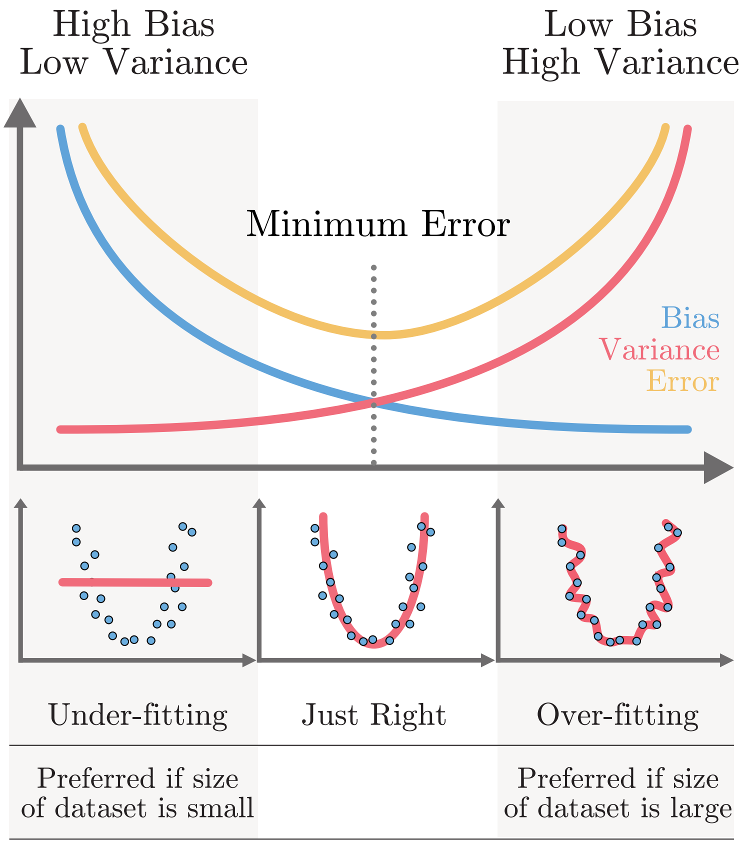
$$variance = \mathbb{E}\left[\left(f'(x) - \mathbb{E}[f'(x)]\right)^2\right]$$

- 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

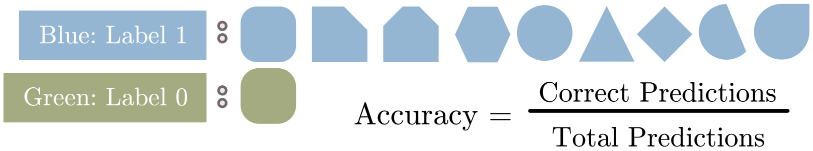




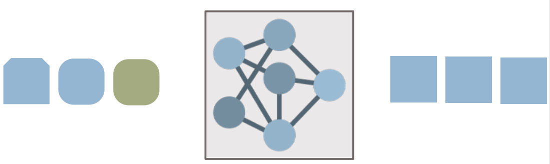
Bias variance Trade-off

- 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



$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$



Classifier that always **predicts label blue** yields prediction accuracy of 90%

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

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

Performance metrics associated with Class 1

		Actual Labels	
		1	0
Predicted Labels	1	True Positive	False Positive
	0	False Negative	True Negative

(Is your prediction correct?) (What did you predict)

True Negative

(Your prediction is correct) (You predicted 0)

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

$$\textbf{False +ve rate} = \frac{FP}{TN + FP}$$

$$\textbf{F1 score} = 2x\frac{(\text{Prec x Rec})}{(\text{Prec} + \text{Rec})}$$

$$\textbf{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

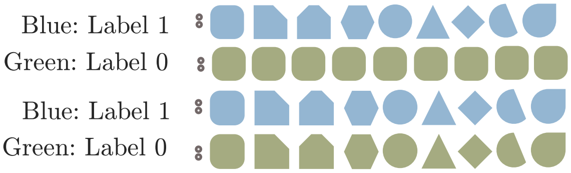
$$\textbf{Specificity} = \frac{TN}{TN + FP}$$

$$\textbf{Recall, Sensitivity} = \frac{TP}{TP + FN}$$

True +ve rate

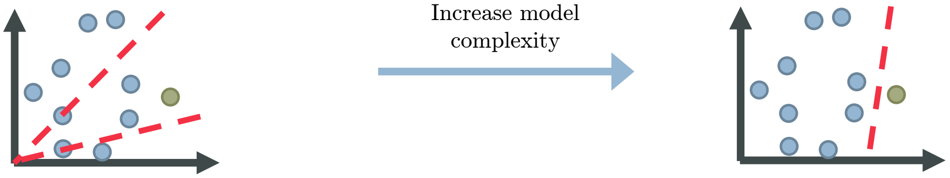
Possible solutions

1. **Data Replication:** Replicate the available data until the number of samples are comparable
2. **Synthetic Data:** Images: Rotate, dilate, crop, add noise to existing input images and create new data
3. **Modified Loss:** Modify the loss to reflect greater error when misclassifying smaller sample set



$$loss = a * \textbf{loss}_{\textbf{green}} + b * \textbf{loss}_{\textbf{blue}} \qquad a > b$$

