SECA4002 – DEEP LEARNING NEURAL NETWORKS

Dr. V. Vedanarayanan B.E., M.E., PhD
Course Co-ordinator
Assistant Professor, Department of ECE,
School of Electrical and Electronics
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

Course Outcomes

SECA4002 – DEEP LEARNING NEURAL NETWORKS

At The end of this Course, Student will be able to

CO1	Select suitable model parameters for different machine learning techniques
CO2	Evaluate the performance of existing deep learning models for various applications
CO3	Realign high dimensional data using reduction techniques
CO4	Analyze the performance of various optimization and generalization techniques in deep learning
CO5	Modify the existing architectures for domain specific applications
CO6	Develop a real time application using deep learning neural networks

Course Objectives

SECA4002 – DEEP LEARNING NEURAL NETWORKS

Course Objectives:

- To present the mathematical, statistical and computational challenges of building neural networks
- To study the concepts of deep learning
- To introduce dimensionality reduction techniques
- To enable the students to know deep learning techniques to support real-time applications
- To examine the case studies of deep learning techniques

SECA4002 – DEEP LEARNING NEURAL NETWORKS

Detailed Syllabus:

UNIT 3: DIMENTIONALITY REDUCTION

Linear (PCA, LDA) and manifolds, metric learning - Auto encoders and dimensionality reduction in networks - Introduction to Convnet - Architectures – AlexNet, VGG, Inception, ResNet - Training a Convnet: weights initialization, batch normalization, hyperparameter optimization.

SECA4002 – DEEP LEARNING NEURAL NETWORKS

Recommended Text Books/ Reference Books

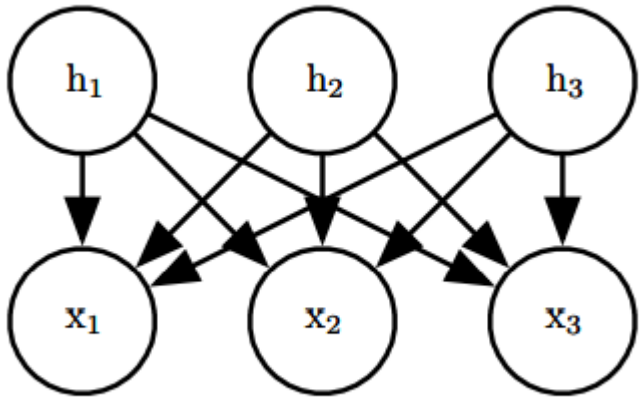
- Cosma Rohilla Shalizi, Advanced Data Analysis from an Elementary Point of View, 2015.
- Deng & Yu, Deep Learning: Methods and Applications, Now Publishers, 2013.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.
- Michael Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

Linear factor Models

- linear factor models are used as building blocks of mixture modelsor of larger, deep probabilistic models
- A linear factor model is defined by the use of a stochastic linear decoder function that generates x by adding noise to a linear transformation of h
- allow us to discover explanatory factors that have a simple joint distribution
- A linear factor model describes the data-generation process as follows.

we sample the explanatory factors h from a distribution

$$h \sim p(h)$$



$$\mathbf{x} = \boldsymbol{W}\mathbf{h} + \boldsymbol{b} + \text{noise}$$

From the above Diagram, the observed data vector x is obtained by a linear combination of independent latent factors h, plus some noise

Linear factor Models

- Different models, such as probabilistic PCA, factor analysis or ICA, make different choices about the form of the noise and of the prior p(h).

  where p(h) is a factorial distribution, with p(h) = Π i p(hi),

- Next we sample the real-valued observable variables given the factors x = W h + b + noise,

  where the noise is typically Gaussian and diagonal

Dimensionality Reduction:

- High dimensionality is challenging and redundant
- It is natural to try to reduce dimensionality
- We reduce the dimensionality by feature combination i.e., combine old features X to create new features Y

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \rightarrow f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}\right) = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} = y \quad \text{with } k < d$$

Dimensionality Reduction

For example, $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 + x_2 \\ x_3 + x_4 \end{bmatrix} = \boldsymbol{y}$

Ideally, the new vector ***y*** should retain from ***x*** all information important for classification

- An important machine learning method for dimensionality reduction is called **Principal Component Analysis**.

Principle Component Analysis - PCA

- It is a method that uses simple matrix operations from linear algebra and statistics to calculate a projection of the original data into the same number or fewer dimensions.
- t can be thought of as a projection method where data with m-columns (features) is projected into a subspace with m or fewer columns, whilst retaining the essence of the original data.
- PCA is an operation applied to a dataset, represented by an n x m matrix A that results in a projection of A which we will call B.

PCA is ...

- A backbone of modern data analysis.
- A black box that is widely used but poorly understood.

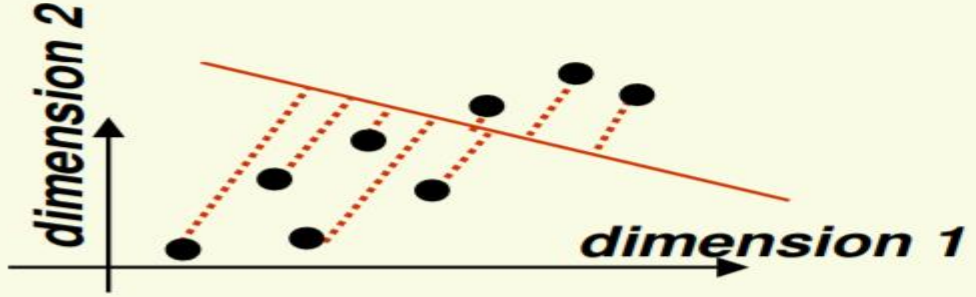
Image Source: http://www.sci.utah.edu/~shireen/pdfs/tutorials/Elhabian_PCA09.pdf

Principle Component Analysis - PCA

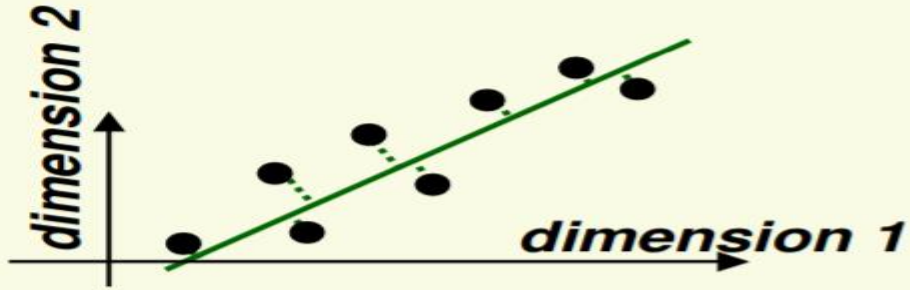
- It is a mathematical tool from applied linear algebra.
- It is a simple, non-parametric method of extracting relevant information from confusing data sets.
- It provides a roadmap for how to reduce a complex data set to a lower dimension.
- Main idea: seek most accurate data representation in a lower dimensional space

Example in 2-D

- Project data to 1-D subspace (a line) which minimize the projection error



large projection errors, bad line to project to



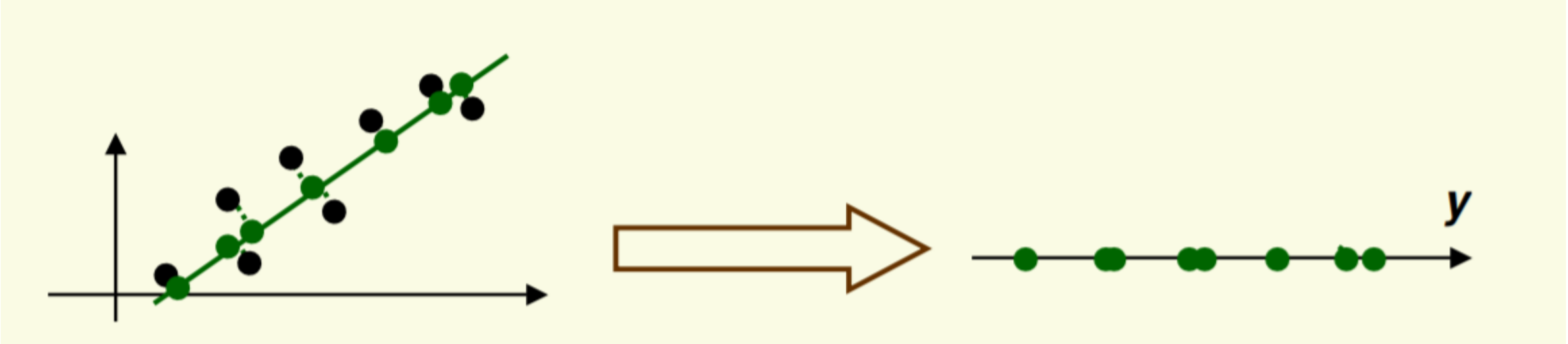
small projection errors, good line to project to

Notice that the the good line to use for projection lies in the direction of largest variance

Image Source: https://www.csd.uwo.ca/~oveksler/Courses/CS434a_541a/Lecture7.pdf

Principle Component Analysis - PCA

- After the data is projected on the best line, need to transform the coordinate system to get 1D representation for vector y

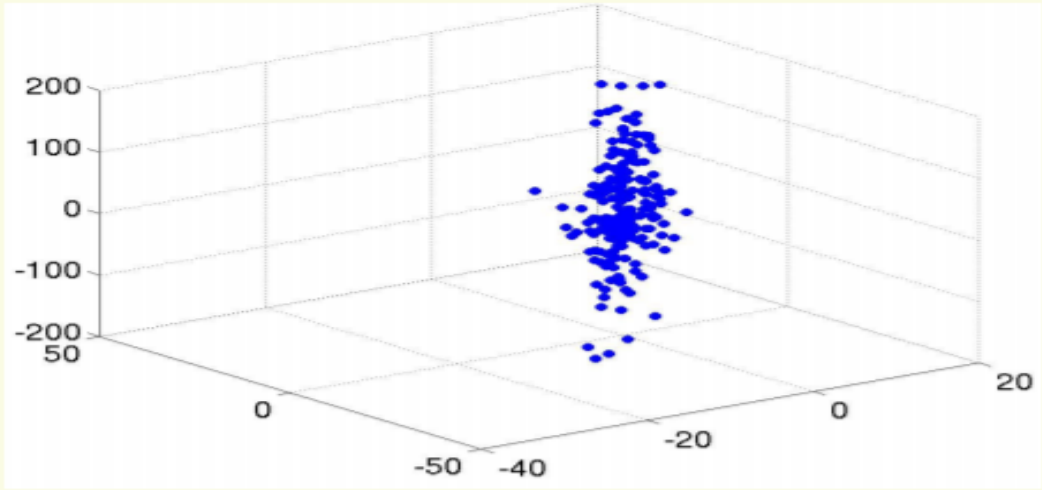


- Note that new data y has the same variance as old data x in the direction of the green line PCA preserves largest variances in the data.

