ABOUT MACHINE LEARNING

Machine Learning is a type of artificial intelligence that gives computers the capability to predict outcomes without explicitly being programmed. Artificial intelligent systems are utilized to perform complex errands in a way that is comparative to how people solve problems. Machine learning begins with data – numbers, photographs, or content, like bank exchanges, pictures of individuals or indeed pastry kitchen things, records, time data from sensors, or sales/business report. The data is assembled and arranged to be utilized as a training data, or the information the machine learning model will be prepared on. The more the data, the way better the program and the more will be the accuracy.

There are three subcategories of machine learning;

Supervised Learning Machine learning models are prepared with labeled data sets, which permit the models to memorize and develop more precise over time. All the features and labels of data are related to each other.

Unsupervised Learning In this Machine learning model a program seeks for patterns/designs in unlabeled data. Unsupervised machine learning can discover designs or patterns that individuals aren't unequivocally searching for.

Reinforcement Learning Machine learning trains model through trial and mistake to require the leading activity by building a remunerate framework. It is basically a feedback system where the output send back it's feedback to the model so that the model can predict the right output.

Machine learning is also associated with a few other artificial intelligence subfields:-

Natural language processing

NLP is a field of machine learning in which machines learn to get it common language as talked and composed by people, rather than the information and numbers ordinarily utilized to program computers. These permits machines to recognize language, understand it and react to it , as well as makes new text and interpret between languages. Familiar examples are chatbots and digital assistant Alexa.

Neural network

Artificial neural networks are modeled on the human brain, in which thousands or millions of handling are interconnected and organized into layers. In an ANN, cells are related with each cell processing inputs and yield result that's sent to other neurons.

Deep learning

Deep learning networks are simply neural networks with multiple layers. The layered network can handle broad sums of information and decide the 'weight' of each link within the network.

MACHINE LEARNING BASED AGE AND GENDER PREDICTIONS

Age detection is the method of consequently perceiving the age of an individual exclusively from a photo of their face.

The nonstop progression of AI models for classification and facial acknowledgment has picked up a portion of thought and noteworthiness these days and have colossally constituted in finding courses of action for complex genuine life issues. The Gender and Age Prediction is a Deep Learning application. For age and gender forecasts, researchers have come up with different algorithms utilizing classification and ML concepts. Most primitive type algorithms are utilized to infer numerous secondary algorithms with changes. “Fisherfaces” and “Eigenfaces” algorithms are considered to be as primary ones. Also Deep Convolutional Neural Networks (CNN) is another method that can be utilized. Both Fisherfaces algorithm and deep CNN are frequently used for age and gender predictions. When it comes to CNN as of now trained sets are accessible with their results such as age_net and gender_net. Those net files can be utilized when processing.

To choose whether an individual is male or female, you have got to learn the discriminative features of both classes. The Eigenfaces method is based on the Principal Component Analysis, which is an unsupervised statistical model and not much reasonable for gender prediction .The Fisherfaces method achieves a 98% recognition rate in a subject-independent cross-validation.

In spite of the fact that much potential laid in more profound CNN models (networks with more neuron layers), as it were as of late have it got to be predominant, taking after the dramatic increment in both the computational power (due to Graphical Processing Units).

Data-sets for age and gender estimation from real-world social pictures are in this manner moderately constrained in size and directly no match in estimate with the much larger image classification data-sets. Over fitting is common problem when machine learning based methods are used on such small image collections. This issue is aggravate when considering deep convolutional neural networks due to their tremendous numbers of model parameters.

ABOUT DEEP LEARNING

At its least complex, deep learning can be thought of as a way to automate prescient analytics. Whereas conventional machine learning algorithms are direct, deep learning algorithms are stacked in a hierarchy of expanding complexity and abstraction.

Each algorithm within the hierarchy applies a nonlinear transformation to its input and uses what it learns to make a statistical model as yield. Iterations proceed until the yield has come to a worthy level of precision. The number of processing layers through which data must pass is what propelled the label deep.

To achieve a satisfactory level of accuracy, deep learning programs require get to to gigantic sums of training data and processing power, not one or the other of which were effectively accessible to programmers until the period of big data and cloud computing.

Since deep learning programming can make complex statistical models specifically from its claim iterative output, it is able to form precise prescient models from expansive amounts of unlabeled, unstructured data

Since deep learning models handle data in ways comparative to the human brain, they can be connected to numerous tasks people do. Deep learning is right now utilized in most common **image recognition** tools, **natural language processing** (NLP) and **speech recognition** software

The hardware necessities for deep learning models can as well make limitations. Multicore high-performing graphics processing units (GPUs) and other comparable processing units are required to guarantee improved productivity and diminished time utilization. In any case, these units are costly and utilize expansive sums of energy.
Other hardware prerequisites join random access memory and a hard disk drive (HDD) or RAM-based solid-state drive (SSD).

