Facial Keypoint Detection

Image Processing using Machine Learning

Data Preprocessing

- Data preprocessing is the act of processing your data before you actually implement it or make inferences / analysis off of it. You check your data, make sure it’s valid and that the it can be read accurately and that any lack of data which is missing can be adjusted for correctly without throwing off your outputted data.

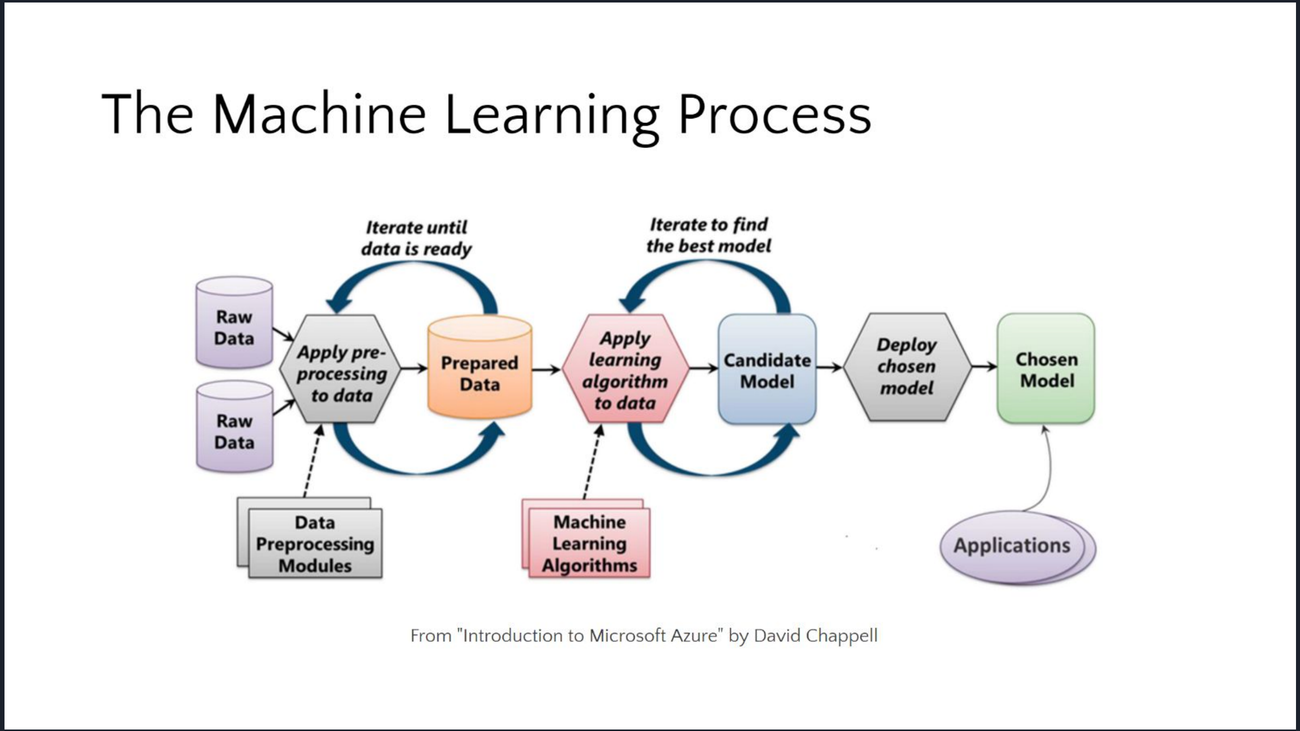
- The idea of data pre processing has been around for more than 50 years. Back in 1964 when there was data of a given head, it obviously had to read the data. But what if the head was tilted? The computers back then couldn’t manage those computations so we had to analyze the head and it’s tilt, angle, etc. Then manually do the calculations to get the data of the head untitled that way we have accurate data when computed.

- The inputs are essentially just the raw data, we use it to find where there are missing values or null values which would throw our data off.