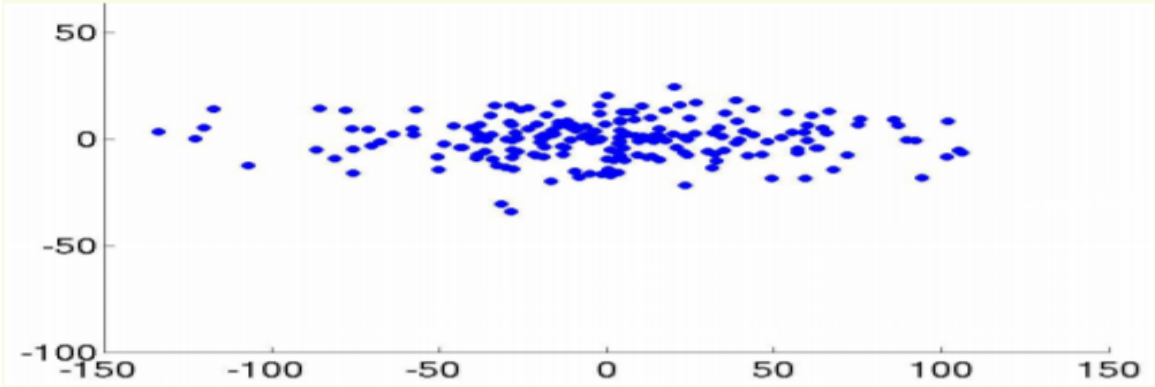
Image Source: https://www.csd.uwo.ca/~oveksler/Courses/CS434a_541a/Lecture7.pdf

Principle Component Analysis - PCA

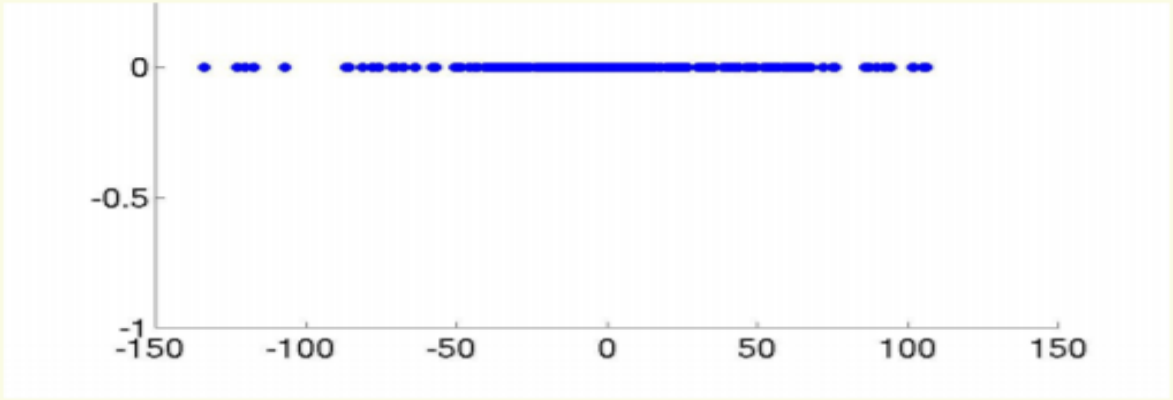
PCA: Approximation of Elliptical Cloud in 3D



best 2D approximation

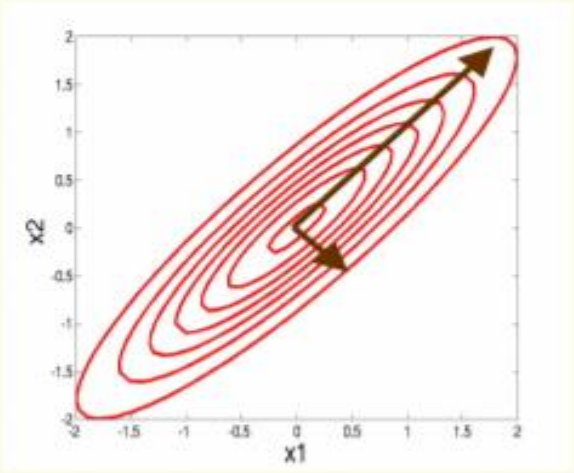


best 1D approximation



Principle Component Analysis - PCA

- What is the direction of largest variance in data?
- Recall that if $\boldsymbol{x}$ has multivariate distribution $N(\mu,\Sigma)$, direction of largest variance is given by eigenvector corresponding to the largest eigenvalue of Σ



- This is a hint that we should be looking at the covariance matrix of the data (note that PCA can be applied to distributions other than Gaussian)

Principle Component Analysis -STEPs

Steps for PCA:

- Start with the data for n observations on p variables.
- Form a matrix of size n x p with deviations from mean for each of the variables.
- Calculate the covariance matrix (p x p).
- Calculate the eigenvalues and eigenvectors of the covariance matrix.
- Choose principal components and form a feature vector.
- Derive the new data set.

Principle Component Analysis -STEPs

- **NOTES**
  - Since PCA uses the eigenvectors of the covariance matrix Σ_x, it is able to find the independent axes of the data under the unimodal Gaussian assumption
    - For non-Gaussian or multi-modal Gaussian data, PCA simply de-correlates the axes
  - The main limitation of PCA is that it does not consider class separability since it does not take into account the class label of the feature vector
    - PCA simply performs a coordinate rotation that aligns the transformed axes with the directions of maximum variance
    - **There is no guarantee that the directions of maximum variance will contain good features for discrimination**
- **Historical remarks**
  - Principal Components Analysis is the oldest technique in multivariate analysis
  - PCA is also known as the Karhunen-Loève transform (communication theory)
  - PCA was first introduced by Pearson in 1901, and it experienced several modifications until it was generalized by Loève in 1963

Principle Component Analysis -PCA

- By finding the eigenvalues and eigenvectors of the covariance matrix, we find that the eigenvectors with the largest eigenvalues correspond to the dimensions that have the strongest correlation in the dataset. This is the principal component.
- PCA is a useful statistical technique that has found application in: – fields such as face recognition and image compression – finding patterns in data of high dimension.

PCA Theorem

Let $x_1\ x_2\ \dots\ x_n$ be a set of n N x 1 vectors and let $\bar{x}$ be their average:

$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iN} \end{bmatrix} \qquad \bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{i=n} \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iN} \end{bmatrix}$$

Principle Component Analysis

Let X be the N x n matrix with columns
$x_1 - \bar{x}, x_2 - \bar{x}, \ldots x_n - \bar{x}$:

$$X = \left[\begin{array}{cccc} x_1 - \bar{x} & x_2 - \bar{x} & \cdots & x_n - \bar{x} \end{array} \right]$$

Note: subtracting the mean is equivalent to translating the coordinate system to the location of the mean.

Principle Component Analysis

Let $Q = X X^T$ be the N x N matrix:

$$Q = XX^T = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix} \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ (\mathbf{x}_2 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \end{bmatrix}$$

Notes:

1. Q is square
2. Q is symmetric
3. Q is the *covariance* matrix [aka scatter matrix]
4. Q can be very large (in vision, N is often the number of pixels in an image!)

Principle Component Analysis

Theorem:

Each x_j can be written as: $$\mathbf{x}_j = \bar{\mathbf{x}} + \sum_{i=1}^{i=n} g_{ji}\mathbf{e}_i$$

where e_i are the n eigenvectors of Q with non-zero eigenvalues.

Notes:

1. The eigenvectors e_1 e_2 ... e_n span an ***eigenspace***
2. e_1 e_2 ... e_n are N x 1 orthonormal vectors (directions in N-Dimensional space)
3. The scalars g_{ji} are the coordinates of x_j in the space.

$$g_{ji} = (\mathbf{x}_j - \bar{\mathbf{x}}).\mathbf{e}_i$$

Principle Component Analysis

Using PCA to Compress Data

- Expressing x in terms of e_1 ... e_n has not changed the size of the data
- However, if the points are highly correlated many of the coordinates of x will be zero or closed to zero.

note: this means they lie in a lower-dimensional linear subspace

Principle Component Analysis

Using PCA to Compress Data

- Sort the eigenvectors e_i according to their eigenvalue:

$$\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n$$

- Assuming that $\lambda_i \approx 0$ if $i > k$

- Then

$$\mathbf{x}_j \approx \bar{\mathbf{x}} + \sum_{i=1}^{i=k} g_{ji}\mathbf{e}_i$$

Principle Component Analysis - Advantages

1. Removes Correlated Features:

In a real world scenario, it is very common that we get thousands of features in our dataset. You cannot run your algorithm on all the features as it will reduce the performance of your algorithm and it will not be easy to visualize that many features in any kind of graph. Hence the data set should be reduced.

We need to find out the correlation among the features (correlated variables). Finding correlation manually in thousands of features is nearly impossible, frustrating and time-consuming. PCA performs this task effectively

After implementing the PCA on your dataset, all the Principal Components are independent of one another. There is no correlation among them.

2. Improves Algorithm Performance:

With so many features, the performance of your algorithm will drastically degrade. PCA is a very common way to speed up your Machine Learning algorithm by getting rid of correlated variables which don't contribute in any decision making. The training time of the algorithms reduces significantly with less number of features.

So, if the input dimensions are too high, then using PCA to speed up the algorithm is a reasonable choice.

Principle Component Analysis - Advantages & Disadvantages

3. Reduces Overfitting: Overfitting mainly occurs when there are too many variables in the dataset. So, PCA helps in overcoming the overfitting issue by reducing the number of features.

4. Improves Visualization: It is very hard to visualize and understand the data in high dimensions. PCA transforms a high dimensional data to low dimensional data (2 dimension) so that it can be visualized easily.

We can use 2D Scree Plot to see which Principal Components result in high variance and have more impact as compared to other Principal Components.

Disadvantages of Principal Component Analysis

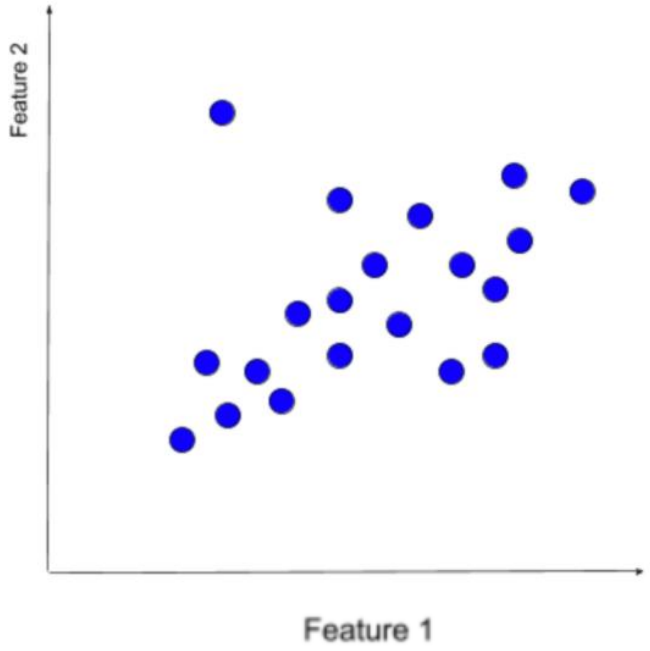
1. Independent variables become less interpretable: After implementing PCA on the dataset, your original features will turn into Principal Components. Principal Components are the linear combination of your original features. Principal Components are not as readable and interpretable as original features.

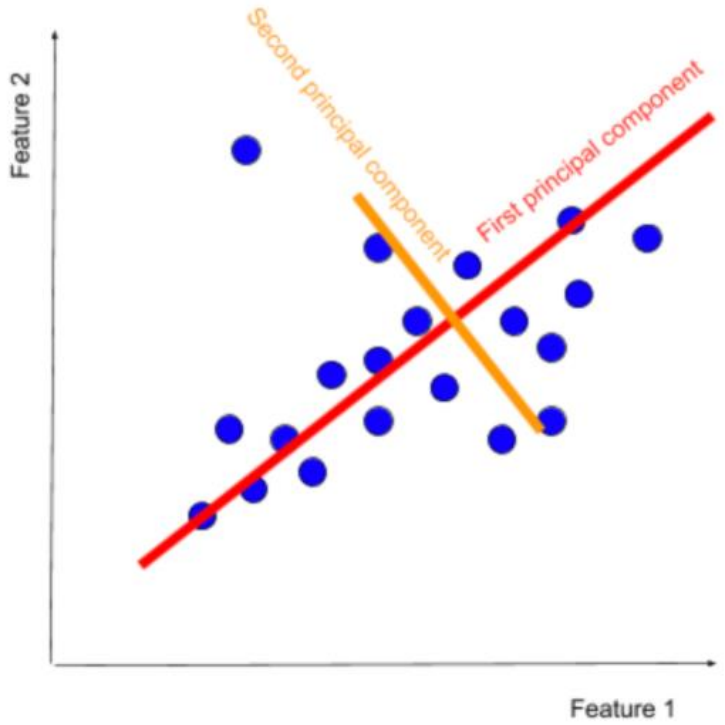
2. Data standardization is must before PCA: You must standardize your data before implementing PCA, otherwise PCA will not be able to find the optimal Principal Components.

Principle Component Analysis - Advantages & Disadvantages

Disadvantages of Principal Component Analysis

3. Information Loss: Although Principal Components try to cover maximum variance among the features in a dataset, if we don't select the number of Principal Components with care, it may miss some information as compared to the original list of features.