4. **Change the algorithm:** Increase the model/algorithm complexity so that the two classes are perfectly separable (Con: Overfitting)



No straight line (y=ax) passing through origin can perfectly separate data. **Best solution:** line y=0, predict all labels blue

Straight line (y=ax+**b**) can perfectly separate data. Green class will no longer be predicted as blue

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

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

$$C[i,j] = cov(x_i, x_j)$$

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

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

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

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

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

Figure 1

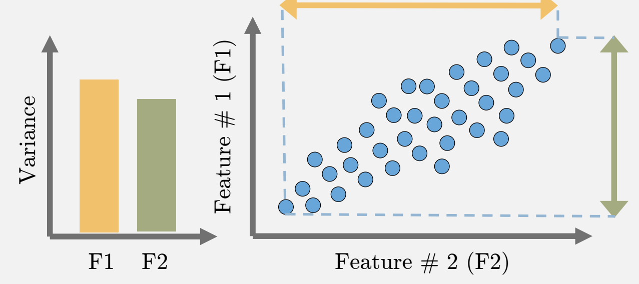


Figure 2

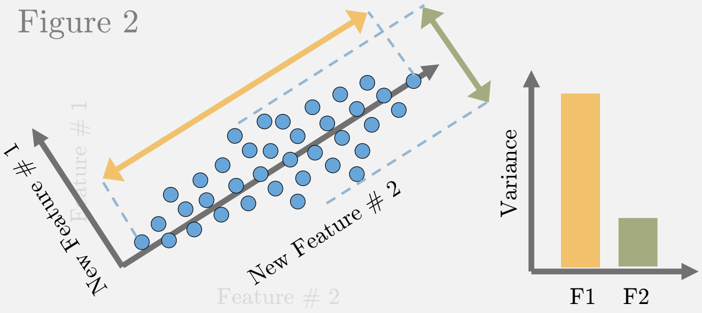


Figure 3

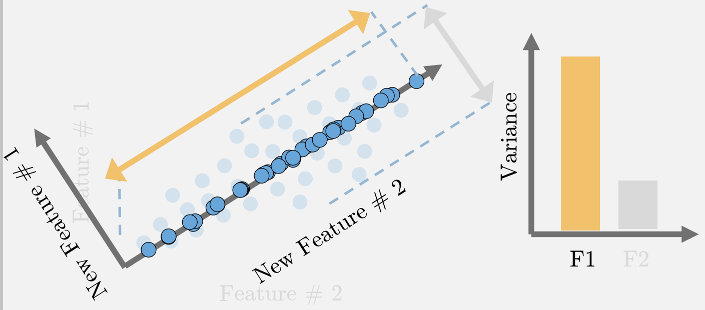


Figure 1: Datapoints with feature vectors as x and y-axis

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?

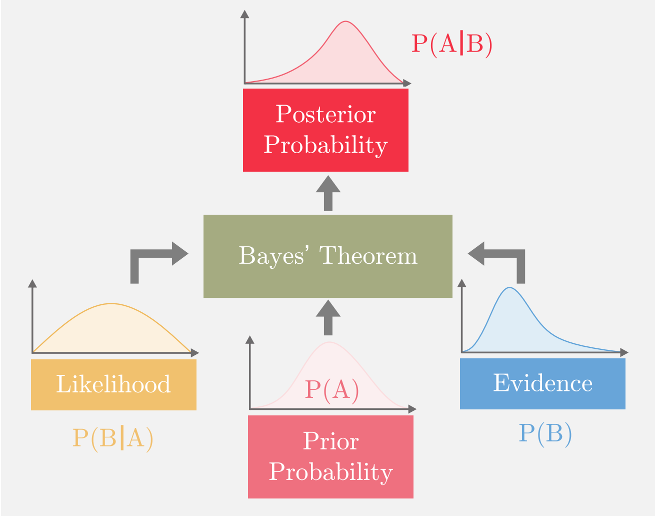
- Describes the probability of an event, based on prior knowledge of conditions that might be related to the event.

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

- 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)



P(A|B)

Posterior Probability

Bayes' Theorem

Likelihood

P(B|A)

P(A)

Prior Probability

Evidence

P(B)

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 Aposteriori Probability (MAP) Estimation

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

$$\hat{y}_{MAP} = argmax_y \;\; P(y)\prod_i P(x_i|y)$$

y that maximizes the product of prior and likelihood

Maximum Likelihood Estimation (MLE)

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

$$\hat{y}_{MLE} = argmax_y \prod_i P(x_i|y)$$

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, …), using Bayes' theorem

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

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

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

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

Cheat Sheet – Regression Analysis

What is Regression Analysis?

Fitting a function f(.) 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.

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

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

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. $n \rightarrow \infty, \sigma^2 \rightarrow 0$

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

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

$\mathcal{N}\left(\mu, \sigma^2\right) \rightarrow$ Gaussian with mean μ and variance σ^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.

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

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

5. LASSO Regression:

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

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

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

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

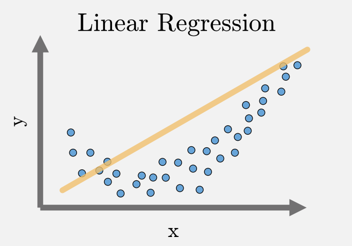
Can fit either a line, or polynomial with sigmoid activation minimizing the sum of mean-squared error for each datapoint. The labels y are binary class labels.