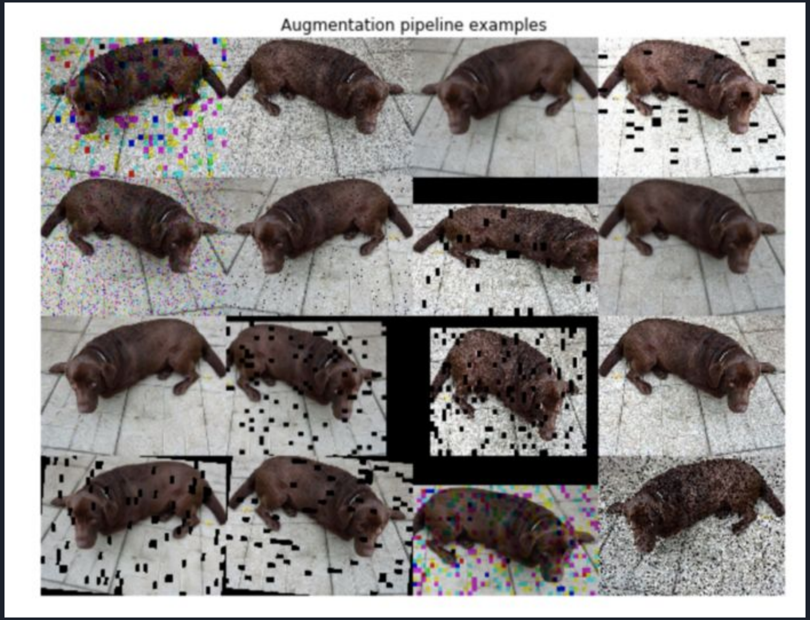
- The outputs would just be a dataframe containing the rows with null / missing values. This dataset would of course have to be changed to be used as a training for our machine.

- Parameters would solely include the data set, we can’t do or read anything beyond the data set. We can only analyze what we have inputted and what is outputted.

Data Preprocessing



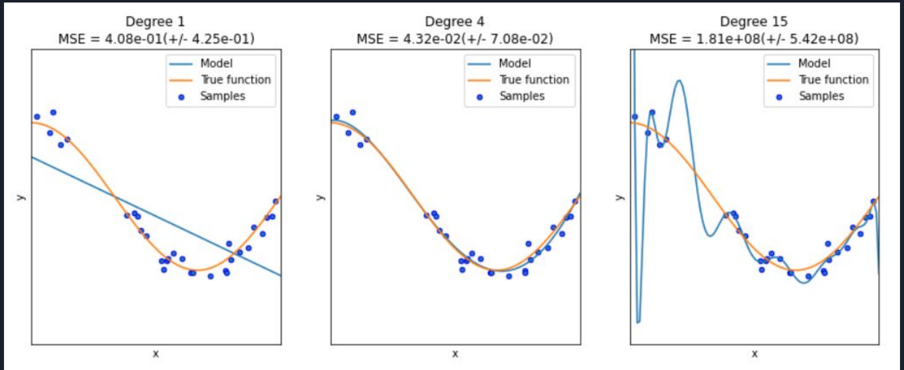
Data Augmentation

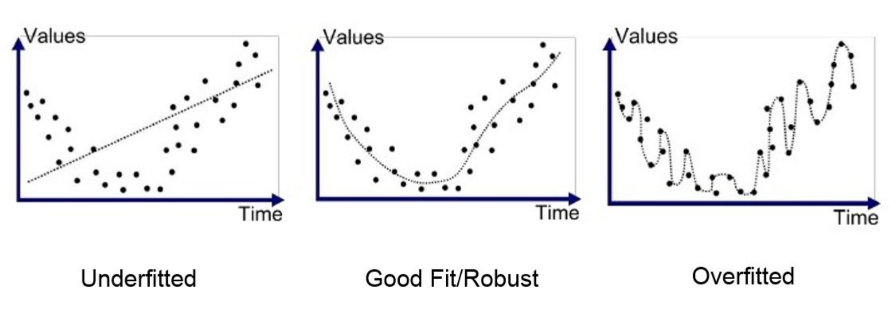


Data augmentation is a common technique used in ML/DL Models which expands the size of a training set by augmenting the existing set.

The most common techniques used in Image Data Augmentation include:

- **Geometric transformations** - flipping, cropping, rotations, translations, etc.
- **Color space transformations** - intensification of colors
- **Kernel filters** - sharpening or blurring an image
- **Random Erasing** - erasing part of the image





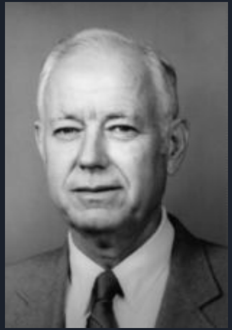
Data Augmentation is used to achieve greater variety and data size.