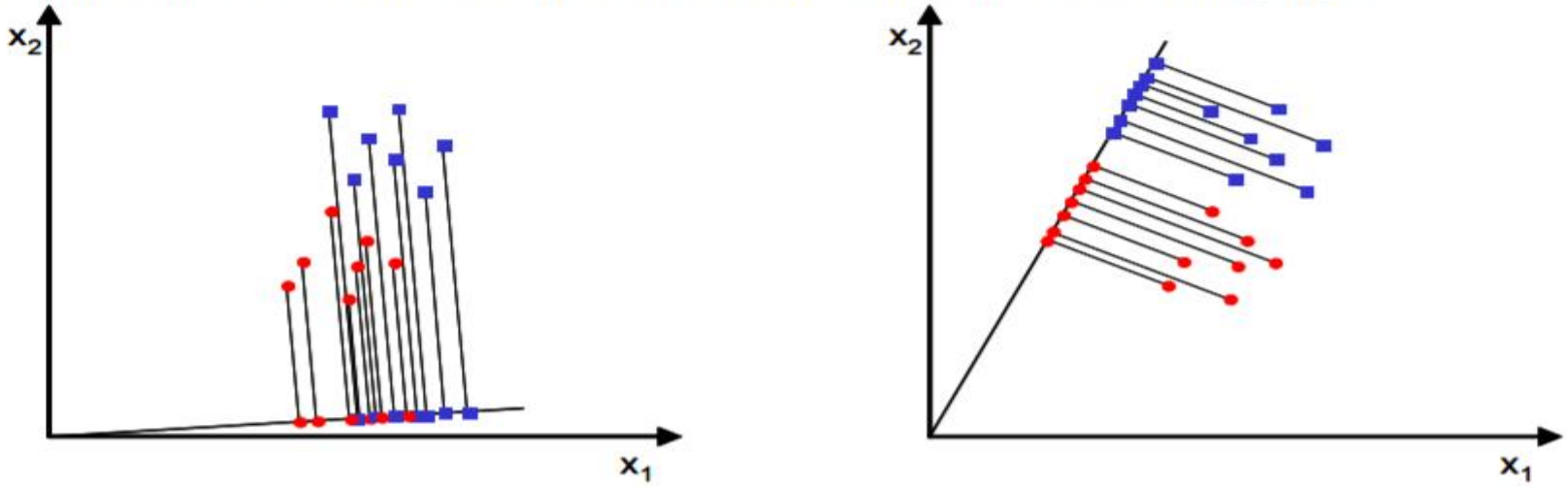


Linear Discriminative Analysis - LDA

- **The objective of LDA is to perform dimensionality reduction while preserving as much of the class discriminatory information as possible**
  - Assume we have a set of D-dimensional samples $\{x^{(1}, x^{(2}, \ldots, x^{(N}\}$, N_1 of which belong to class ω_1, and N_2 to class ω_2. We seek to obtain a scalar $\boldsymbol{y}$ by projecting the samples $\boldsymbol{x}$ onto a line

$$y = w^T x$$

  - Of all the possible lines we would like to select the one that maximizes the separability of the scalars
    - This is illustrated for the two-dimensional case in the following figures



Content & Image Source: http://www.facweb.iitkgp.ac.in/~sudeshna/courses/ml08/lda.pdf

Linear Discriminative Analysis - LDA

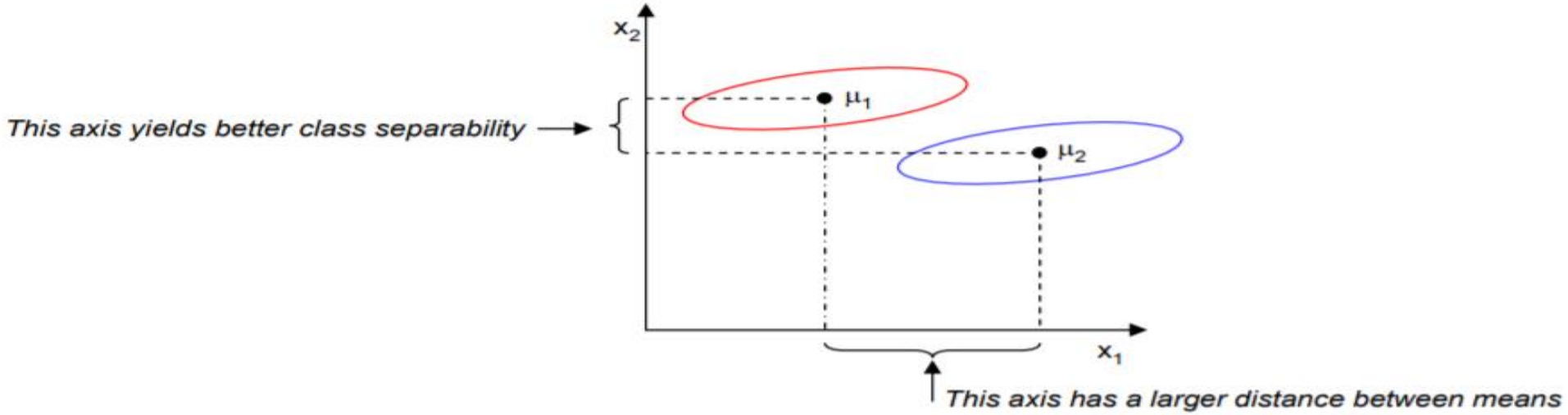
- **In order to find a good projection vector, we need to define a measure of separation between the projections**
  - The mean vector of each class in ***x*** and ***y*** feature space is

$$\mu_i = \frac{1}{N_i}\sum_{x\in\omega_i} x \quad \text{and} \quad \tilde{\mu}_i = \frac{1}{N_i}\sum_{y\in\omega_i} y = \frac{1}{N_i}\sum_{x\in\omega_i} w^T x = w^T \mu_i$$

  - We could then choose the distance between the projected means as our objective function

$$J(w) = \left|\tilde{\mu}_1 - \tilde{\mu}_2\right| = \left|w^T(\mu_1 - \mu_2)\right|$$

  - However, the distance between the projected means is not a very good measure since it does not take into account the standard deviation within the classes



Content & Image Source: http://www.facweb.iitkgp.ac.in/~sudeshna/courses/ml08/lda.pdf

Linear Discriminative Analysis - LDA

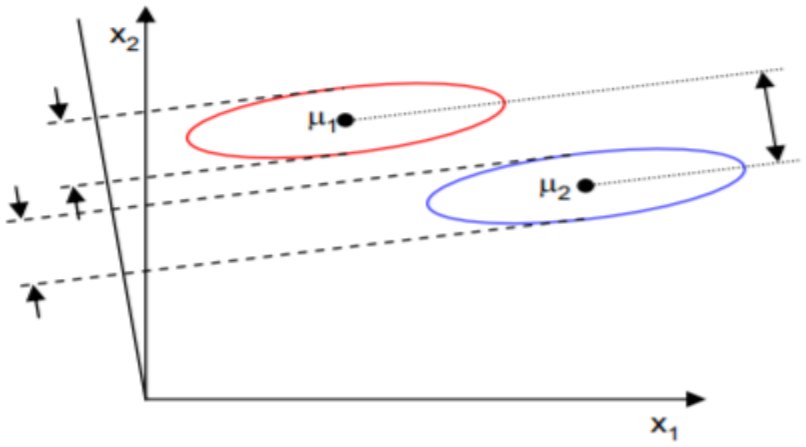
- **The solution proposed by Fisher is to maximize a function that represents the difference between the means, normalized by a measure of the within-class scatter**
  - For each class we define the scatter, an equivalent of the variance, as

$$\tilde{s}_i^2 = \sum_{y \in \omega_i} (y - \tilde{\mu}_i)^2$$

    - where the quantity $\left(\tilde{s}_1^2 + \tilde{s}_2^2\right)$ is called the within-class scatter of the projected examples
  - The Fisher linear discriminant is defined as the linear function $\mathbf{w}^T\mathbf{x}$ that maximizes the criterion function

$$J(w) = \frac{\left|\tilde{\mu}_1 - \tilde{\mu}_2\right|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

  - Therefore, we will be looking for a projection where examples from the same class are projected very close to each other and, at the same time, the projected means are as farther apart as possible



Content & Image Source: http://www.facweb.iitkgp.ac.in/~sudeshna/courses/ml08/lda.pdf

Linear Discriminative Analysis - LDA

- In order to find the optimum projection w*, we need to express J(w) as an explicit function of w
- We define a measure of the scatter in multivariate feature space **x**, which are scatter matrices

$$S_i = \sum_{x \in \omega_i} (x - \mu_i)(x - \mu_i)^T$$

$$S_1 + S_2 = S_W$$

  - where S_W is called the **within-class scatter matrix**
- The scatter of the projection **y** can then be expressed as a function of the scatter matrix in feature space **x**

$$\tilde{s}_i^2 = \sum_{y \in \omega_i} (y - \tilde{\mu}_i)^2 = \sum_{x \in \omega_i} (w^T x - w^T \mu_i)^2 = \sum_{x \in \omega_i} w^T (x - \mu_i)(x - \mu_i)^T w = w^T S_i w$$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = w^T S_W w$$

- Similarly, the difference between the projected means can be expressed in terms of the means in the original feature space

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (w^T \mu_1 - w^T \mu_2)^2 = w^T \underbrace{(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T}_{S_B} w = w^T S_B w$$

  - The matrix S_B is called the **between-class scatter**. Note that, since S_B is the outer product of two vectors, its rank is at most one
- We can finally express the Fisher criterion in terms of S_W and S_B as

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

Content & Image Source: http://www.facweb.iitkgp.ac.in/~sudeshna/courses/ml08/lda.pdf

Linear Discriminative Analysis - LDA

- To find the maximum of J(w) we derive and equate to zero

$$\frac{d}{dw}[J(w)] = \frac{d}{dw}\left[\frac{w^T S_B w}{w^T S_W w}\right] = 0 \Rightarrow$$

$$\Rightarrow [w^T S_W w]\frac{d[w^T S_B w]}{dw} - [w^T S_B w]\frac{d[w^T S_W w]}{dw} = 0 \Rightarrow$$

$$\Rightarrow [w^T S_W w]2S_B w - [w^T S_B w]2S_W w = 0$$

- Dividing by $w^T S_W w$

$$\frac{[w^T S_W w]}{[w^T S_W w]}S_B w - \frac{[w^T S_B w]}{[w^T S_W w]}S_W w = 0 \Rightarrow$$

$$\Rightarrow S_B w - J S_W w = 0 \Rightarrow$$

$$\Rightarrow S_W^{-1} S_B w - J w = 0$$

- Solving the generalized eigenvalue problem ($S_W^{-1} S_B w = J w$) yields

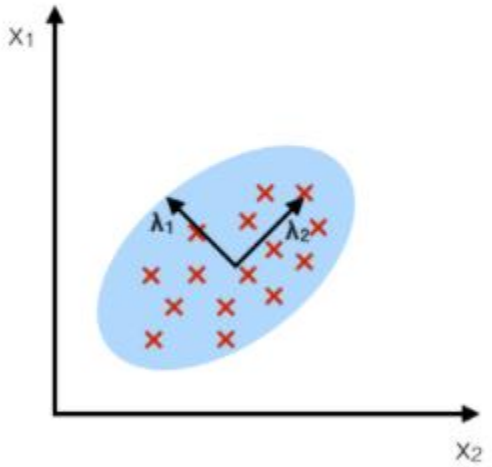
$$w^* = \underset{w}{\text{argmax}}\left\{\frac{w^T S_B w}{w^T S_W w}\right\} = S_W^{-1}(\mu_1 - \mu_2)$$

  - This is know as **Fisher's Linear Discriminant** (1936), although it is not a discriminant but rather a specific choice of direction for the projection of the data down to one dimension

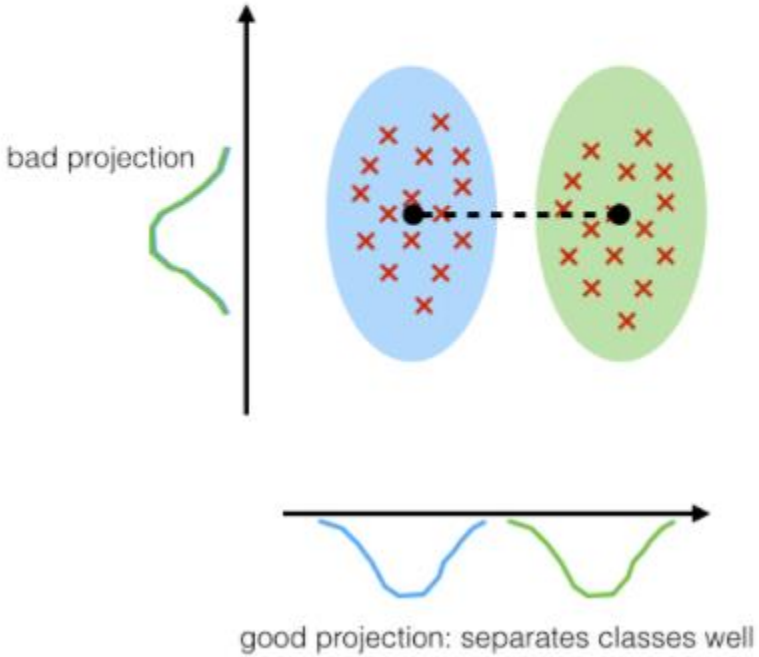
Content & Image Source: http://www.facweb.iitkgp.ac.in/~sudeshna/courses/ml08/lda.pdf

PCA Vs LDA

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



Content & Image Source: https://sebastianraschka.com/Articles/2014_python_lda.html

LDA in 5simple steps

1. Compute the d-dimensional mean vectors for the different classes from the dataset.
2. Compute the scatter matrices (in-between-class and within-class scatter matrix).
3. Compute the eigenvectors ($\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_d$) and corresponding eigenvalues ($\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_d$) for the scatter matrices.
4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimensional matrix $\boldsymbol{W}$ (where every column represents an eigenvector).
5. Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication: $\boldsymbol{Y} = \boldsymbol{X} \times \boldsymbol{W}$ (where $\boldsymbol{X}$ is a $n \times d$-dimensional matrix representing the n samples, and $\boldsymbol{y}$ are the transformed $n \times k$-dimensional samples in the new subspace).