Classification using Traditional Machine Learning vs. Deep Learning

Classification using a machine learning algorithm has 2 phases:

Training phase: In this phase, we train a machine learning algorithm using a dataset comprised of the images and their corresponding labels.

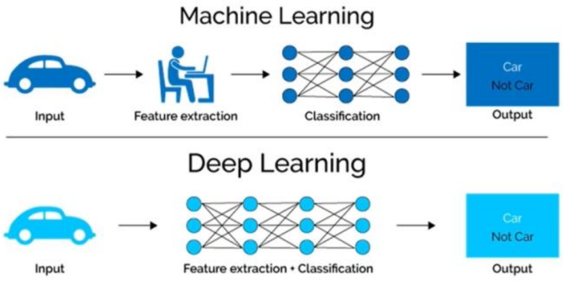
Prediction phase: In this stage, we utilize the trained model to anticipate labels of unseen pictures.

The training phase for an image classification problem has 2 primary steps:

Feature Extraction: In this stage, we utilize domain knowledge to extricate new features that will be utilized by the machine learning algorithm.

Model Training: In this phase, we utilize a clean dataset composed of the images' features and the corresponding labels to train the machine learning model.

The foremost differentiate between conventional machine learning
and deep learning algorithms is inside the feature engineering. In conventional machine learning algorithms, we have to be hand-craft the features. By differentiate,
in deep learning algorithms feature engineering is done automatically by the algorithm.



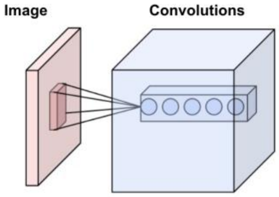
CONVOLUTIONAL NEURAL NETWORKS (CNNS OR CONVNETS)

The Convolution Neural Network (ConvNet / CNN) is deep learning algorithm that is utilized as a feature extractor inside the proposed solution. CNN takes input pictures and allots value to and can recognize between different aspects / objects (learnable weights and predispositions) of the image. ConvNet needs much less pre-processing than other
classification algorithms. Whereas the filters are hand-made in primitive methods, ConvNet can learn these filters / features with satisfactory training.

Convolutional neural networks are an uncommon sort of feed-forward networks. These models are laid out to mimic the conduct of a visual cortex. CNNs perform exceptionally well on visual recognition tasks. CNNs have special layers called **convolutional layers** and **pooling layers** that permit the network to encode certain images properties.

Convolution Layer

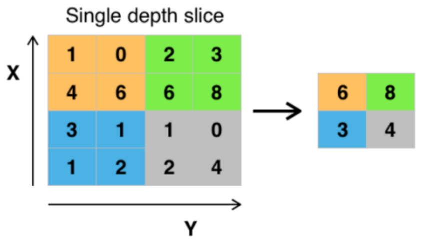
This layer comprises of a set of learnable filters that we slide over the picture spatially, computing dot products between the entries of the filter and the input picture. The channels got to increase to the complete profundity of the input picture. These filters will activate when they see same particular structure within the images.



Neurons of a convolutional layer, connected to their receptive field
(source: Wikipedia)

Pooling Layer

Pooling is a form of non-linear down-sampling. The objective of the pooling layer is to continuously decrease the spatial size of the representation to reduce the amount of parameters and computation within the network, and subsequently to also control overfitting. There are a few functions to execute pooling among which max pooling is the preeminent common one. Pooling is frequently connected with filters of estimate 2x2 connected with a stride of 2 at each depth slice. A pooling layer of estimate 2x2 with stride of 2 shrivels the input picture to a 1/4 of its unique estimate.



Max pooling with a 2x2 filter and stride = 2
(source: Wikipedia)

RelU (Rectified Linear Unit) Activation Function

Activation functions present non-linearity to the model which permits it to memorize complex useful mappings between the inputs and reaction variables. There are very some different activation functions like sigmoid, tanh, RelU, Defective RelU, etc. RelU work could be a piecewise linear function that yields the input straightforwardly in case is positive i.e. > 0, otherwise, it'll yield zero.In hone RelU activation function is connected right after a convolution layer and after that that yield is max pooled.

Loss function

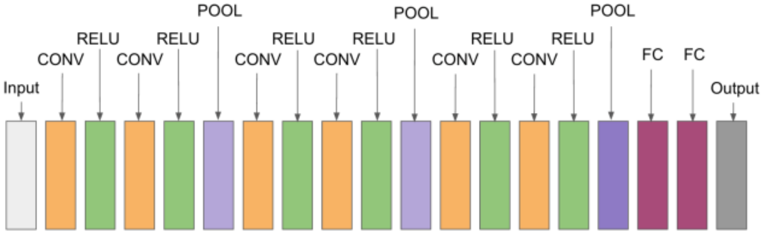
Loss is nothing but a forecast blunder of Neural Net. The loss function will yield a higher number in case the expectations are off the actual target values while something else it'll yield a lower number.

