

[medium.com](https://medium.com)

# Cheat Sheets for Machine Learning Interview Topics - Aqeel Anwar - Medium

*Aqeel Anwar*

6–7 minutes



Illustration credit [rawpixels.com](https://rawpixels.com)



## Updates:

Dec 25, 2021: Added Auto Encoder and variational Encoder

Dec 25, 2020: Added Ensemble Methods

Download the updated version of the cheat sheets from

<http://cheatsheets.aqeel-anwar.com/>

A couple of years ago I started applying for internships in the area of Machine Learning and ML system design. I had been studying and actively researching in the area of ML for a few

years then. I was familiar with most of the basic topics. But when I started interviewing, I realized that though I had a general understanding of the topics, I required a quick go-through before I can answer it perfectly.

So I decided to refresh my concepts. I realized that before every interview, I was required to go through the topics again. So, I created my handwritten notes. Skimming through them was much easier than going through slides and book chapters. It provided me with a quick boost to my understanding in a short amount of time. I decided to convert my hand-written notes into compact cheat sheets that might come in handy for ML interviews and daily data-scientist life in general.

The rest of the article is based on those cheat sheets. For each topic, I provide

- An overview in form of a cheat sheet
- Example interview questions
- Suggested articles for a detailed understanding of the topic.

**Note 1:** These cheat sheets are aimed at refreshing the concepts and are not meant to provide in-depth understandings of the topics for beginners.

**Note 2:** The article is constantly updated for more cheat sheets.

**Source:** All of these cheat sheets (and more) can be downloaded in pdf format from [www.cheatsheets.aqeel-anwar.com](http://www.cheatsheets.aqeel-anwar.com).

## Bias and Variance in Machine Learning Models

### a) Overview:

### 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?

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

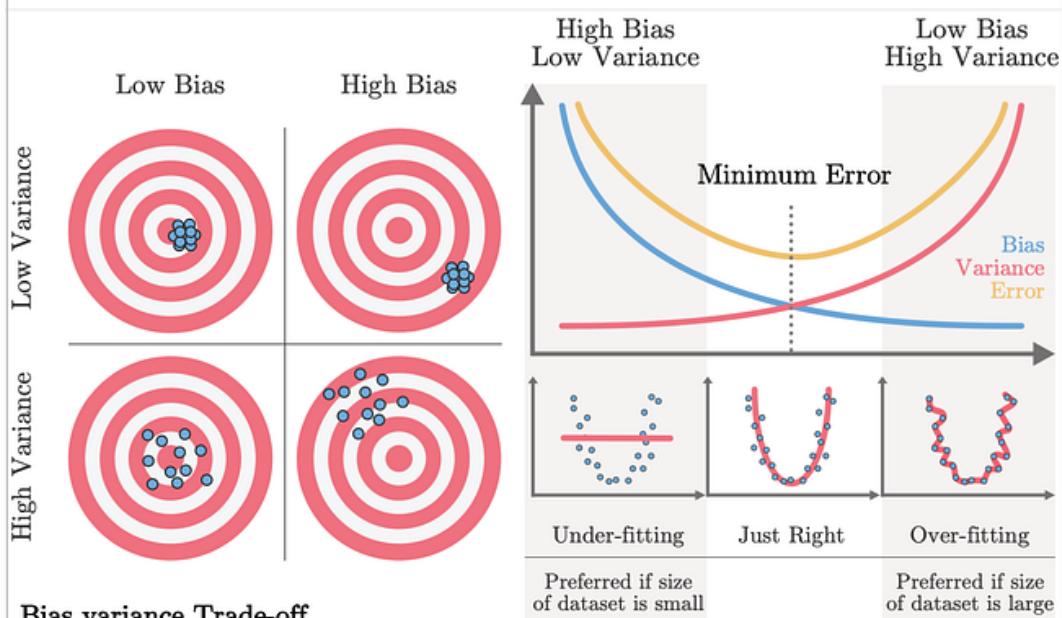
- Average variability in the model prediction for the given dataset
- The variance of the estimated function tells you how much the function can adjust to the change in the dataset

### 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



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



Bias-variance tradeoff cheat sheet — Image by author

## b) Example Questions:

1. What is Bias in ML models?
2. What is Variance in ML models?
3. What is the trade-off between bias and variance?
4. What are the demerits of a high bias / high variance ML model?
5. How do you select the model (high bias or high variance) based

on the training data size?