Content & Image Source: https://sebastianraschka.com/Articles/2014_python_lda.html

LDA – Pros & Cons

Advantages of LDA:

1. Simple prototype classifier: Distance to the class mean is used, it's simple to interpret.
2. Decision boundary is linear: It's simple to implement and the classification is robust.
3. Dimension reduction: It provides informative low-dimensional view on the data, which is both useful for visualization and feature engineering.

Shortcomings of LDA:

1. Linear decision boundaries may not adequately separate the classes. Support for more general boundaries is desired.
2. In a high-dimensional setting, LDA uses too many parameters. A regularized version of LDA is desired.
3. Support for more complex prototype classification is desired.

Content & Image Source: https://yangxiaozhou.github.io/data/2019/10/02/linear-discriminant-analysis.html

Manifold Learning

- Manifold learning for dimensionality reduction has recently gained much attention to assist image processing tasks such as **segmentation, registration, tracking, recognition, and computational anatomy.**
- The drawbacks of PCA in handling dimensionality reduction problems for **non-linear weird** and **curved shaped surfaces** necessitated development of more advanced algorithms like **Manifold Learning**.
- There are different variants of Manifold Learning that solves the problem of **reducing data dimensions** and feature-sets obtained from real world problems representing uneven weird surfaces by sub-optimal data representation.
- This kind of data representation **selectively chooses data points** from a low-dimensional manifold that is embedded in a high-dimensional space in an attempt to **generalize linear frameworks** like PCA.

Content & Image Source: https://techairesearch.com/dimensionality-reduction-pca-ica-and-manifold-learning/

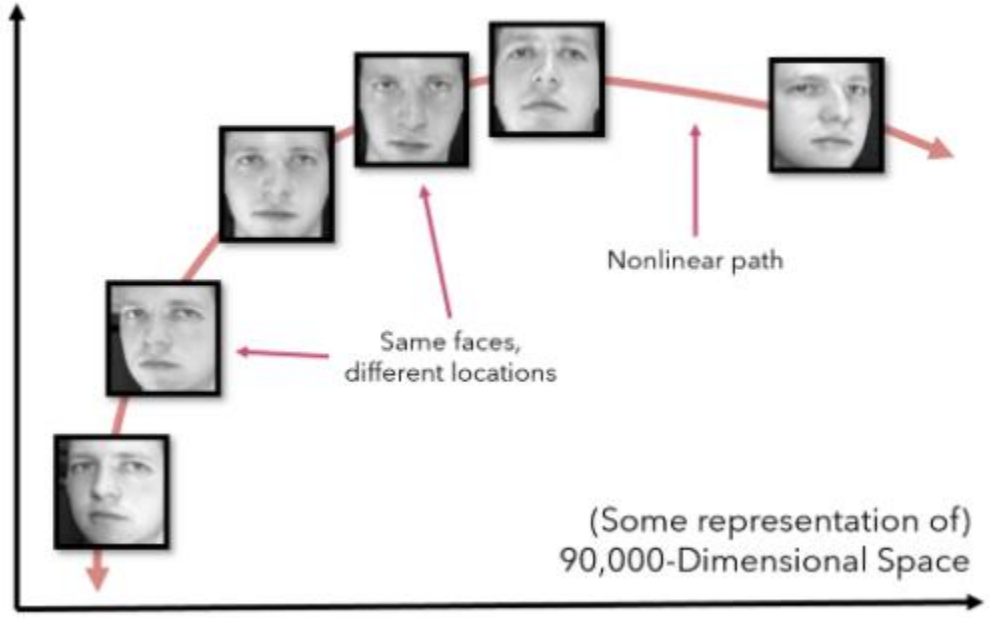
Manifold Learning – Continued.,

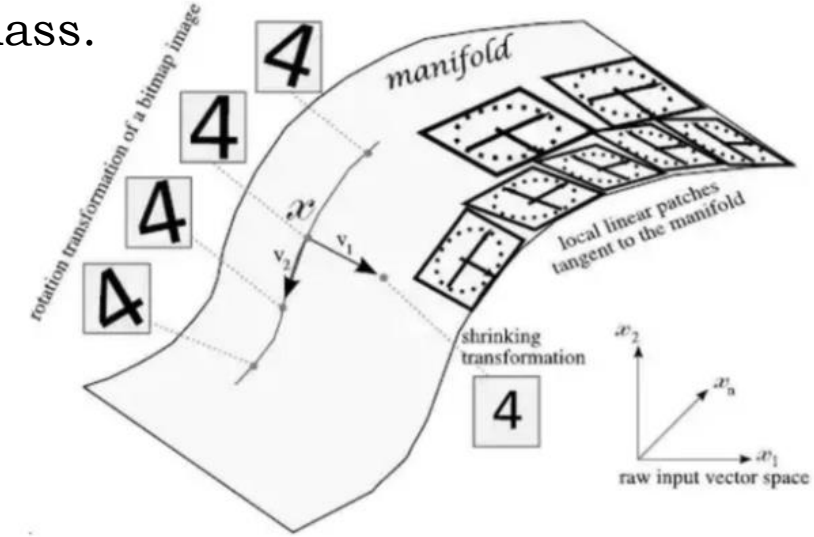
- Manifolds give a look of **flat** and **featureless** space that behaves like Euclidean space. Manifold learning problems are **unsupervised** where it learns the **high-dimensional structure** of the data from the data itself, without the use of **predetermined classifications** and **loss of importance of information** regarding some characteristic of the original variables.
- The goal of the manifold-learning algorithms is to recover the **original domain structure**, up to some **scaling** and rotation. The nonlinearity of these algorithms allows them to reveal the domain structure even when the manifold is not **linearly embedded**. It uses some **scaling** and **rotation** for this purpose.
- *Manifold learning algorithms are divided in to two categories:*
  - **Global methods:** Allows high-dimensional data to be mapped from high-dimensional to low-dimensional such that the global properties are preserved. Examples include **Multidimensional Scaling (MDS), Isomaps** covered in the following sections.
  - **Local methods:** Allows high-dimensional data to be mapped to low dimensional such that local properties are preserved. Examples are **Locally linear embedding (LLE), Laplacian eigenmap (LE), Local tangent space alignment (LSTA), Hessian Eigenmapping (HLLE)**

Content & Image Source: https://techairesearch.com/dimensionality-reduction-pca-ica-and-manifold-learning/

Manifold Learning – Continued.,

consider 300 by 300 pixel headshots. Under perfect conditions, each of the images would be centered perfectly, but in reality, there are many additional degrees of freedom to consider, such as lighting or the tilt of the face. If we were to treat a headshot as a point in 90,000 dimensional space, changing various effects like tilting the head or looking in a different direction move it nonlinearly through space, even though it is the same object with the same class.





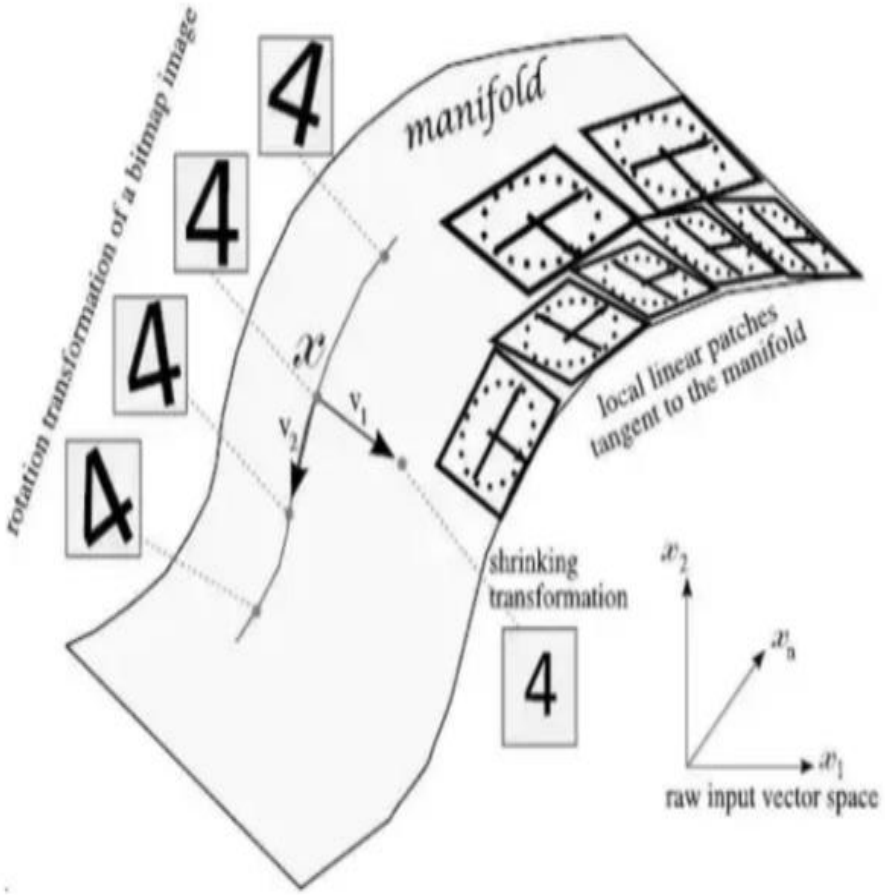
PCA attempts to create several linear hyperplanes to represent dimensions.

Manifold learning attempts to learn manifolds, which are smooth, curved surfaces within the multidimensional space. As in the image above, these are often formed by subtle transformations on an image that would otherwise fool PCA.

