Introduction

We will be using Haar, dlib, Multi-task Cascaded Convolutional Neural Network (MTCNN), and OpenCV's DNN module. If you already know about them or don't want to go in their technical details, feel free to skip this section and move straight on to the code. Otherwise, let's learn how they work.

Haar Cascades

They were proposed way back in 2001 by Paul Viola and Micheal Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features." It is super fast to work with and like the simple CNN, it extracts a lot of features from images. The best features are then selected via Adaboost. This reduces the original 160000+ features to 6000 features. But applying all these features in a sliding window will still take a lot of time. So they introduced a Cascade of Classifiers, where the features are grouped. If a window fails at the first stage, these remaining features in that cascade are not processed. If it passes then the next feature is tested and the same procedure is repeated. If a window can pass all the features then it is classified as a face region.

Haar cascades require a lot of positive and negative training images to train. Thankfully, these cascades come bundled with the OpenCV library along with the trained XML files.

Dlib Frontal Face Detector

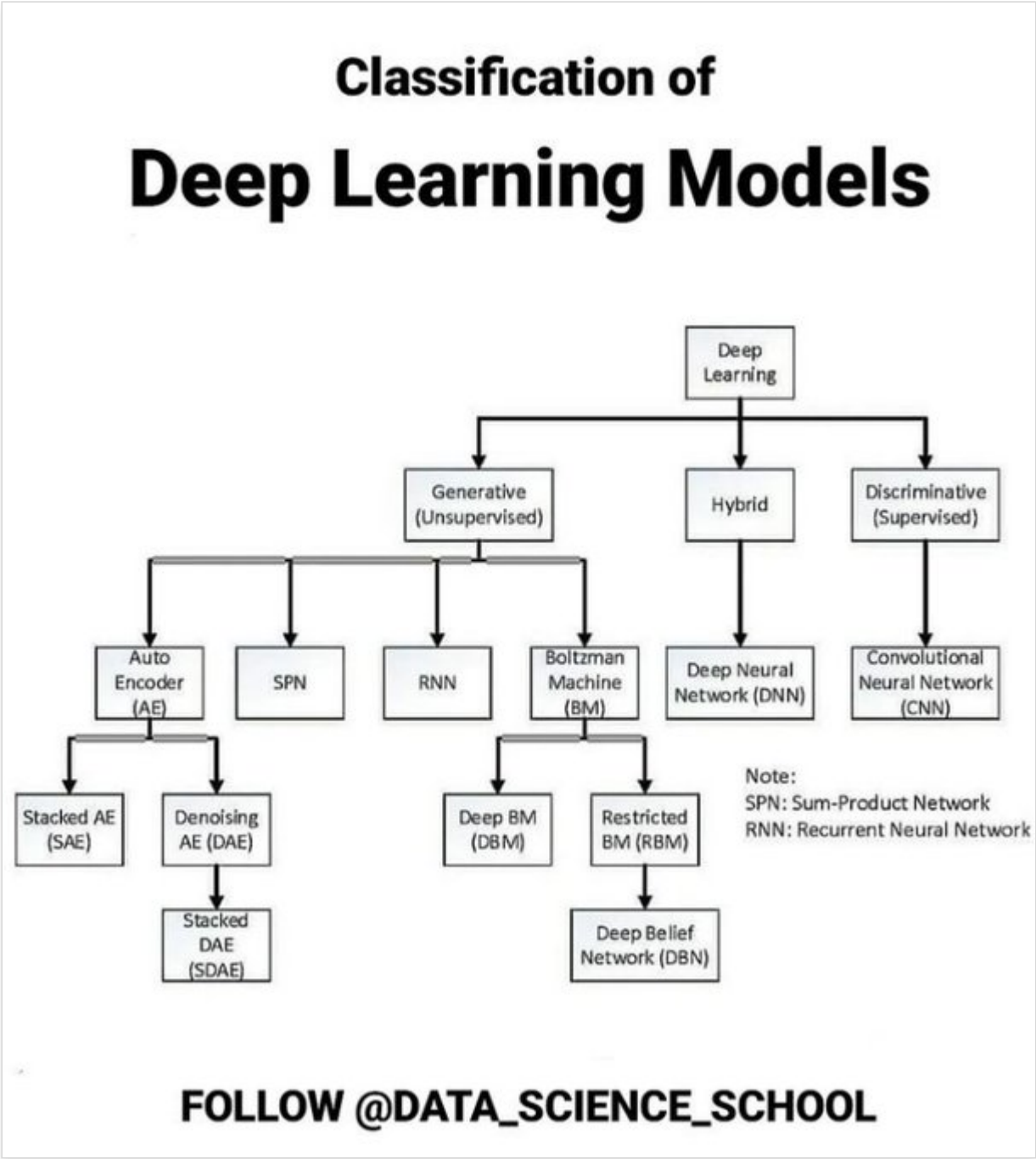
Dlib is a C++ toolkit containing machine learning algorithms used to solve real-world problems. Although it is written in C++ it has python bindings to run it in python. It also has the great facial landmark keypoint detector which I used in one of my earlier articles to make a real-time gaze tracking system.
The frontal face detector provided by dlib works using features extracted by Histogram of Oriented Gradients (HOG) which are then passed through an SVM. In the HOG feature descriptor, the distribution of the directions of gradients is used as features. Moreover, Dlib provides a more advanced CNN based face detector, however, that does not work in real-time on a CPU which is one of the goals we are looking for so it has been ignored in this article.