### c) Detailed Article:

- [Understanding the Bias-Variance Tradeoff and visualizing it with example and python code](#)

## Imbalanced data in Machine Learning

### a) Overview:

**Cheat Sheet – Imbalanced Data in Classification**



Blue: Label 1  
Green: Label 0

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  
(You predicted 0)

True Positive  
(Your prediction is correct)

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

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

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

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

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

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

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

Blue: Label 1  
Green: Label 0

Blue: Label 1  
Green: Label 0

$loss = a * loss_{green} + b * loss_{blue}$      $a > b$

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

Increase model complexity

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

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

## Imbalanced data in classification — Image by Author

### b) Example Questions:

1. What is imbalanced data in classification?
2. Is accuracy a good performance metric? When does it fail to capture the performance of an ML system?
3. What are Precision and Recall? Give an example
4. How to address the issue of imbalanced data?

### c) Detailed Articles:

- [A walk through imbalanced classes in machine learning through a visual cheat sheet](#)

## Bayes' Theorem

### a) Overview:

### 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 Apriori 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.

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

$y$  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  $(x_1, x_2, x_3, \dots)$ , is given by. We assume we don't have any prior knowledge of the quantity being estimated.

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

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

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

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

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



Bayes' Theorem and Classifier — Image by Author

## b) Example Questions:

1. What is Bayes' theorem?
2. Toy example to implement Bayes' theorem
3. What is the difference between MLE and MAP?
4. When are MAP and MLE equal?

## c) Detailed Articles:

- [A Gentle Introduction to Bayes Theorem for Machine Learning](#)

## Principal Component Analysis and Dimensionality Reduction

### a) Overview:

#### 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

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

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

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

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

**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  $V_{reduced} = V[:, 0 : m]$
3. Transform the data for the new basis (feature vectors)  $X_{reduced} = X_{new} \times V_{reduced}$
4. The importance of the feature vector is proportional to the magnitude of the eigen value

Figure 1

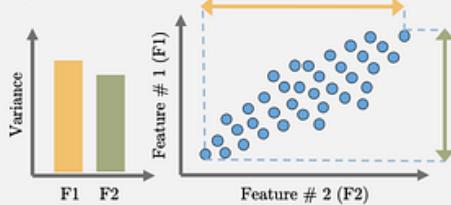


Figure 2

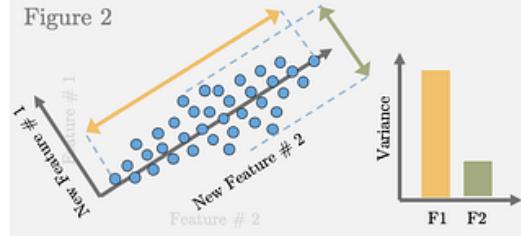
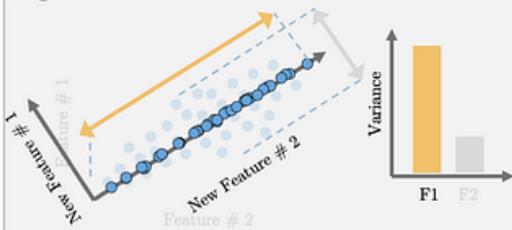


Figure 3



**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

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



PCA and Dimensionality Reduction — Image by Author

## b) Example Questions:

1. What is Principal Component Analysis?
2. How can we use PCA to reduce dimensions?
3. What do the eigenvalues signify in the context of PCA? (*Greater the magnitude of eigenvalue, the more information is preserved if we keep that corresponding eigenvector as a feature vector for our data*)

## c) Detailed Articles:

- [Understanding Principle Component Analysis\(PCA\) step by step](#)

step.