Content & Image Source: https://towardsdatascience.com/manifold-learning-t-sne-lle-isomap-made-easy-42cfd61f5183/

Manifold Learning – Continued.,

- PCA attempts to create several linear hyperplanes to represent dimensions.
- **Manifold learning attempts to learn manifolds, which are smooth, curved surfaces within the multidimensional space. As in the image above, these are often formed by subtle transformations on an image that would otherwise fool PCA.**



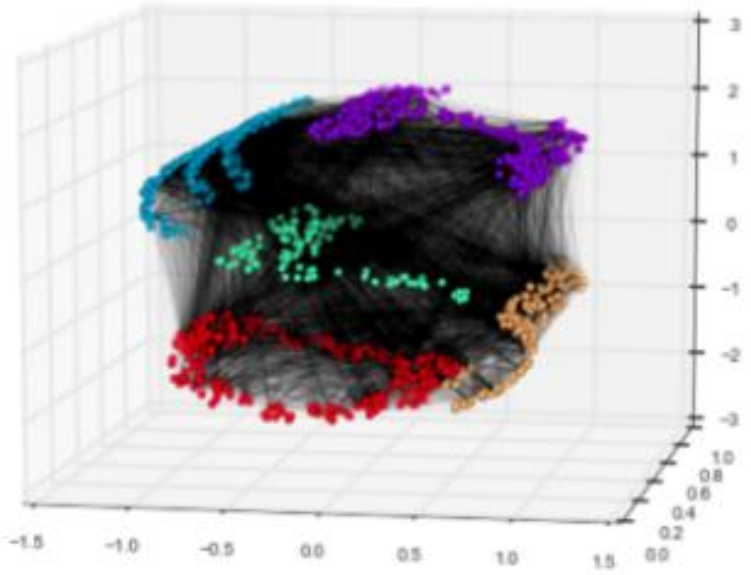
Content & Image Source: https://towardsdatascience.com/manifold-learning-t-sne-lle-isomap-made-easy-42cfd61f5183/

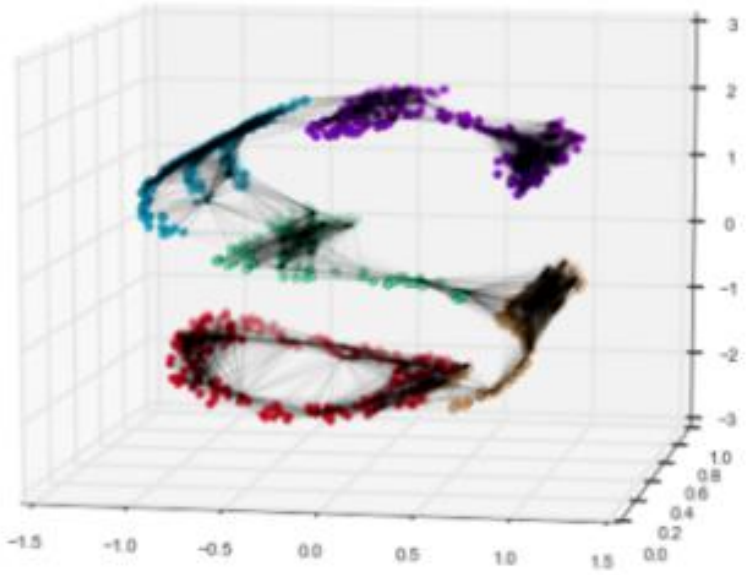
Manifold Learning – Continued.,

- ‘local linear patches’ that are tangent to the manifold can be extracted.
- These patches are usually in abundance & that they can accurately represent the manifold.
- Since these manifolds are not modelled by any one mathematical function but by several small linear patches, these linear neighborhoods can model any manifold.
- The fundamental assumptions or aspects of manifold learning algorithms:
  - ❖ There exist nonlinear relationships in the data that can be modelled through manifolds — surfaces that span multiple dimensions, are smooth, and not too complex. The manifolds are continuous.
  - ❖ It is not important to maintain the multi-dimensional shape of data. Instead of ‘flattening’ or ‘projecting’ it (as with PCA) with specific directions to maintain the general shape of the data, it is alright to perform more complex manipulations like unfolding a coiled strip or flipping a sphere inside out.
  - ❖ The best method to model manifolds is to treat the curved surface as being composed of several neighborhoods. If each data point manages to preserve the distance not with all the other points but only the ones close to it, the geometric relationships can be maintained in the data.

Content & Image Source: https://towardsdatascience.com/manifold-learning-t-sne-lle-isomap-made-easy-42cfd61f5183/

Manifold Learning – Continued.,

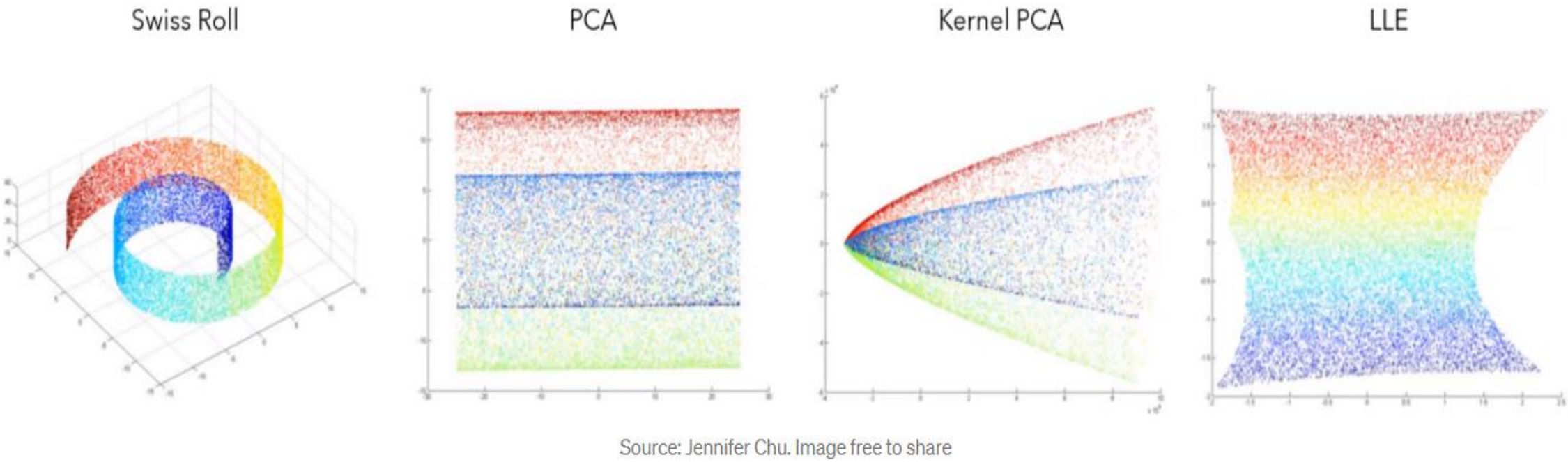




- On the left is a more PCA-like approach towards preserving the shape of the data, where each point is connected to each other.
- On the right, however, is an approach in which only the distance between neighborhoods of data points are valued.

Content & Image Source: https://towardsdatascience.com/manifold-learning-t-sne-lle-isomap-made-easy-42cfd61f5183/

Manifold Learning – Continued.,



Source: Jennifer Chu. Image free to share

- On the Swiss Roll dataset, PCA and even specialized variations like Kernel PCA fail to capture the gradient of values. Locally Linear Embeddings (LLE), a manifold learning algorithm, on the other hand, is able to.

Content & Image Source: https://towardsdatascience.com/manifold-learning-t-sne-lle-isomap-made-easy-42cfd61f5183/

Manifold Learning – Continued.,

- **Three popular manifold learning algorithms:**
  - IsoMap (Isometric Mapping)
  - Locally Linear Embeddings &
  - t-SNE.

IsoMap :

Isomap seeks a lower-dimensional representation that maintains 'geodesic distances' between the points. A geodesic distance is a generalization of distance for curved surfaces. Hence, instead of measuring distance in pure Euclidean distance with the Pythagorean theorem-derived distance formula, Isomap optimizes distances along a discovered manifold.

Content & Image Source: https://towardsdatascience.com/manifold-learning-t-sne-lle-isomap-made-easy-42cfd61f5183/

Manifold Learning – Continued.,

Locally Linear Embeddings (LLE) :

Locally Linear Embeddings use a variety of tangent linear patches (as demonstrated with the diagram above) to model a manifold. It can be thought of as performing a PCA on each of these neighborhoods locally, producing a linear hyperplane, then comparing the results globally to find the best nonlinear embedding.

The goal of LLE is to 'unroll' or 'unpack' in distorted fashion the structure of the data, so often LLE will tend to have a high density in the center with extending rays..

t-SNE :

t-SNE is one of the most popular choices for high-dimensional visualization, and stands for **t-distributed Stochastic Neighbor Embeddings**. The algorithm converts relationships in original space into t-distributions, or normal distributions with small sample sizes and relatively unknown standard deviations. This makes t-SNE very sensitive to the local structure, a common theme in manifold learning. It is considered to be the go-to visualization method because of many advantages it possesses

Content & Image Source: https://towardsdatascience.com/manifold-learning-t-sne-lle-isomap-made-easy-42cfd61f5183/

Manifold Learning – Continued.,

Advantages of t-SNE :

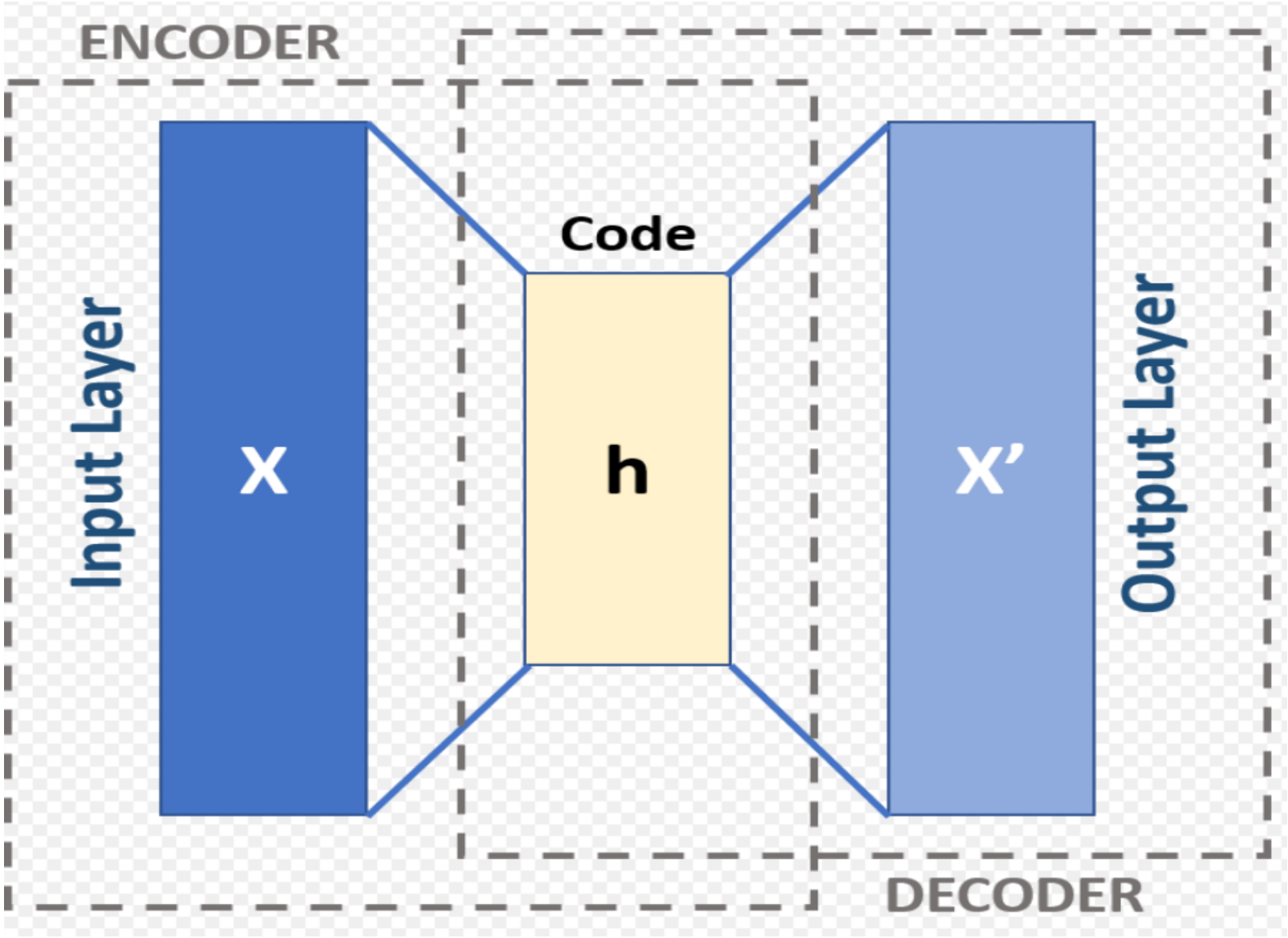
- It is able to reveal the structure of the data at many scales.
- It reveals data that lies in multiple manifolds and clusters
- Has a smaller tendency to cluster points at the center.

Points to Remember:

- PCA fails to model nonlinear relationships because it is linear.
- Nonlinear relationships appear in datasets often because external forces like lighting or the tilt can move a data point of the same class nonlinearly through Euclidean space.
- Manifold learning attempts to generalize PCA to perform dimensionality reduction on all sorts of dataset structures, with the main idea that manifolds, or curved, continuous surfaces, should be modelled by preserving and prioritizing local over global distance.
- Isomap tries to preserve geodesic distance, or distance measured not in Euclidean space but on the curved surface of the manifold.
- LLE can be thought of as representing the manifold as several linear patches, in which PCA is performed on.
- t-SNE takes more of an 'extract' approach opposed to an 'unrolling' approach, but still, like other manifold learning algorithms, prioritizes the preservation of local distances by using probability and t-distributions.