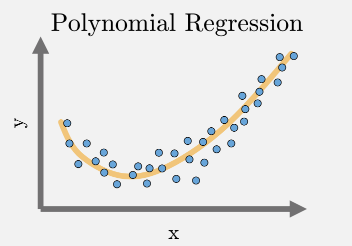
$$min_\beta \sum_i \|y_i - \sigma(f_\beta(x_i))\|^2$$

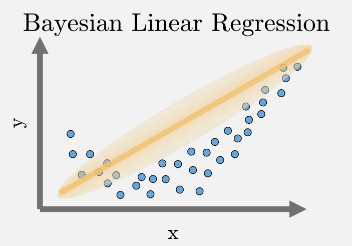
$$f_\beta(x_i) = f_\beta^{poly}(x_i) \ or \ f_\beta^{linear}(x_i)$$

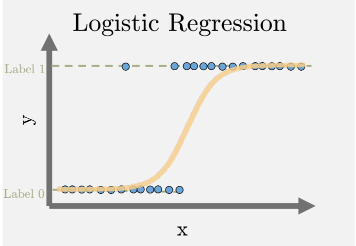
$$\sigma(t) = \frac{1}{1+e^{-t}}$$

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	$f_\beta^{poly}(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots$	$\sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2.$
Bayesian Linear	Gaussian distribution for each point	$\mathcal{N}\left(f_\beta(x_i), \sigma^2\right)$	$\sum_i \|y_i - \mathcal{N}\left(f_\beta(x_i), \sigma^2\right)\|^2$
Ridge	Linear/polynomial	$f_\beta^{poly}(x_i) \ or \ f_\beta^{linear}(x_i)$	$\sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2 + \sum_{j=0}^{n} \beta_j^2$
LASSO	Linear/polynomial	$f_\beta^{poly}(x_i) \ or \ f_\beta^{linear}(x_i)$	$\sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2 + \sum_{j=0}^{n} \lvert\beta_j\rvert$
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

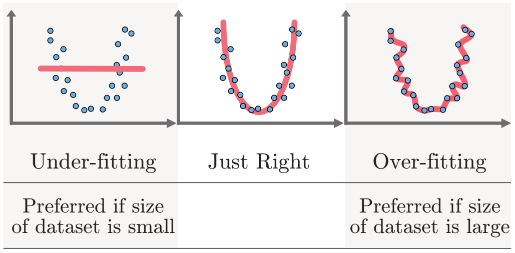


Figure 1. Overfitting

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.

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

- **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

$$loss = error(y, \hat{y})$$ [redacted] $$\lambda \geq 0,\ \lambda \propto model\ bias,\quad \lambda \propto \frac{1}{model\ variance}$$

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

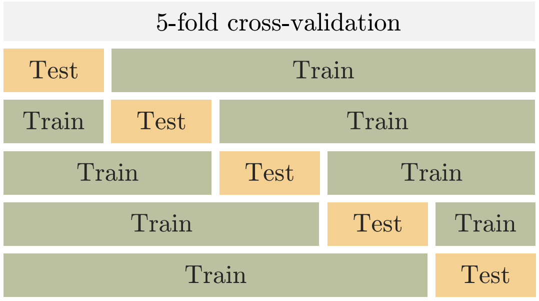
$$loss = error(p, \hat{p})$$ [redacted] $$\lambda \geq 0,\ \lambda \propto model\ bias,\quad \lambda \propto \frac{1}{model\ variance}$$

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.



5-fold cross-validation

Test	Train			
Train	Test	Train		
Train		Test	Train	
Train			Test	Train
Train				Test

Figure 2. K-fold CV

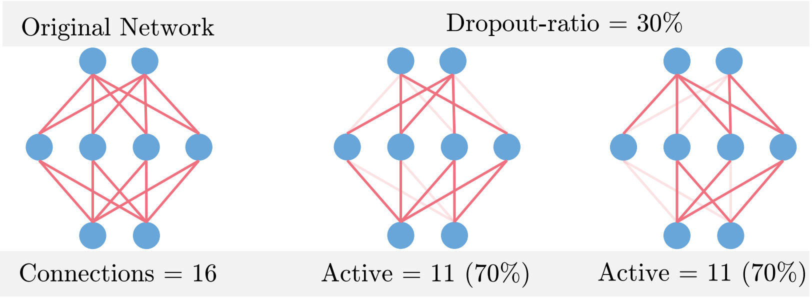


Figure 3. Drop-out

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 size		in	out	# of Param
227	227	3	55	55	96	conv1	4	0	11	11	3	96	34944
55	55	96	27	27	96	maxpool1	2	0	3	3	96	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	256	256	0
13	13	256	13	13	384	conv3	1	1	3	3	256	384	885120
13	13	384	13	13	384	conv4	1	1	3	3	384	384	1327488
13	13	384	13	13	256	conv5	1	1	3	3	384	256	884992
13	13	256	6	6	256	maxpool5	2	0	3	3	256	256	0
						fc6			1	1	9216	4096	37752832
						fc7			1	1	4096	4096	16781312
						fc8			1	1	4096	1000	4097000
						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	3	224	224	64	conv3-64	1	3	3	3	64	1792
2	224	224	64	224	224	64	conv3064	1	3	3	64	64	36928
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664
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
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0
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	28	28	512	conv3-512	1	3	3	512	512	2359808
	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	14	14	512	conv3-512	1	3	3	512	512	2359808
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0
14	1	1	25088	1	1	4096	fc		1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc		1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc		1	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 do solve the vanishing gradient problem. The basic building block of ResNet is a Residual block that is repeated throughout the network.