It is practiced heavily in cases where data size is either too small or lacking in variation.

This allows the model to generalize trends and patterns throughout the data and reduce under/overfitting.

Prediction Methods

- Believe it or not, methods related to computer automated facial recognition can be traced back to the 1960s.
- However, these methods relied entirely on humans using graphics tablets to identify “landmarks” on the face such as eye centres, mouth etc.
- These were then mathematically rotated by a computer to compensate for pose variation. The distances between landmarks were then automatically computed and compared between images to determine a person’s identity based on the distances between these features.
- However, at this time it was easier for computers to defeat a Grandmaster at chess than it was for them to recognize a face.



Woody Bledsoe

American mathematician, computer scientist, and prominent educator.

Helen Chan Wolf

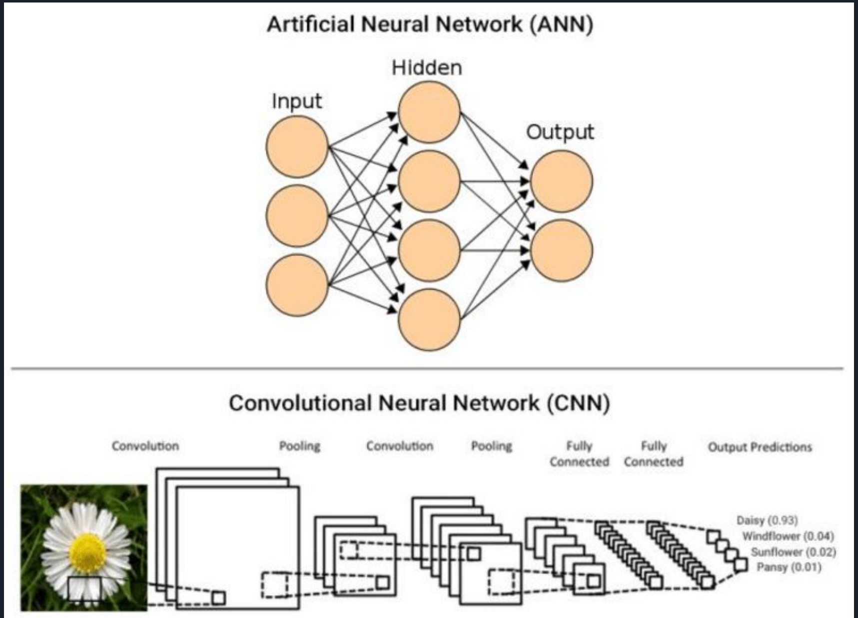
AI pioneer who worked on facial recognition technology at the SRI International. Help Shakey the robot, the world's first autonomous robot.

Prediction Methods

- Methods used in the past have varied widely based on the desired results (i.e. facial tracking, identification, facial keypoint detection) as well as the form of input (i.e. live video, 2D photos, 3D models).
- However, most modern approaches for facial recognition make use of camera technology improvements, mapping processes, machine learning, and neural networks.
- Neural networks in particular can be traced back to the 1950’s where researchers attempted to create computer simulated models that could imitate the complexity of human brain functions.
- In the past 10 years, the best-performing artificial-intelligence systems have resulted from a technique called “deep learning” which is an approach to AI that utilizes these neural networks to make predictions.
- However, Neural Networks as a general trend have fallen in and out of popularity over the years due to many inadequacies in computing power as well as theoretical models.