Content & Image Source: https://towardsdatascience.com/manifold-learning-t-sne-lle-isomap-made-easy-42cfd61f5183/

AutoEncoders



- An **autoencoder** is a type of artificial neural network used to learn efficient coding's of unlabeled data.
- It is based on unsupervised learning
- The encoding is validated and refined by attempting to regenerate the input from the encoding.
- The autoencoder learns a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore insignificant data ("noise").

Content & Image Source: Https://www.Autoencoder schema - Autoencoder - Wikipedia

AutoEncoders

- An Autoencoder attempts to encode the data by compressing it into the lower dimensions (bottleneck layer or code)
- Then it decodes the data to reconstruct the original input.
- The bottleneck layer (or code) holds the compressed representation of the input data
- the number of output units must be equal to the number of input units
- The encoder encodes the provided data into a lower dimension which is the size of the bottleneck layer and the decoder decodes the compressed data into its original form
- The number of neurons in the layers of the encoder will be decreasing as we move on with further layers, whereas the number of neurons in the layers of the decoder will be increasing as we move on with further layers.
- The encoder contains 32, 16, and 7 units in each layer respectively and the decoder contains 7, 16, and 32 units in each layer respectively
- The code size/ the number of neurons in bottle-neck must be less than the number of features in the data
- Before entering the data into the Autoencoder, the data must be scaled between 0 and 1 using MinMaxScaler

Content & Image Source: Https://www.Autoencoder schema - Autoencoder - Wikipedia

AutoEncoders

When we use Autoencoders for dimensionality reduction we'll be extracting the bottleneck layer and thus reducing the dimensions. This process can be regarded as feature extraction

Types of Autoencoders

- Deep Autoencoder
- Sparse Autoencoder
- Under complete Autoencoder
- Variational Autoencoder
- LSTM Autoencoder

Hyperparameters of an Autoencoder

- Code size or the number of units in the bottleneck layer
- Input and output size, which is the number of features in the data
- Number of neurons or nodes per layer
- Number of layers in encoder and decoder
- Activation function
- Optimization function

Content & Image Source: https:// File:Autoencoder structure.png - Wikimedia Commons

AutoEncoders

Training:

Geoffrey Hinton developed the deep belief network technique for training many-layered deep autoencoders.

His method involves treating each neighboring set of two layers as a restricted Boltzmann machine, then using backpropagation to fine-tune the results

Joint training (i.e. training the whole architecture together with a single global reconstruction objective to optimize) would be better for deep auto-encoders]

However, the success of joint training depends heavily on the regularization strategies adopted.

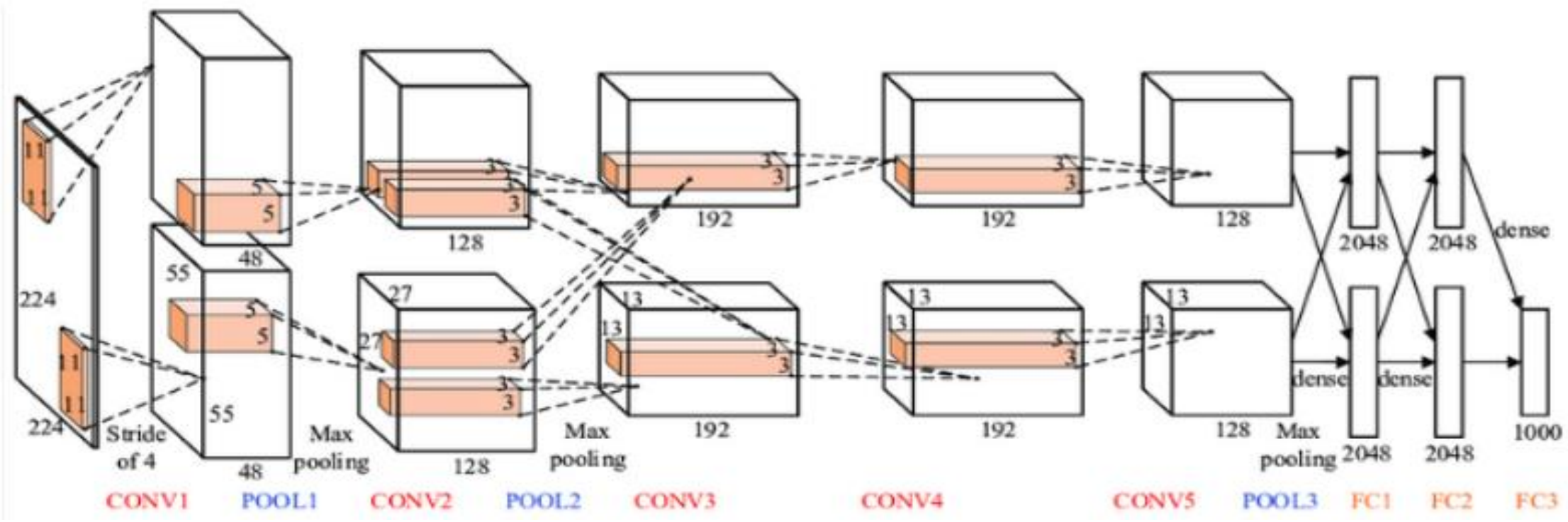
Applications of Autoencoders :

- ✓ Dimensionality reduction
- ✓ Anomaly detection
- ✓ Image denoising
- ✓ Image compression
- ✓ Image generation

Content & Image Source: https:// File:Autoencoder structure.png - Wikimedia Commons

Alex Net

Alexnet model was proposed in 2012 in the research paper named Imagenet Classification with Deep Convolution Neural Network by Alex Krizhevsky and his colleagues.



AlexNet Architecture

Content & Image Source: https:// File:Autoencoder structure.png - Wikimedia Commons

Alex Net

- The Alexnet has eight layers with learnable parameters
- The model has five layers with a combination of max pooling followed by 3 fully connected layers
- The fully connected layers use Relu activation except the output layer
- They found out that using the Relu as an activation function accelerated the speed of the training process by almost six times.
- They also used the dropout layers, which prevented the model from overfitting.
- The model is trained on the Imagenet dataset. The Imagenet dataset has arounf 14 million images across a 1000 classes.
- The input to this model is the images of size 227X227X3
- The first convolution layer with 96 filters of size 11X11 with stride 4
- The activation function used in this layer is relu. The output feature map is 55X55X96
- Next, we have the first Maxpooling layer, of size 3X3 and stride 2
- Next the filter size is reduced to 5X5 and 256 such filtersare added
- The stride value is 1 and padding 2. The activation function used is again relu. The output size we get is 27X27X256

Content & Image Source: https:// File:Autoencoder structure.png - Wikimedia Commons

Alex Net

- Next we have a max-pooling layer of size 3X3 with stride 2. The resulting feature map size is 13X13X256
- The third convolution operation with 384 filters of size 3X3 stride 1 and also padding 1is done next. In this stage the activation function used is relu. The output feature map is of shape 13X13X384
- Then the fourth convolution operation with 384 filters of size 3X3. The stride value along with the padding is 1.The output size remains unchanged as 13X13X384.
- After this, we have the final convolution layer of size 3X3 with 256 such filters. The stride and padding are set to 1,also the activation function is relu. The resulting feature map is of shape 13X13X256

If we look at the architecture now, the number of filters is increasing as we are going deeper. Hence more features are extracted as we move deeper into the architecture. Also, the filter size is reducing, which means a decrease in the feature map shape.

Content & Image Source: https:// File:Autoencoder structure.png - Wikimedia Commons

Alex Net

- Points to Remember:

  - It has 8 layers with learnable parameters.
  - The input to the Model is RGB images.
  - It has 5 convolution layers with a combination of max-pooling layers.
  - Then it has 3 fully connected layers.
  - The activation function used in all layers is Relu
  - It has two Dropout layers.
  - The activation function used in the output layer is Softmax.
  - The total number of parameters in this architecture is 62.3 million.

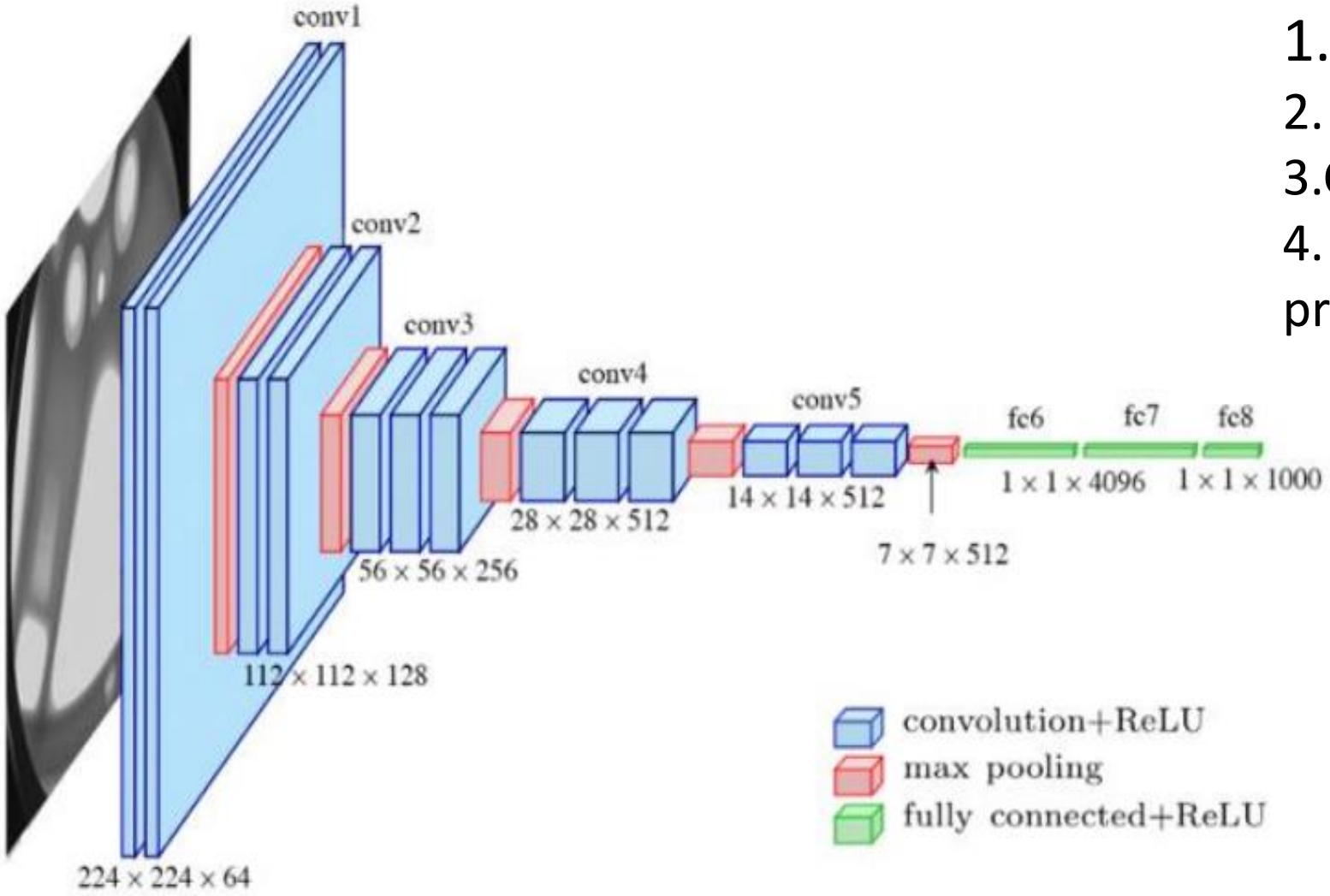
Content & Image Source: https:// File:Autoencoder structure.png - Wikimedia Commons

VGG-16 Net

- The major shortcoming of too many hyper-parameters of AlexNet was solved by VGG Net by replacing large kernel-sized filters (11 and 5 in the first and second convolution layer, respectively) with multiple 3×3 kernel-sized filters one after another.
- The architecture developed by Simonyan and Zisserman was the 1st runner up of the Visual Recognition Challenge of 2014.
- The architecture consist of 3*3 Convolutional filters, 2*2 Max Pooling layer with a stride of 1.
- Padding is kept same to preserve the dimension.
- There are 16 layers in the network where the input image is RGB format with dimension of 224*224*3, followed by 5 pairs of Convolution(filters: 64, 128, 256,512,512) and Max Pooling.
- The output of these layers is fed into three fully connected layers and a softmax function in the output layer.
- In total there are 138 Million parameters in VGG Net