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
	112	112	64	56	56	64	maxpool	2	0.5	3	3	64	64	0
2	56	56	64	56	56	64	conv2-1	1	1	3	3	64	64	36928
3	56	56	64	56	56	64	conv2-2	1	1	3	3	64	64	36928
4	56	56	64	56	56	64	conv2-3	1	1	3	3	64	64	36928
5	56	56	64	56	56	64	conv2-4	1	1	3	3	64	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
	7	7	512	1	1	512	avg pool	7	0	7	7	512	512	0
18	1	1	512	1	1	1000	fc					512	1000	513000
							Total							**11,511,784**

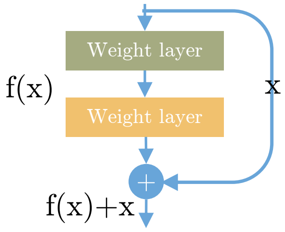


Figure 1 ResNet Block

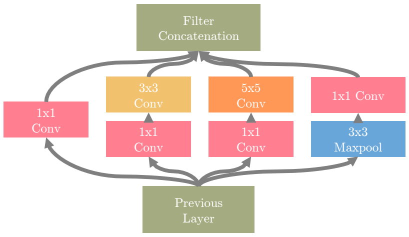


Figure 2 Inception Block

Inception – 2014

Why: Lager 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.