Convolution	ReLU	Pooling
• Connections sparsity reduces overfitting.	• Introduces non linearity.	• Reduces dimensions and computation
• Conv and pooling gives location invariant feature detection.	• Speeds up training, faster to compute.	• Reduces overfitting.
• Parameter sharing		• Makes the model tolerant towards small distortion and variations.

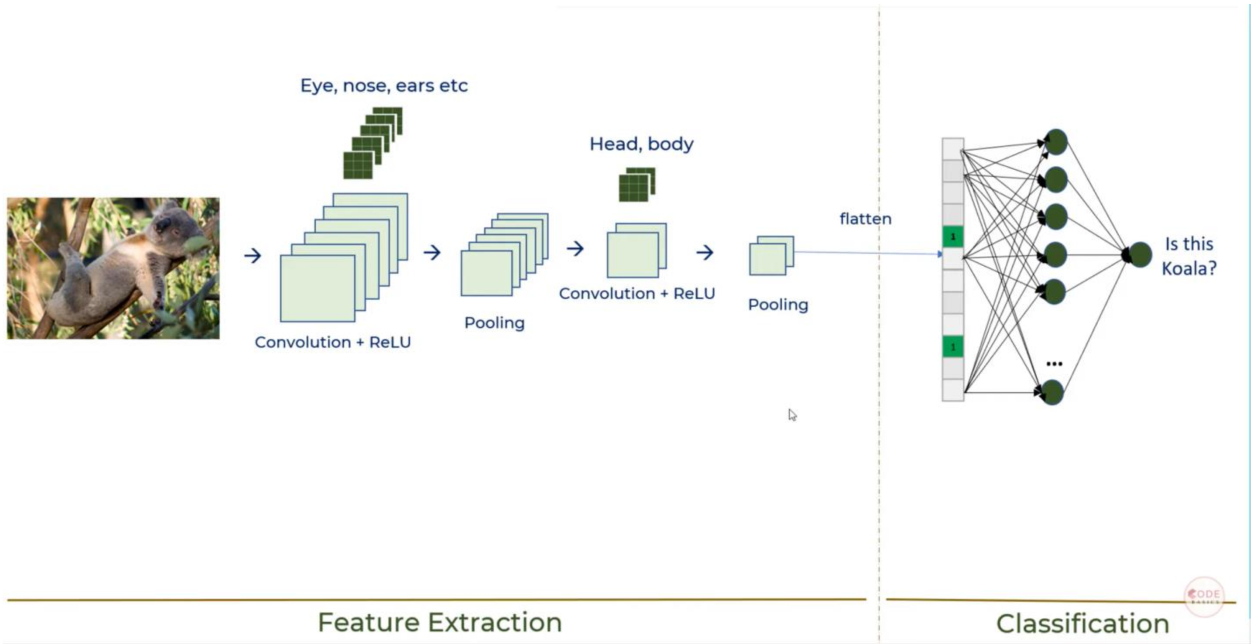
The best engineering of a convolutional neural systems begins with an input layer (pictures) taken after by a arrangement of convolutional layers and pooling layers, and closes with fully-connected layers. The convolutional layers are as a rule taken after by one layer of ReLU actuation functions.

The convolutional, pooling and ReLU layers act as
learnable features extractors, whereas the completely connected layers acts as a machine learning classifier. Besides, the early layers of the network encode generic patterns of
the pictures, whereas afterward layers encode the details patterns of the images.

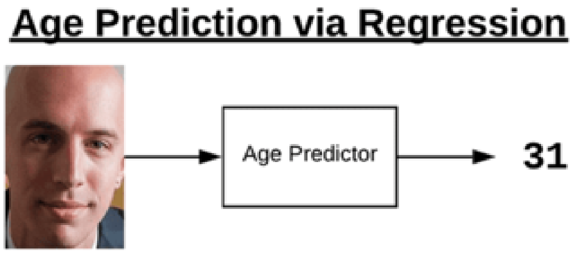
Note that as it were the convolutional layers and fully-connected layers have weights. These weights are learned within the training phase.