Content & Image Source: https:// File:Autoencoder structure.png - Wikimedia Commons

VGG-16 Net



Drawbacks of VGG Net:

1. Long training time
2. Heavy model
3. Computationally expensive
4. Vanishing/exploding gradient problem

Content & Image Source: https:// File:Autoencoder structure.png - Wikimedia Commons

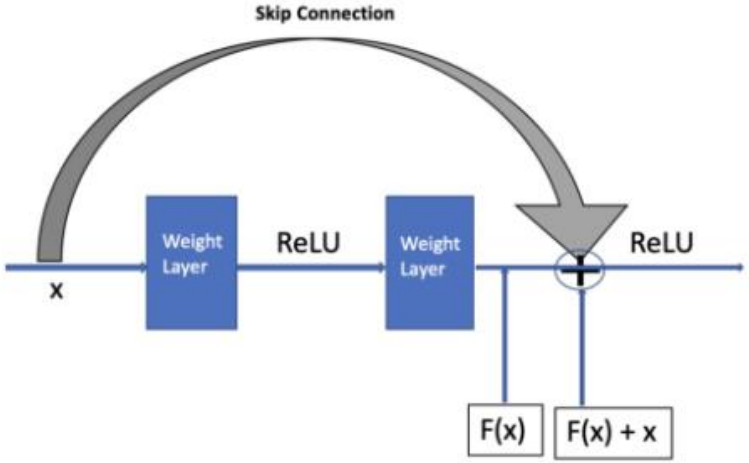
ResNet

ResNet, the winner of ILSVRC-2015 competition is a deep network with over 100 layers. Residual networks (ResNet) is similar to VGG nets however with a sequential approach they also use “Skip connections” and “batch normalization” that helps to train deep layers without hampering the performance.

After VGG Nets, as CNNs were going deep, it was becoming hard to train them because of vanishing gradients problem that makes the derivate infinitely small.

Therefore, the overall performance saturates or even degrades.

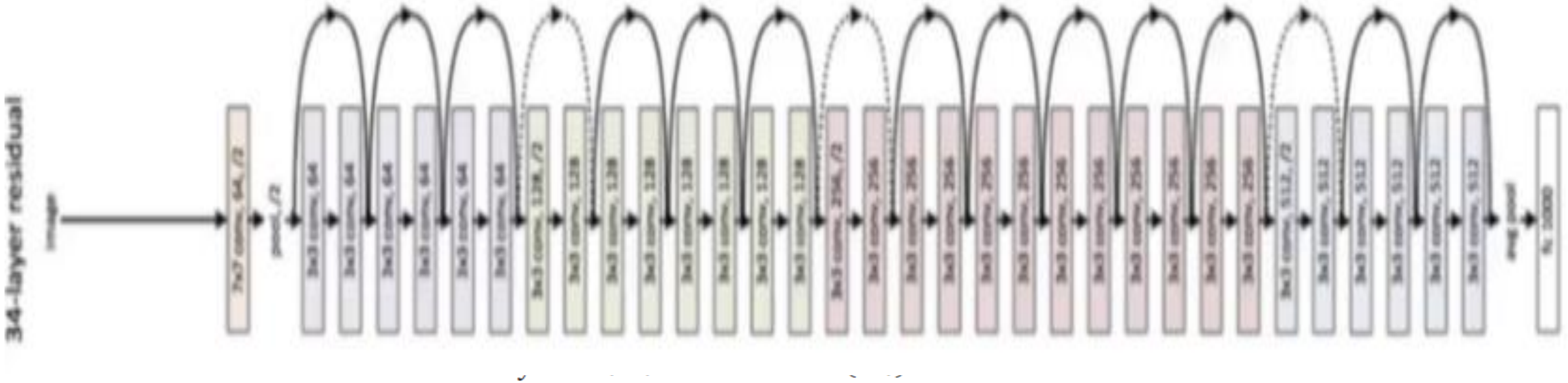
The idea of skips connection came from highway network where gated shortcut connections were used



A Residual Block

Content Source: https:// File:Autoencoder structure.png - Wikimedia Commons
Image Source: Understanding Residual Network (ResNet)Architecture | Analytics Vidhya

ResNet

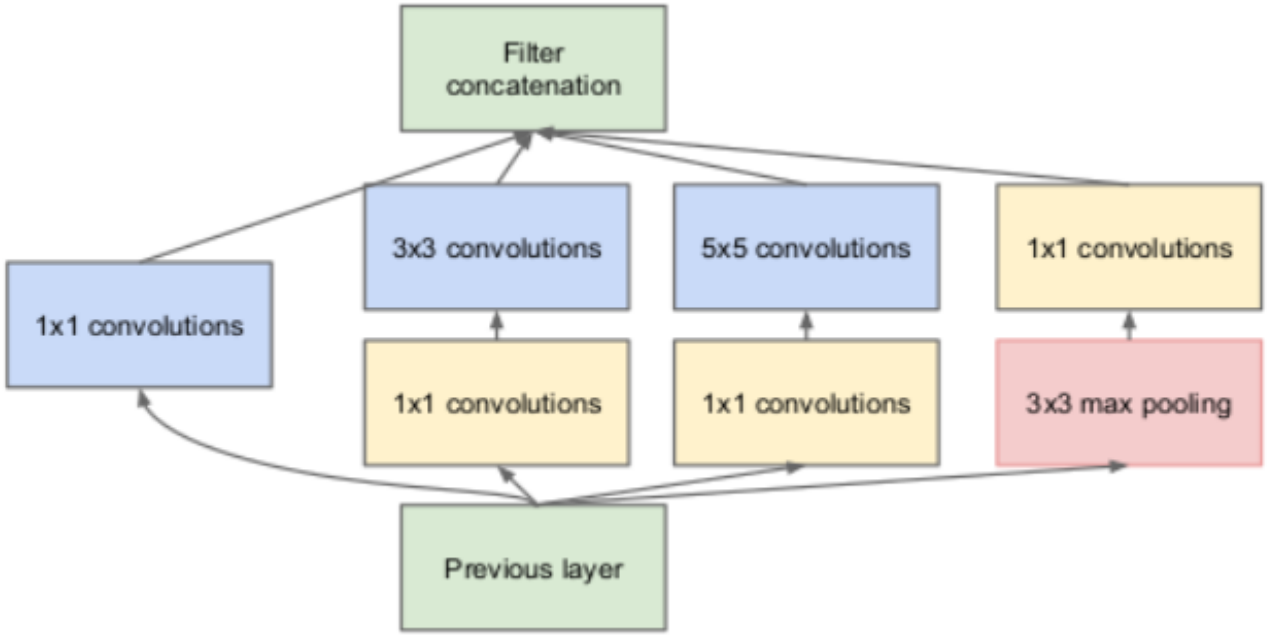


2 followed by a MaxPooling operation.

- It consists of four residual blocks (config:- 3,4,6 and 3 respectively)
- Channels for each block are constant— 64, 128, 256, 512 respectively.
- Only 3x3 kernels have been used in these blocks.
- Except for the first block, each block starts with a 3x3 kernel of stride of 2.
- The dotted lines are **skip connections**.

Image & Content Source: Understanding Residual Network (ResNet)Architecture | Analytics Vidhya

INCEPTION Net



convolution
max pool
convolution
max pool
inception (3a)
inception (3b)
max pool
inception (4a)
inception (4b)
inception (4c)
inception (4d)
inception (4e)
max pool
inception (5a)
inception (5b)
avg pool
dropout (40%)
linear
softmax

- Inception network also known as GoogleLe Net was proposed by developers at google in “Going Deeper with Convolutions” in 2014.
- The motivation of InceptionNet comes from the presence of sparse features Salient parts in the image that can have a large variation in size.
- Due to this, the selection of right kernel size becomes extremely difficult as big kernels are selected for global features and small kernels when the features are locally located.

Image & Content Source: https://www.analyticsvidhya.com/blog/2018/10/understanding-inception-network-from-scratch

INCEPTION Net

- The InceptionNets resolves this by stacking multiple kernels at the same level.
- Typically it uses 5*5, 3*3 and 1*1 filters in one go.
- For better understanding refer to the image below:.

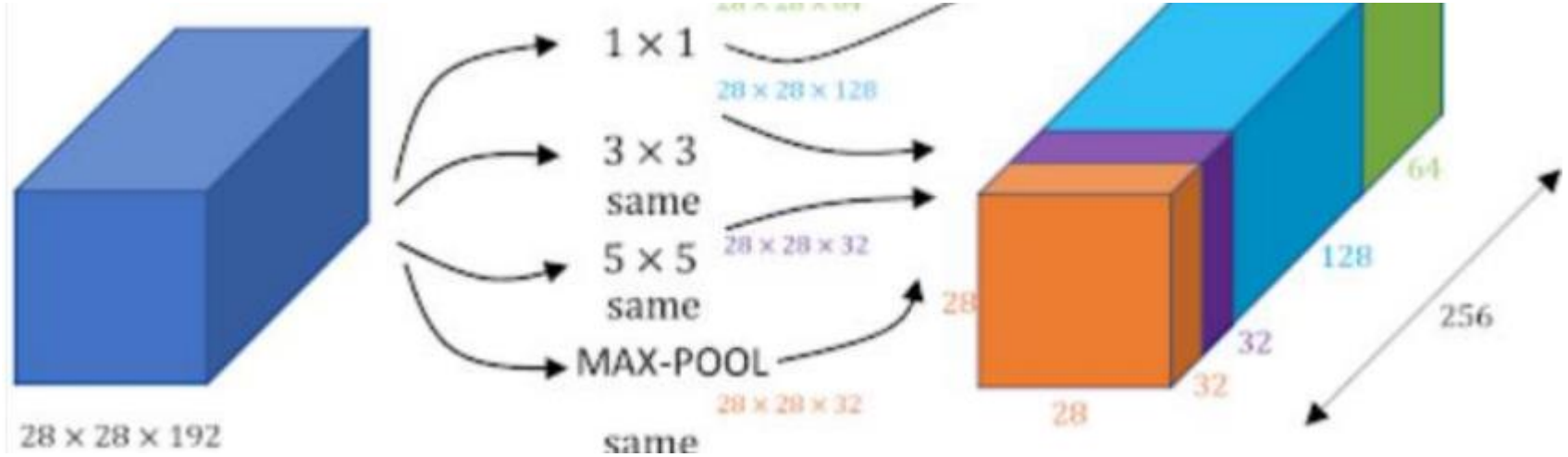


Image & Content Source: https://www.analyticsvidhya.com/blog/2018/10/understanding-inception-network-from-scratch

INCEPTION Net

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Image & Content Source: https://www.analyticsvidhya.com/blog/2018/10/understanding-inception-network-from-scratch

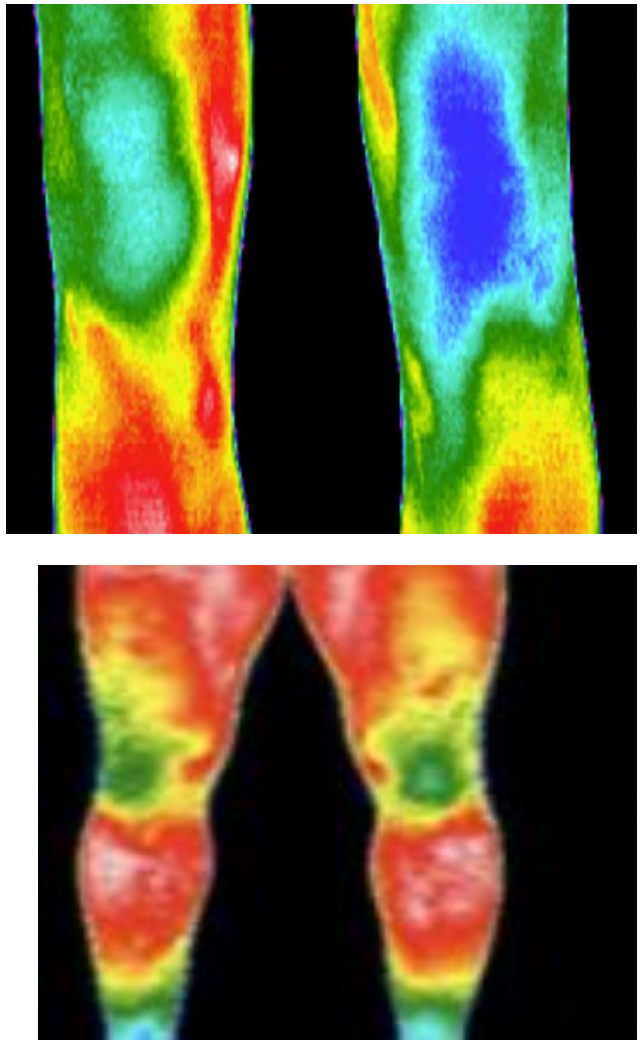
Hyperparameter Optimization

- Hyperparameter optimization in machine learning intends to find the hyperparameters of a given machine learning algorithm that deliver the best performance as measured on a validation set.
- Hyperparameters, in contrast to model parameters, are set by the machine learning engineer before training.
- The number of trees in a random forest is a hyperparameter while the weights in a neural network are model parameters learned during training
- Hyperparameter optimization finds a combination of hyperparameters that returns an optimal model which reduces a predefined loss function and in turn increases the accuracy on given independent data
- **Hyperparameter Optimization methods**