# Regression in Machine Learning

## a) Overview:

Cheat Sheet – Regression Analysis			
<b>What is Regression Analysis?</b>			
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			
<b>1. Linear Regression:</b> Fits a <b>line</b> minimizing the <b>sum of mean-squared error</b> for each datapoint.			
$\min_{\beta} \sum_i \ y_i - f_{\beta}^{\text{linear}}(x_i)\ ^2$ $f_{\beta}^{\text{linear}}(x_i) = \beta_0 + \beta_1 x_i$			
<b>2. Polynomial Regression:</b> Fits a <b>polynomial</b> of order $k$ ( $k+1$ unknowns) minimizing the <b>sum of mean-squared error</b> for each datapoint.			
$\min_{\beta} \sum_{i=0}^m \ y_i - f_{\beta}^{\text{poly}}(x_i)\ ^2$ $f_{\beta}^{\text{poly}}(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \beta_k x_i^k$			
<b>3. Bayesian Regression:</b> For each datapoint, fits a <b>gaussian distribution</b> by minimizing the <b>mean-squared error</b> . As the number of data points $x_i$ increases, it converges to point $\mathcal{N}(\mu, \sigma^2) \rightarrow$ Gaussian with mean $\mu$ and variance $\sigma^2$ estimates i.e. $n \rightarrow \infty, \sigma^2 \rightarrow 0$			
<b>4. Ridge Regression:</b> Can fit either a <b>line</b> , <b>or polynomial</b> minimizing the <b>sum of mean-squared error</b> for each datapoint and the <b>weighted L2 norm</b> of the function parameters beta.			
$\min_{\beta} \sum_{i=0}^m \ y_i - f_{\beta}(x_i)\ ^2 + \sum_{j=0}^k \beta_j^2$ $f_{\beta}(x_i) = f_{\beta}^{\text{poly}}(x_i) \text{ or } f_{\beta}^{\text{linear}}(x_i)$			
<b>5. LASSO Regression:</b> Can fit either a <b>line</b> , <b>or polynomial</b> minimizing the <b>sum of mean-squared error</b> for each datapoint and the <b>weighted L1 norm</b> of the function parameters beta.			
$\min_{\beta} \sum_{i=0}^m \ y_i - f_{\beta}(x_i)\ ^2 + \sum_{j=0}^k  \beta_j $ $f_{\beta}(x_i) = f_{\beta}^{\text{poly}}(x_i) \text{ or } f_{\beta}^{\text{linear}}(x_i)$			
<b>6. Logistic Regression:</b> Can fit either a <b>line</b> , <b>or polynomial with sigmoid activation</b> minimizing the <b>binary cross-entropy loss</b> for each datapoint. The labels $y$ are binary class labels.			
$\min_{\beta} \sum_i -y_i \log(\sigma(f_{\beta}(x_i))) - (1 - y_i) \log(1 - \sigma(f_{\beta}(x_i)))$ $f_{\beta}(x_i) = f_{\beta}^{\text{poly}}(x_i) \text{ or } f_{\beta}^{\text{linear}}(x_i)$ $\sigma(t) = \frac{1}{1 + e^{-t}}$			
<b>Visual Representation:</b>			
<b>Summary:</b>			
What does it fit?	Estimated function	Error Function	
Linear	$A \text{ line in } n \text{ dimensions}$	$f_{\beta}^{\text{linear}}(x_i) = \beta_0 + \beta_1 x_i$	$\sum_{i=0}^m \ y_i - f_{\beta}(x_i)\ ^2$
Polynomial	$A \text{ polynomial of order } k$	$f_{\beta}^{\text{poly}}(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots$	$\sum_{i=0}^m \ y_i - f_{\beta}(x_i)\ ^2$
Bayesian Linear	$\text{Gaussian distribution for each point}$	$\mathcal{N}(f_{\beta}(x_i), \sigma^2)$	$\sum_i \ y_i - \mathcal{N}(f_{\beta}(x_i), \sigma^2)\ ^2$
Ridge	$\text{Linear/polynomial}$	$f_{\beta}^{\text{poly}}(x_i) \text{ or } f_{\beta}^{\text{linear}}(x_i)$	$\sum_{i=0}^m \ y_i - f_{\beta}(x_i)\ ^2 + \sum_{j=0}^k \beta_j^2$
LASSO	$\text{Linear/polynomial}$	$f_{\beta}^{\text{poly}}(x_i) \text{ or } f_{\beta}^{\text{linear}}(x_i)$	$\sum_{i=0}^m \ y_i - f_{\beta}(x_i)\ ^2 + \sum_{j=0}^k  \beta_j $
Logistic	$\text{Linear/polynomial with sigmoid}$	$\sigma(f_{\beta}(x_i))$	$\min_{\beta} \sum_i -y_i \log(\sigma(f_{\beta}(x_i))) - (1 - y_i) \log(1 - \sigma(f_{\beta}(x_i)))$