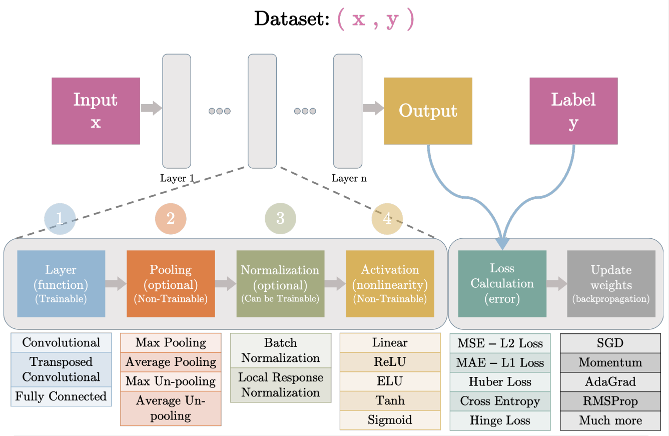
	GoogLeNet - Structural Details														
	Input Image			output			Layer	Input Layer	Stride	Pad	Kernel		in	out	Param
	227	227	3	112	112	64	conv1	input	2	1	7	7	3	64	9472
	112	112	64	56	56	64	maxpool1	conv1	2	0.5	3	3	64	64	0
	56	56	64	56	56	64	conv1x1	maxpool1	1	0	1	1	64	64	4160
	56	56	64	56	56	192	conv2-1		1	1	3	3	64	192	110784
	56	56	192	28	28	192	maxpool2		2	0.5	3	3	192	192	0
inception (3a)	28	28	192	28	28	96	conv1x1a	maxpool2	1	0	1	1	192	96	18528
	28	28	96	28	28	16	conv1x1b	maxpool2	1	0	1	1	192	16	3088
	28	28	192	28	28	192	maxpool-a	maxpool2	1	1	3	3	192	192	0
	28	28	192	28	28	64	conv1x1c	maxpool2	1	0	1	1	192	64	12352
	28	28	96	28	28	128	conv3-3	conv1x1a	1	1	3	3	96	128	110720
	28	28	16	28	28	32	conv5x5	conv1x1b	1	2	5	5	16	32	12832
	28	28	192	28	28	32	conv1x1d	maxpool-a	1	0	1	1	192	32	6176
				28	28	256	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
inception (3b)	28	28	256	28	28	128	conv1x1a	depth-concat	1	0	1	1	256	128	32896
	28	28	128	28	28	32	conv1x1b	depth-concat	1	0	1	1	256	32	8224
	28	28	192	28	28	256	maxpool-a	depth-concat	1	1	3	3	256	256	0
	28	28	192	28	28	128	conv1x1c	depth-concat	1	0	1	1	256	128	32896
	28	28	96	28	28	192	conv3-3	conv1x1a	1	1	3	3	128	192	221376
	28	28	16	28	28	96	conv5x5	conv1x1b	1	2	5	5	32	96	76896
	28	28	192	28	28	64	conv1x1d	maxpool-a	1	0	1	1	256	64	16448
				28	28	480	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
	28	28	480	14	14	480	maxpool3	depth-concat	2	0.5	3	3	480	480	0
inception (4a)	14	14	480	14	14	96	conv1x1a	maxpool3	1	0	1	1	480	96	46176
	14	14	480	14	14	16	conv1x1b	maxpool3	1	0	1	1	480	16	7696
	14	14	480	14	14	480	maxpool-a	maxpool3	1	1	3	3	480	480	0
	14	14	480	14	14	192	conv1x1c	maxpool3	1	0	1	1	480	192	92352
	14	14	96	14	14	208	conv3-3	conv1x1a	1	1	3	3	96	208	179920
	14	14	16	14	14	48	conv5x5	conv1x1b	1	2	5	5	16	48	19248
	14	14	192	14	14	64	conv1x1d	maxpool-a	1	0	1	1	480	64	30784
				14	14	512	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
inception (4b)	14	14	512	14	14	112	conv1x1a	depth-concat	1	0	1	1	512	112	57456
	14	14	512	14	14	24	conv1x1b	depth-concat	1	0	1	1	64	24	1560
	14	14	512	14	14	64	maxpool-a	depth-concat	1	1	3	3	64	64	0
	14	14	512	14	14	160	conv1x1c	depth-concat	1	0	1	1	64	160	10400
	14	14	96	14	14	224	conv3-3	conv1x1a	1	1	3	3	112	224	226016
	14	14	16	14	14	64	conv5x5	conv1x1b	1	2	5	5	24	64	38464
	14	14	160	14	14	64	conv1x1d	maxpool-a	1	0	1	1	64	64	4160
				14	14	512	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
inception (4c)	14	14	512	14	14	128	conv1x1a	depth-concat	1	0	1	1	512	128	65664
	14	14	512	14	14	24	conv1x1b	depth-concat	1	0	1	1	64	24	1560
	14	14	512	14	14	64	maxpool-a	depth-concat	1	1	3	3	64	64	0
	14	14	512	14	14	128	conv1x1c	depth-concat	1	0	1	1	64	128	8320
	14	14	96	14	14	256	conv3-3	conv1x1a	1	1	3	3	128	256	295168
	14	14	16	14	14	64	conv5x5	conv1x1b	1	2	5	5	24	64	38464
	14	14	128	14	14	64	conv1x1d	maxpool-a	1	0	1	1	64	64	4160
				14	14	512	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
inception (4d)	14	14	512	14	14	144	conv1x1a	depth-concat	1	0	1	1	512	144	73872
	14	14	512	14	14	32	conv1x1b	depth-concat	1	0	1	1	64	32	2080
	14	14	512	14	14	64	maxpool-a	depth-concat	1	1	3	3	64	64	0
	14	14	512	14	14	112	conv1x1c	depth-concat	1	0	1	1	64	112	7280
	14	14	96	14	14	288	conv3-3	conv1x1a	1	1	3	3	144	288	373536
	14	14	16	14	14	64	conv5x5	conv1x1b	1	2	5	5	32	64	51264
	14	14	112	14	14	64	conv1x1d	maxpool-a	1	0	1	1	64	64	4160
				14	14	528	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
inception (4e)	14	14	528	14	14	160	conv1x1a	depth-concat	1	0	1	1	528	160	84640
	14	14	528	14	14	32	conv1x1b	depth-concat	1	0	1	1	64	32	2080
	14	14	528	14	14	64	maxpool-a	depth-concat	1	1	3	3	64	64	0
	14	14	528	14	14	256	conv1x1c	depth-concat	1	0	1	1	64	256	16640
	14	14	96	14	14	320	conv3-3	conv1x1a	1	1	3	3	160	320	461120
	14	14	16	14	14	128	conv5x5	conv1x1b	1	2	5	5	32	128	102528
	14	14	256	14	14	128	conv1x1d	maxpool-a	1	0	1	1	64	128	8320
				14	14	832	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
	14	14	832	7	7	832	maxpool4	depth-concat	2	0.5	3	3	832	832	0
inception (5a)	7	7	832	7	7	160	conv1x1a	maxpool4	1	0	1	1	832	160	133280
	7	7	832	7	7	32	conv1x1b	maxpool4	1	0	1	1	832	32	26656
	7	7	832	7	7	832	maxpool-a	maxpool4	1	1	3	3	832	832	0
	7	7	832	7	7	256	conv1x1c	maxpool4	1	0	1	1	832	256	213248
	7	7	96	7	7	320	conv3-3	conv1x1a	1	1	3	3	160	320	461120
	7	7	16	7	7	128	conv5x5	conv1x1b	1	2	5	5	32	128	102528
	7	7	256	7	7	128	conv1x1d	maxpool-a	1	0	1	1	832	128	106624
				7	7	832	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
inception (5b)	7	7	832	7	7	192	conv1x1a	depth-concat	1	0	1	1	832	192	159936
	7	7	832	7	7	48	conv1x1b	depth-concat	1	0	1	1	832	48	39984
	7	7	832	7	7	832	maxpool-a	depth-concat	1	1	3	3	832	832	0
	7	7	832	7	7	384	conv1x1c	depth-concat	1	0	1	1	832	384	319872
	7	7	96	7	7	384	conv3-3	conv1x1a	1	1	3	3	192	384	663936
	7	7	16	7	7	128	conv5x5	conv1x1b	1	2	5	5	48	128	153728
	7	7	384	7	7	128	conv1x1d	maxpool-a	1	0	1	1	128	128	16512
				7	7	1024	depth-concat	conv1x1c, conv3x3, conv5x5, conv1x1d							
	7	7	1024	1	1	1024	avgpool	depth-concat	1	0	7	7	1024	1024	0
	1	1	1024	1	1	1000	fc	depth-concat	1	0	1	1	1024	1000	1025000
									Total						**6,414,360**

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

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.