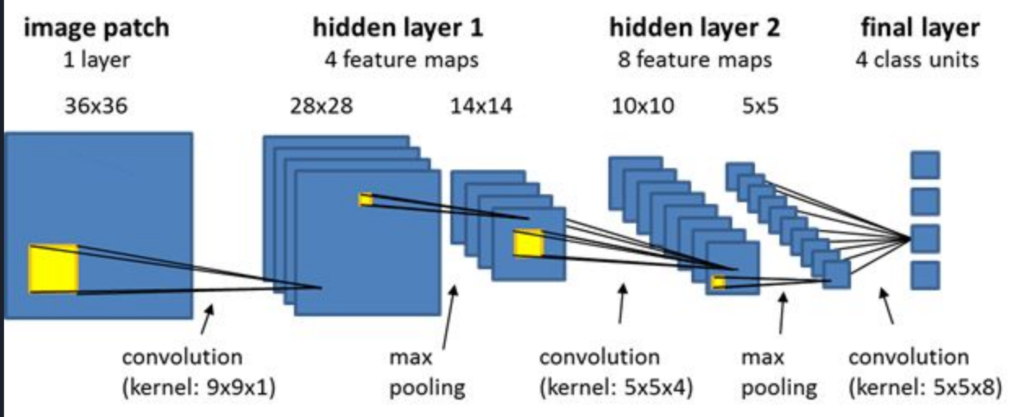
Prediction Methods

- For our approach we have chosen to make use of Convolutional Neural Networks since they can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.
- A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters.
- The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights.
- Thus, the goal of a CNN compared to a traditional forward-feed neural network is to reduce the image to a form that is easier to process without losing features that are critical for getting a good prediction.



Prediction Methods

- Convolutional Neural Networks (CNN) consist of an input layer, hidden layers, and an output layer.
- The hidden layers include layers that perform convolutions. Typically this includes a layer that does multiplication or other dot products.
- This is followed by other convolution layers such as pooling layers, fully connected layers and normalization layers.
- The neurons in one layer of a CNN do not connect to all the neurons in the next layer, only a small region of them.
- The data frame containing all of our training data as well as the known predictions will be used as the input to our CNN for training. For predictions our input will be the data frame containing our image data matrix.
- The final output will be reduced to a single vector of xy-coordinates that correspond to the given input image, organized along the depth dimension of the CNN.



Plotting Methods

Matplotlib:
A plotting library for the Python programming language and python extensions.

Matplotlib was originally written by John D. Hunter. However due to his death it was Michael Droettboom who was nominated as matplotlib's lead developer.

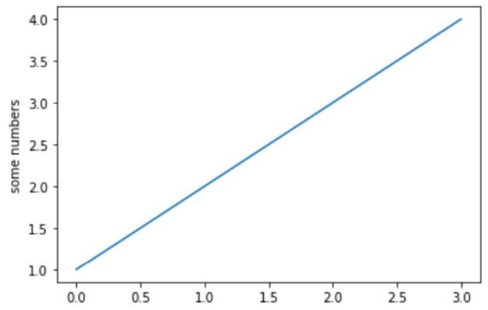
It was originally conceived to visualize electrocorticography (ECoG) data of epilepsy patients during post-doctoral research in neurobiology.

Input:
Coordinates that contain an x and y value.

Output:
Points on a chart or image.

Parameters:
Numerical values.

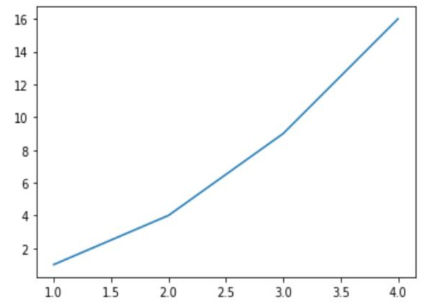
```python
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.ylabel('some numbers')
plt.show()
```



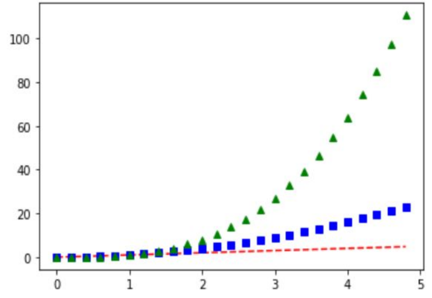

```python
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
```

Out[4]:

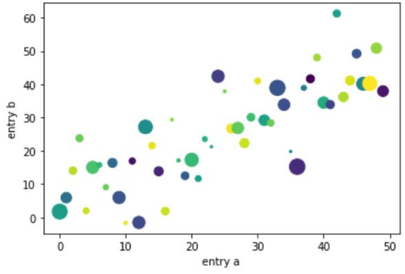
```
[<matplotlib.lines.Line2D at 0x7fc5d142efd0>]
```



```python
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



```python
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

plt.scatter('a', 'b', c='c', s='d', data=data)
plt.xlabel('entry a')
plt.ylabel('entry b')
plt.show()
```



Thank You For Your Time!