Regression Analysis — Image by Author

## b) Example Questions:

1. What is Regression in ML?
2. How can we introduce regularization in regression? (*LASSO and Ridge*)
3. What impact does LASSO and Ridge regression has on the weights of the model? (*Ridge tries to reduce the size of the weights learned, whereas LASSO tries to force them to zero creating a more sparse set of weights*)
4. When does the prediction by Bayesian linear regression approach the prediction of linear regression? (*When the number of data points is large enough*)
5. Is logistic regression a misnomer? (*Yes, because it is not regression, but classification based on regression*)

### c) Detailed Articles:

- [5 Types of Regression and their properties](#)
- [Introduction to Bayesian Linear Regression](#)

## Regularization in Machine Learning

### a) Overview:

**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

**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.

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

- **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

Under-fitting      Just Right      Over-fitting

Preferred if size of dataset is small      Preferred if size of dataset is large

Figure 1. Overfitting

average magnitude of all weights

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

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

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

### 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.

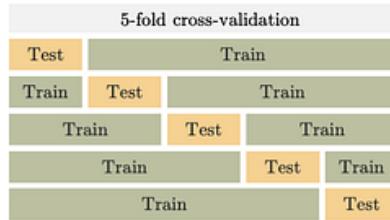


Figure 2. K-fold CV

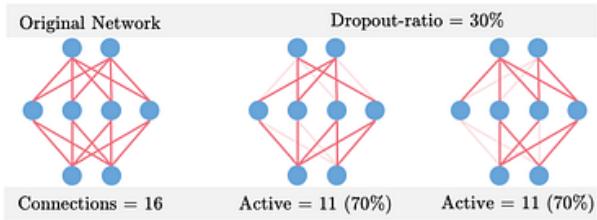


Figure 3. Drop-out

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



Regularization in ML — Image by Author

## b) Example Questions:

1. What is regularization in ML?
2. How can we address over-fitting?
3. What is K-fold cross-validation?
4. What is the difference between L1 and L2 regularization?
5. Why do we use dropout?

## c) Detailed Articles:

- [Regularization](#)
- [Regularization in Machine Learning](#)

# Basics of Convolutional Neural Network

## a) Overview:

# Cheat Sheet – Convolutional Neural Network

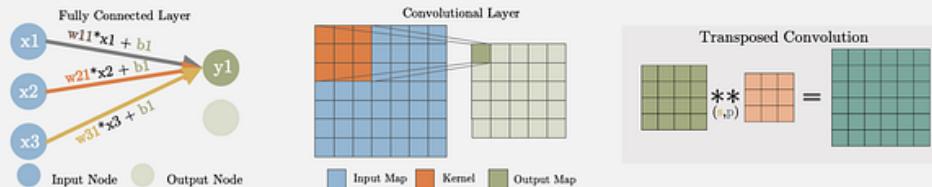
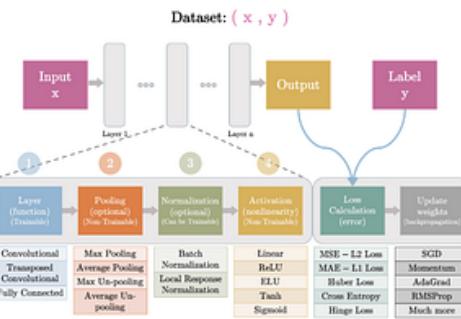
## Convolutional Neural Network:

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. | <table border="1"> <tr><td>Convolutional</td><td>Max Pooling</td><td>Linear</td><td>SCE</td></tr> <tr><td>Transposed</td><td>Average Pooling</td><td>ReLU</td><td>Momentum</td></tr> <tr><td>Convolutional</td><td>Max U-pooling</td><td>ELU</td><td>AdaGrad</td></tr> <tr><td>Fully Connected</td><td>Average U-pooling</td><td>Tanh</td><td>RMSProp</td></tr> <tr><td></td><td></td><td>Sigmoid</td><td>Adam</td></tr> </table>                         | Convolutional | Max Pooling   | Linear        | SCE | Transposed      | Average Pooling | ReLU          | Momentum | Convolutional | Max U-pooling | ELU        | AdaGrad | Fully Connected   | Average U-pooling | Tanh          | RMSProp |  |  | Sigmoid    | Adam |
| Convolutional                                                                                  | Max Pooling                                                                                                                                                                                                                                                                                                                                                                                                                                               | Linear        | SCE           |               |     |                 |                 |               |          |               |               |            |         |                   |                   |               |         |  |  |            |      |
| Transposed                                                                                     | Average Pooling                                                                                                                                                                                                                                                                                                                                                                                                                                           | ReLU          | Momentum      |               |     |                 |                 |               |          |               |               |            |         |                   |                   |               |         |  |  |            |      |
| Convolutional                                                                                  | Max U-pooling                                                                                                                                                                                                                                                                                                                                                                                                                                             | ELU           | AdaGrad       |               |     |                 |                 |               |          |               |               |            |         |                   |                   |               |         |  |  |            |      |
| Fully Connected                                                                                | Average U-pooling                                                                                                                                                                                                                                                                                                                                                                                                                                         | Tanh          | RMSProp       |               |     |                 |                 |               |          |               |               |            |         |                   |                   |               |         |  |  |            |      |
|                                                                                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Sigmoid       | Adam          |               |     |                 |                 |               |          |               |               |            |         |                   |                   |               |         |  |  |            |      |
| a. Fully Connected: Linear functions between the input and the output.                         | <table border="1"> <tr><td>Max Pooling</td><td>Normalization</td><td>MSE – L2 Loss</td><td>SCD</td></tr> <tr><td>Average Pooling</td><td>Local Response</td><td>MAE – L1 Loss</td><td>Momentum</td></tr> <tr><td>Max U-pooling</td><td>Normalization</td><td>Huber Loss</td><td>AdaGrad</td></tr> <tr><td>Average U-pooling</td><td></td><td>Cross Entropy</td><td>RMSProp</td></tr> <tr><td></td><td></td><td>Hinge Loss</td><td>Adam</td></tr> </table> | Max Pooling   | Normalization | MSE – L2 Loss | SCD | Average Pooling | Local Response  | MAE – L1 Loss | Momentum | Max U-pooling | Normalization | Huber Loss | AdaGrad | Average U-pooling |                   | Cross Entropy | RMSProp |  |  | Hinge Loss | Adam |
| Max Pooling                                                                                    | Normalization                                                                                                                                                                                                                                                                                                                                                                                                                                             | MSE – L2 Loss | SCD           |               |     |                 |                 |               |          |               |               |            |         |                   |                   |               |         |  |  |            |      |
| Average Pooling                                                                                | Local Response                                                                                                                                                                                                                                                                                                                                                                                                                                            | MAE – L1 Loss | Momentum      |               |     |                 |                 |               |          |               |               |            |         |                   |                   |               |         |  |  |            |      |
| Max U-pooling                                                                                  | Normalization                                                                                                                                                                                                                                                                                                                                                                                                                                             | Huber Loss    | AdaGrad       |               |     |                 |                 |               |          |               |               |            |         |                   |                   |               |         |  |  |            |      |
| Average U-pooling                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Cross Entropy | RMSProp       |               |     |                 |                 |               |          |               |               |            |         |                   |                   |               |         |  |  |            |      |
|                                                                                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Hinge Loss    | Adam          |               |     |                 |                 |               |          |               |               |            |         |                   |                   |               |         |  |  |            |      |



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

  - 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
  - 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.
  - Normalization:** Usually used just before the activation functions to limit the unbounded activation from increasing the output layer values too high
  - Local Response Normalization LRN:** A non-trainable layer that square-normalizes the pixel values in a feature map within a local neighborhood.
  - 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.