  - Manual Hyperparameter Tuning
  - Grid Search
  - Random Search
  - Bayesian Optimization
  - Gradient-based Optimization

Image & Content Source: https://www.kdnuggets.com/2020/05/hyperparameter-optimization-machine-learning-models.html

Application of Deep Learning Neural Network- Joint Detection



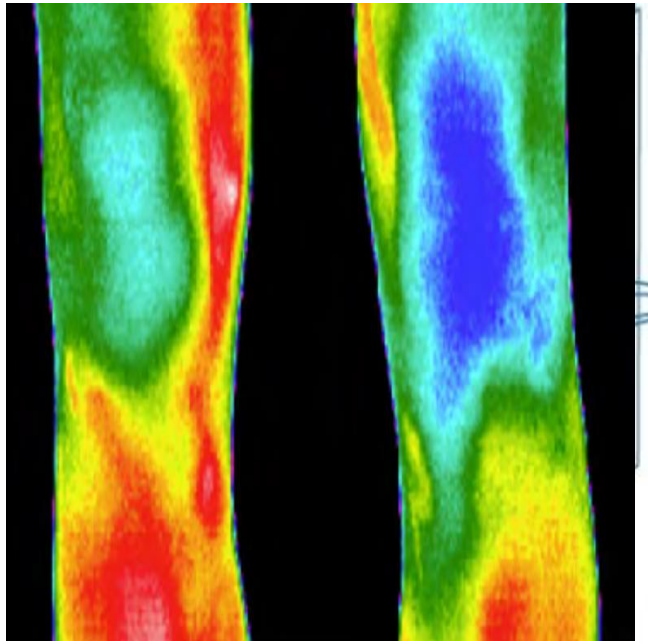
DLNN MODEL

Rheumatoid Arthritis Present

Rheumatoid Arthritis Absent

Image & Content Source: https://www.researchgate.net/figure/Infrared-image-from-the-anterior-view-of-the-knees-TRIATHLON_fig2_221924974

Application of Deep Learning Neural Network- Joint Detection



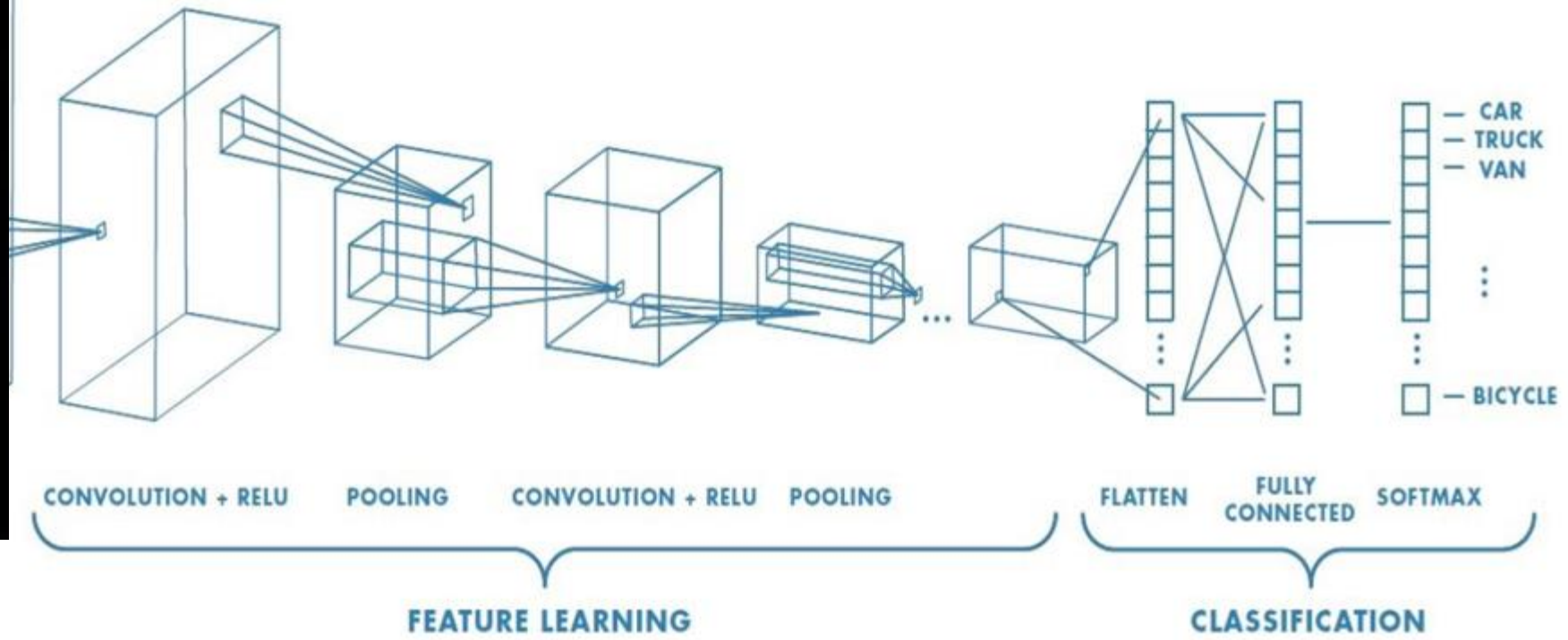


Image & Content Source: https://www.researchgate.net/figure/Infrared-image-from-the-anterior-view-of-the-knees-TRIATHLON_fig2_221924974

Application of Deep Learning Neural Network- Joint Detection

For Detection of Pedestrian

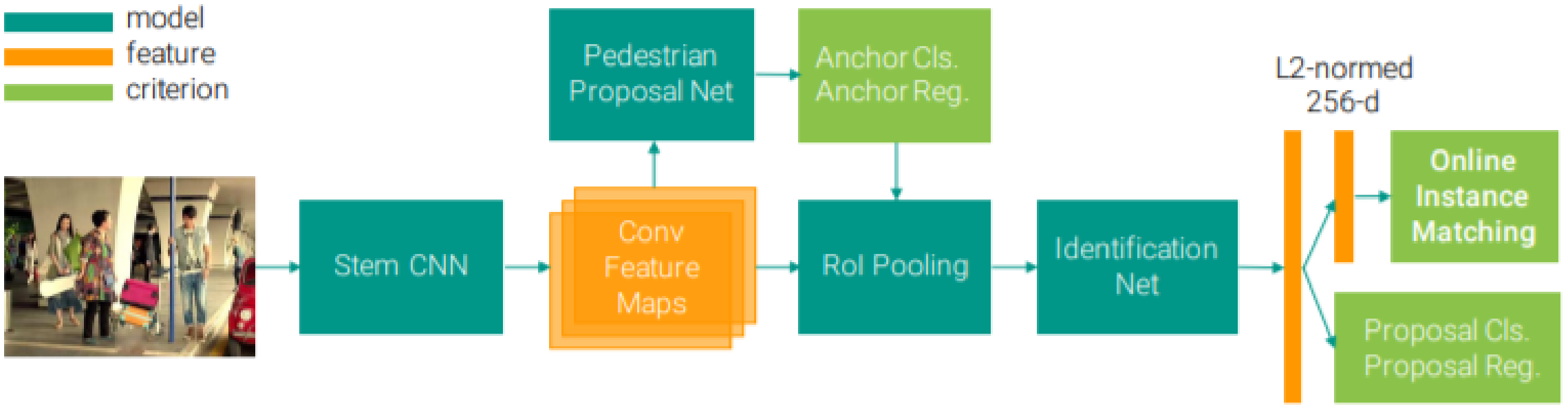


Image & Content Source: https://openaccess.thecvf.com/content_cvpr_2017/papers/Xiao_Joint_Detection_and_CVPR_2017_paper.pdf

Application of Deep Learning Neural Network- Face Recognition

1. **Detect**/identify faces in an image (using a face detection model) — for simplicity, this tutorial will only use images with one face/person in it, not more/less
2. Predict face poses/**landmarks** (for the faces identified in step 1)
3. Using data from step 2 and the actual image, calculate face **encodings** (numbers that describe the face)
4. **Compare** the face encodings of known faces with those from test images to tell who is in the picture

Image & Content Source: https://towardsdatascience.com/facial-recognition-using-deep-learning-a74e9059a150#

Application of Deep Learning Neural Network- Face Recognition

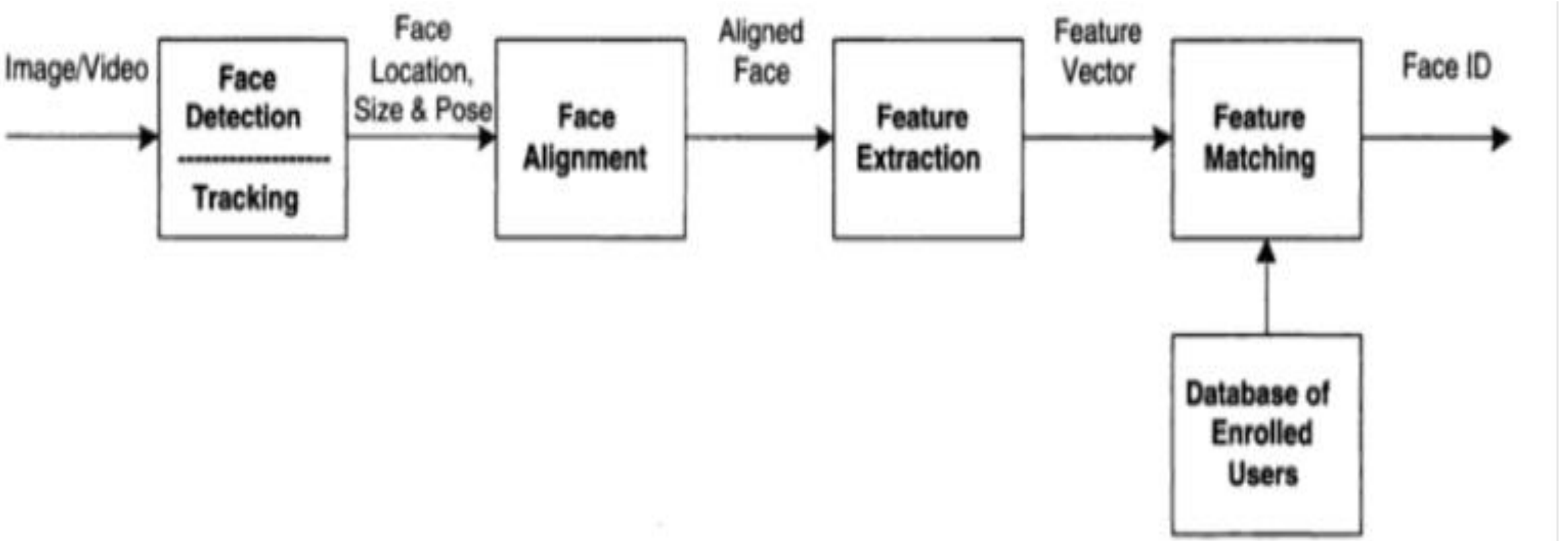


Image & Content Source: https://machinelearningmastery.com/introduction-to-deep-learning-for-face-recognition/

Dr. V. Vedanarayanan B.E., M.E., PhD
Assistant Professor, Department of ECE,
School of Electrical and Electronics
Sathyabama Institute of Science and technology
Vedanarayanan.etc@sathyabama.ac.in