Layer (function)	Pooling (optional)	Normalization (optional)	Activation (nonlinearity)	Loss Calculation (error)	Update weights
Convolutional	Max Pooling	Batch Normalization	Linear	MSE – L2 Loss	SGD
Transposed Convolutional	Average Pooling	Local Response Normalization	ReLU	MAE – L1 Loss	Momentum
Fully Connected	Max Un-pooling		ELU	Huber Loss	AdaGrad
	Average Un-pooling		Tanh	Cross Entropy	RMSProp
			Sigmoid	Hinge Loss	Much more

CNN Template:

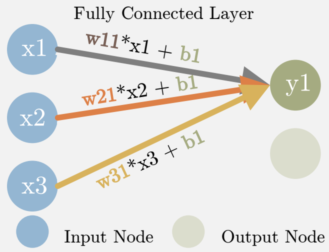
Most of the commonly used hidden layers (not all) follow a pattern

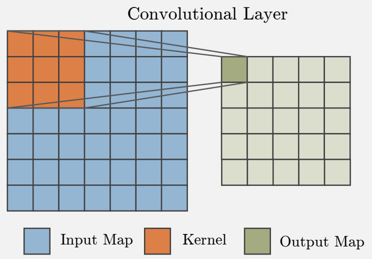
1. **Layer function:** Basic transforming function such as convolutional or fully connected layer.

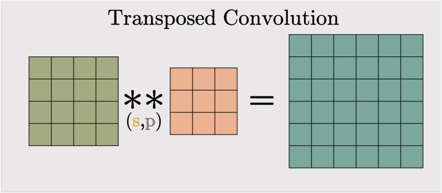
a. **Fully Connected:** Linear functions between the input and the output

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



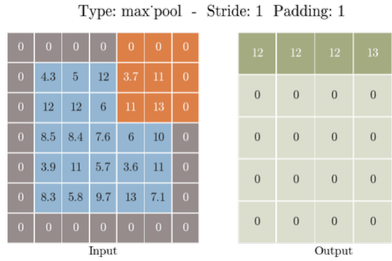




2. **Pooling:** Non-trainable layer to change the size of the feature map

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.



Type: max˙pool - Stride: 1 Padding: 1

0	0	0	0	0	0	0
0	4.3	5	12	3.7	11	0
0	12	12	6	11	13	0
0	8.5	8.4	7.6	6	10	0
0	3.9	11	5.7	3.6	11	0
0	8.3	5.8	9.7	13	7.1	0
0	0	0	0	0	0	0

Input

12	12	12	13	13
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Output

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.

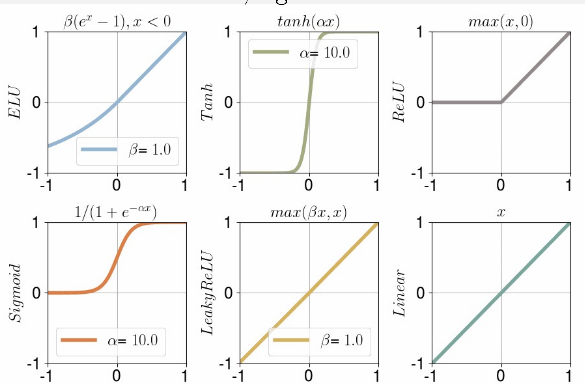
b. **Batch Normalization:** A trainable approach to normalizing the data by learning scale and shift variable during training.