  - Non-parametric/Static functions:** Linear, ReLU
  - Parametric functions:** ELU, tanh, sigmoid, Leaky ReLU
  - Bounded functions:** tanh, sigmoid

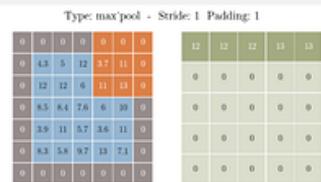
4. **Loss function:** Quantifies how far off the CNN prediction is from the actual labels.

  - Regression Loss Functions:** MAE, MSE, Huber loss
  - Classification Loss Functions:** Cross entropy, Hinge loss

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	7.8	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

Type: max'pool - Stride: 1 Padding: 1

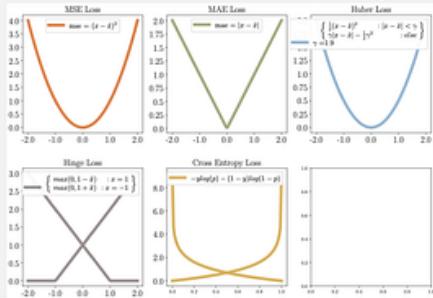
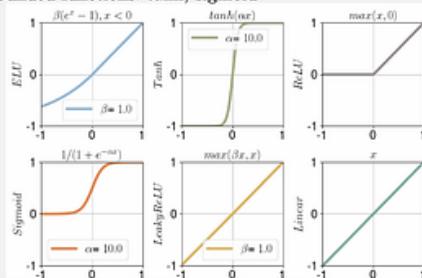
Input      Stride      Padding      Output



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

- a. Regression Loss Functions: MAE, MSE, Huber loss
  - b. Classification Loss Functions: Cross entropy, Hinge loss



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

Convolutional Neural Network — Image by Author

## b) Example Questions:

1. What is CNN?
  2. Explain the difference between the convolutional layer and transposed convolutional layer.
  3. What are some of the loss functions used for classification?

## c) Detailed Article:

- [What is a Convolutional Neural Network?](#)

## Famous DNNs in Machine Learning

### a) Overview:

Cheat Sheet – Famous CNNs																																											
<b>AlexNet – 2012</b>																																											
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 normalization.																																											
<b>VGGNet – 2014</b>																																											
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.																																											
<b>ResNet – 2015</b>																																											
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																																							
<b>Inception – 2014</b>																																											
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.																																											
<table border="1"> <thead> <tr> <th colspan="6">Comparison</th> </tr> <tr> <th>Network</th> <th>Year</th> <th>Salient Feature</th> <th>top5 accuracy</th> <th>Parameters</th> <th>FLOP</th> </tr> </thead> <tbody> <tr> <td>AlexNet</td> <td>2012</td> <td>Deeper</td> <td>84.70%</td> <td>62M</td> <td>1.5B</td> </tr> <tr> <td>VGGNet</td> <td>2014</td> <td>Fixed-size kernels</td> <td>92.30%</td> <td>138M</td> <td>19.6B</td> </tr> <tr> <td>Inception</td> <td>2014</td> <td>Wider - Parallel kernels</td> <td>93.30%</td> <td>6.4M</td> <td>2B</td> </tr> <tr> <td>ResNet-152</td> <td>2015</td> <td>Shortcut connections</td> <td>95.51%</td> <td>60.3M</td> <td>11B</td> </tr> </tbody> </table>								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
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																																						
Source: <a href="https://www.cheatsheets.aqeel-anwar.com">https://www.cheatsheets.aqeel-anwar.com</a>																																											

Famous CNNs — Image by Author

## b) Example Questions:

1. How does the ResNet network address the problem of vanishing gradient?
2. What is one of the main key features of the Inception Network?
3. What are shortcut connections in the ResNet network?