Convolutional Neural Network (CNN)

is an neural network which extracts or identifies a feature in a particular image. This forms one of the most fundamental operations in Machine Learning and is widely used as a base model in majority of Neural Networks like GoogleNet, VGG19 and others for various tasks such as Object Detection, Image Classification and others.

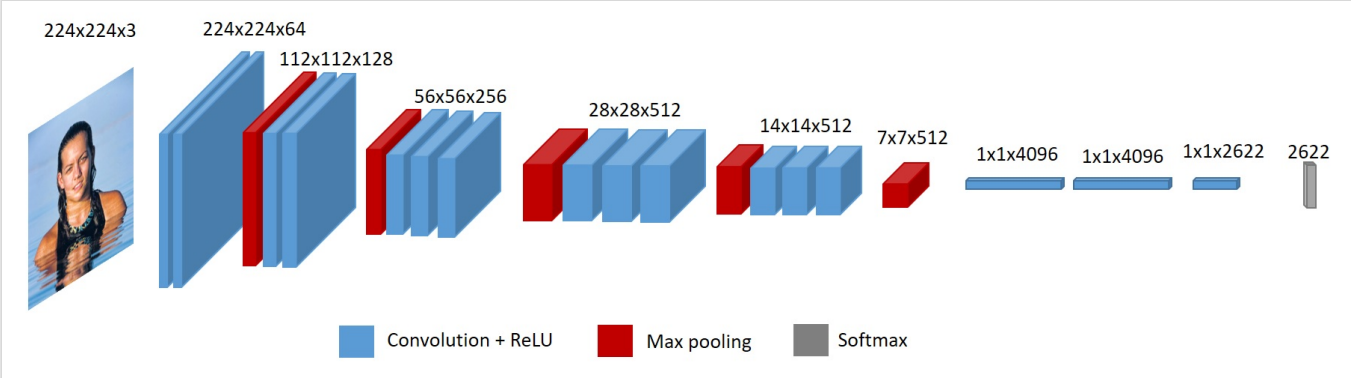
CNN has the following five basic components:

- **Convolution** : to detect features in an image
- **ReLU** : to make the image smooth and make boundaries distinct
- **Pooling** : to help fix distored images
- **Flattening** : to turn the image into a suitable representation
- **Full connection** : to process the data in a neural network



A CNN works in pretty much the same way an ANN works but since we are dealing with images, a CNN has more layers to it than an ANN. In an ANN, the input is a vector, however in a CNN, the input is a multi-channelled image.

Deep face Recognition with VGG-Face in Keras



Conclusion

You can modify this template to create a classification model for any group of images. Just put the images of each category in its respective folder and train the model.

The CNN algorithm has helped us create many great applications around us! Facebook is the perfect example! It has trained its DeepFace CNN model on millions of images and has an accuracy of 97% to recognize anyone on Facebook. This may

CNN is being used in the medical industry as well to help doctors get an early prediction about benign or malignant cancer using the tumor images. Similarly, get an idea about typhoid by looking at the X-ray images, etc.