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

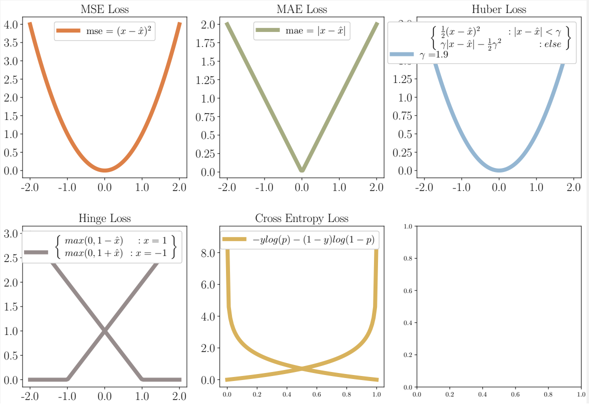
c. **Bounded functions:** tanh, sigmoid



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

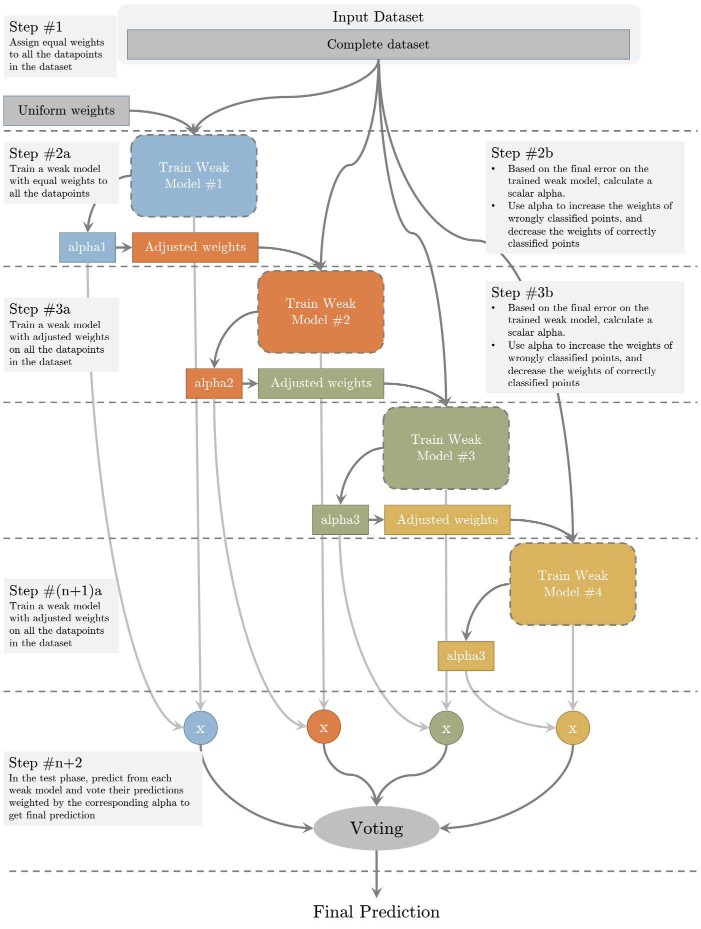
1. **Bagging:** Trains N different weak models (usually of same types – homogenous) with N non-overlapping subset of the input dataset in parallel. In the test phase, each model is evaluated. The label with the greatest number of predictions is selected as the prediction. Bagging methods reduces variance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

Ensemble Method – Boosting



Step #1
Assign equal weights to all the datapoints in the dataset

Step #2a
Train a weak model with equal weights to all the datapoints

Step #2b
- Based on the final error on the trained weak model, calculate a scalar alpha.
- Use alpha to increase the weights of wrongly classified points, and decrease the weights of correctly classified points

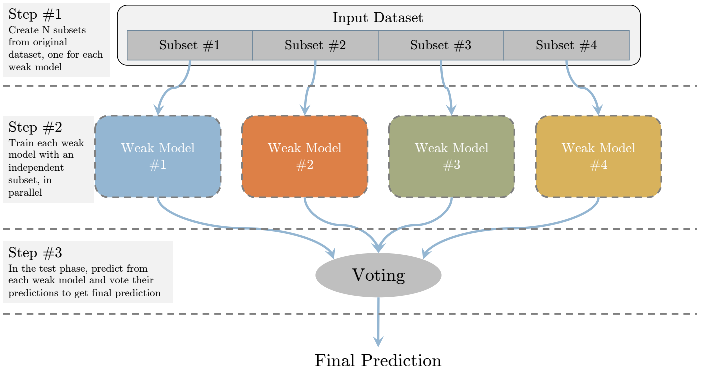
Step #3a
Train a weak model with adjusted weights on all the datapoints in the dataset

Step #3b
- Based on the final error on the trained weak model, calculate a scalar alpha.
- Use alpha to increase the weights of wrongly classified points, and decrease the weights of correctly classified points

Step #(n+1)a
Train a weak model with adjusted weights on all the datapoints in the dataset

Step #n+2
In the test phase, predict from each weak model and vote their predictions weighted by the corresponding alpha to get final prediction

Ensemble Method – Bagging

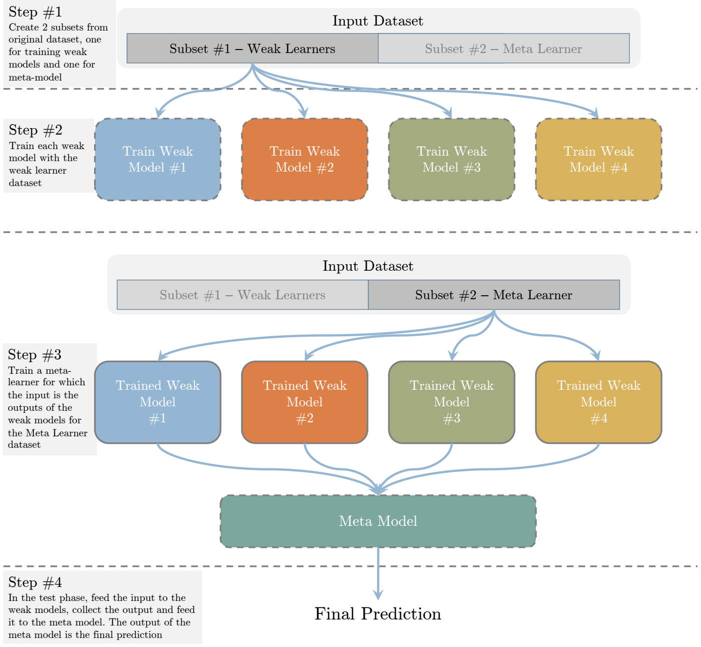


Step #1
Create N subsets from original dataset, one for each weak model

Step #2
Train each weak model with an independent subset, in parallel

Step #3
In the test phase, predict from each weak model and vote their predictions to get final prediction

Ensemble Method – Stacking



Step #1
Create 2 subsets from original dataset, one for training weak models and one for meta-model

Step #2
Train each weak model with the weak learner dataset

Step #3
Train a meta-learner for which the input is the outputs of the weak models for the Meta Learner dataset

Step #4
In the test phase, feed the input to the weak models, collect the output and feed it to the meta model. The output of the meta model is the final prediction

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)

1/4

How to prepare for behavioral interview?