## c) Detailed Articles:

- [Difference between AlexNet, VGGNet, ResNet, and Inception](#)

# Autoencoder and Variational Autoencoder

## a) Overview:

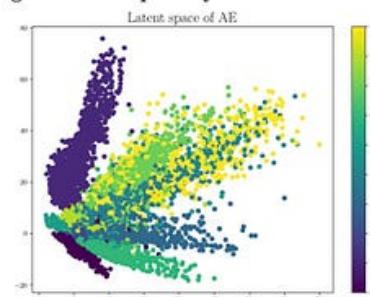
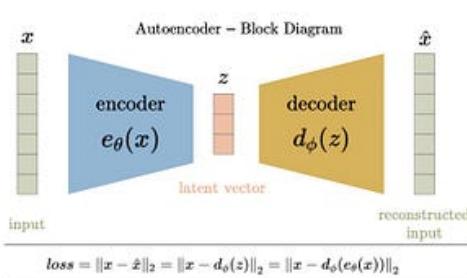
### Cheat Sheet – Autoencoder & Variational Autoencoder

#### Context – Data Compression

- Data compression is an essential phase in training a network. The idea is to compress the data so that the same amount of information can be represented by fewer bits.

#### Auto Encoder (AE)

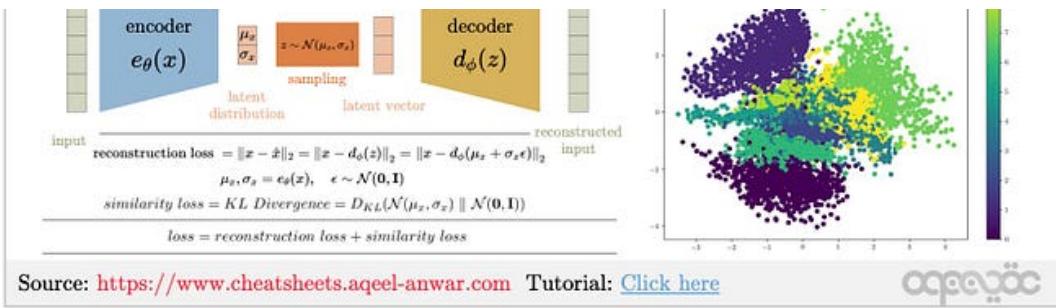
- Autoencoder is used to learn efficient embeddings of unlabeled data for a given network configuration. It consists of two parts, an encoder, and a decoder.
- The encoder compresses the data from a higher-dimensional space to a lower-dimensional space (also called the latent space), while the decoder converts the latent space back to higher-dimensional space.
- The entire encoder-decoder architecture is collectively trained on the loss function which encourages that the input is reconstructed at the output. Hence the loss function is the mean squared error between the encoder input and the decoder output.
- The latent variable is not regularized. Picking a random latent variable will generate garbage output.
- Latent variable is deterministic values and the space lacks the generative capability



#### Variational Auto Encoder (VAE)

- Variational autoencoder addresses the issue of non-regularized latent space in autoencoder and provides the generative capability to the entire space.
- Instead of outputting the vectors in the latent space, the encoder of VAE outputs parameters of a pre-defined distribution in the latent space for every input.
- The VAE then imposes a constraint on this latent distribution forcing it to be a normal distribution.
- The latent variable in the compressed form is mean and variance
- The training loss of VAE is defined as the sum of the reconstruction loss and the similarity loss (the KL divergence between the unit gaussian and decoder output distribution).
- The latent variable is smooth and continuous i.e., random values of latent variable generates meaningful output at the decoder, hence the latent space has generative capabilities.
- The input of the decoder is sampled from a gaussian with mean/variance of the output of encoder.





## b) Example Questions:

1. What is an Autoencoder?
2. Is the latent space of Autoencoder regularised?
3. What is the loss function for a variational autoencoder?
4. What's the difference between an Autoencoder and Variational Autoencoder?

## c) Detailed Articles:

- [Difference between AutoEncoder \(AE\) and Variational AutoEncoder \(VAE\)](#)

## Summary

This article provides a list of cheat sheets covering important topics for a Machine learning interview followed by some example questions. The list of topics and the number of cheat sheets are constantly being added to the article.

**If this article was helpful to you, feel free to clap, share and respond to it. If you want to learn more about Machine Learning and Data Science, follow me @ or connect with me on [LinkedIn](#).**