Collect stories, assign keywords, practice the STAR format

Keywords

List important keywords that will be populated with your personal stories. Most common keywords are given in the table below

Conflict Resolution	Negotiation	Compromise to achieve goal	Creativity	Flexibility	Convincing
Handling Crisis	Challenging Situation	Working with difficult people	Another team priorities not aligned	Adjust to a colleague style	Take Stand
Handling –ve feedback	Coworker view of you	Working with a deadline	Your strength	Your weakness	Influence Others
Handling failure	Handling unexpected situation	Converting challenge to opportunity	Decision without enough data	Conflict Resolution	Mentorship/ Leadership

Stories

1. List all the organizations you have been a part of. For example
   1. Academia: BSc, MSc, PhD
   2. Industry: Jobs, Internship
   3. Societies: Cultural, Technical, Sports
2. Think of stories from step 1 that can fall into one of the keywords categories. The more stories the better. You should have at least 10-15 stories.
3. Create a summary table by assigning multiple keywords to each stories. This will help you filter out the stories when the question asked in the interview. An example can be seen below

Story 1:	[Convincing] [Take Stand] [influence other]
Story 2:	[Mentorship] [Leadership]
Story 3:	[Conflict resolution] [Negotiation]
Story 4:	[decision-without-enough-data]

STAR Format

Write down the stories in the STAR format as explained in the 2/4 part of this cheat sheet. This will help you practice the organization of story in a meaningful way.

2/4

How to prepare for behavioral interview?

Direct*, meaningful*, personalized*, logical*

*(Respective colors are used to identify these characteristics in the example)

Example: “Tell us about a time when you had to convince senior executives”

Situation

Explain the situation and provide necessary context for your story.

S

“I worked as an intern in XYZ company in the summer of 2019. The project details provided to me was elaborative. After some initial brainstorming, and research I realized that the project approach can be modified to make it more efficient in terms of the underlying KPIs. I decided to talk to my manager about it.”

Task

Explain the task and your responsibility in the situation

T

“I had an hour-long call with my manager and explained him in detail the proposed approach and how it could improve the KPIs. I was able to convince him. He asked me if I will be able to present my proposed approach for approval in front of the higher executives. I agreed to it. I was working out of the ABC(city) office and the executives need to fly in from XYZ(city) office.”

Action

Walk through the steps and actions you took to address the issue

A

“I did a quick background check on the executives to know better about their area of expertise so that I can convince them accordingly. I prepared an elaborative 15 slide presentation starting with explaining their approach, moving onto my proposed approach and finally comparing them on preliminary results.

Result

State the outcome of the result of your actions

R

“After some active discussion we were able to establish that the proposed approach was better than the initial one. The executives proposed a few small changes to my approach and really appreciated my stand. At the end of my internship, I was selected among the 3 out of 68 interns who got to meet the senior vice president of the company over lunch.”

3/4

How to answer a behavioral question?

Understand, Extract, Map, Select and Apply

Example: “Tell us about a time when you had to convince senior executives”

Understand

Understand the question

Example: A story where I was able to convince my seniors. Maybe they had something in mind, and I had a better approach and tried to convince them

Extract

Extract keywords and tags

Extract useful keywords that encapsulates the gist of the question

Example:

[Convincing], [Creative], [Leadership]

Map

Map the keyword to your stories

Shortlist all the stories that fall under the keywords extracted from previous step

Example:

Story1, Story2, Story3, Story4, ... , Story N

Select

Select the best story

From the shortlisted stories, pick the one that best describes the question and has not been used so far in the interview

Example: Story3

Apply

Apply the STAR method

Apply the STAR method on the selected story to answer the question

Example: See Cheat Sheet 2/3 for details

4/4

Behavioral Interview Cheat Sheet

Summarizing the behavioral interview

How to prepare for the interview

1 **Gather important topics as keywords**

Understand and collect all the important topics commonly asked in the interview

2 **Collect your stories**

Based on all the organizations you have been a part of, think of all the stories that fall under the keywords above

3 **Practice stories in STAR format**

Practice each story using the STAR format. You will have to answer the question following this format.

4 **Assign keywords to stories**

Assign each of your story one or more keywords. This will help you recall them quickly

5 **Create a summary table**

Create a summary table mapping stories to their associated keywords. This will be used during the behavioral question

How to answer a question during interview

U **Understand the question**

Understand the question and clarify any confusions that you have

E **Extract the keywords**

Try to extract one or more of the keywords from the question

M **Map the keywords to stories**

Based on the keywords extracted, find the stories using the summary table created during preparation (Step 4)

S **Select a story**

Since each keyword maybe assigned to multiple stories, select the one that is most relevant and has not been used.

A **Apply the START format**

Once the story has been shortlisted, apply STAR format on the story to answer the question.