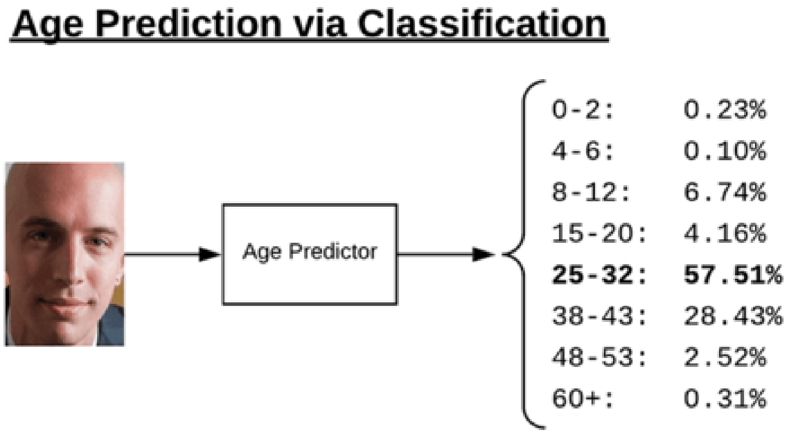


Example of a CNN Architecture



WHY DON’T WE TREATING AGE PREDICTION AS A REGRESSION PROBLEM?





0-2:	0.23%
4-6:	0.10%
8-12:	6.74%
15-20:	4.16%
25-32:	**57.51%**
38-43:	28.43%
48-53:	2.52%
60+:	0.31%

Age prediction with deep learning could be framed as a regression or classification problem.

By discretized ages into “buckets,” this manner treating age prediction as a classification problem — why not outline it as a regression problem instead.

Whereas individuals are intrinsically appalling at foreseeing single age esteem, we are truly exceptionally incredible at predicting age brackets.And in the event that a human battles to precisely anticipate the age of an individual, at that point doubtlessly a machine will battle as well.

Once start treating age prediction as a regression problem, it gets to be basically harder for a model to accurately anticipate a single value speaking to that person’s picture.

However, in the event treating it as a classification problem, characterizing buckets/age brackets for the model, the age predictor model gets to be simpler to train, often yielding significantly higher precision than regression-based prediction alone.

WHY CAFFE MODEL

Caffe is maintained and created by the Berkeley Vision and Learning Center (BVLC) with the assistance of an active community of contributors on GitHub. Being especially motivated by large-scale visual recognition, where a particular sort of deep architecture has accomplished a commanding lead on the state-of-the-art. These Convolutional Neural Networks, or CNNs, are arranged by means of back-propagation through layers of convolutional filters and other operations such as rectification and pooling

But trained models alone are not adequate for quick research advance and developing commercial applications, and few toolboxes offer really off-the-shelf deployment of state-of-the-art models—and those that do are frequently not computationally effective and hence unsuitable for commercial deployment. To address such issues, Caffe is used, a totally opensource framework that manages clear access to significant architectures. The code is composed in clean, effective C++, with CUDA utilized for GPU computation, and about total, well-supported ties to Python/Numpy and MATLAB. Caffe gives a total toolkit for training, testing, finetuning, and deploying models, with well-documented cases for all of these tasks.

Modularity

The software is outlined from the starting to be as modular as possible, permitting easy expansion to new data formats, network layers, and loss functions. Lots of layers and loss functions are as of now implemented, and ample examples show how these are composed into trainable recognition systems for different tasks.

Separation of representation and implementation

Caffe model definitions are composed as config files utilizing the Convention Buffer language. Caffe underpins network architectures within the shape of arbitrary directed acyclic graphs. Upon instantiation, Caffe saves precisely as much memory as required for the network, and abstracts from its underlying location in host or GPU. Switching between a CPU and GPU implementation is exactly one function call.

Test coverage

Each single module in Caffe includes a test, and no new code is recognized into the project without comparing tests. This grants quick changes and refactoring of the codebase, and confers a welcome feeling of tranquility to the researchers utilizing the code.

Python and MATLAB bindings

For quick prototyping and interfacing with existing research code, Caffe gives Python and MATLAB bindings. Both languages may be utilized to build networks and classify inputs. The Python bindings moreover uncover the solver module for simple prototyping of new training procedures.

Pre-trained reference models.

Caffe gives reference models for visual errands, counting the landmark “AlexNet” ImageNet model with varieties and the R-CNN detection model .

Caffe varies from other modern CNN systems in two major ways:

(1) The execution is completely C++ based, which facilitates integration into existing C++ frameworks and interfacing common in industry. The CPU mode expels the hindrance of specialized hardware for deployment and tests once a model is trained.

(2) Reference models are given off-the-shelf for speedy experimentation with state-of-the-art results, without requiring for costly re-learning. By finetuning for related assignments, these models provide a warmstart to modern research and applications. Data is taken care of in mini-batches that pass through the network sequentially. Finetuning, the adjustment of an existing model to new architectures or information, could be a standard method in Caffe. From a portrayal of an existing network and a model definition for the new network, Caffe finetunes the old model weights for the new assignment and initializes new weights as required.

WORKING OF CAFFE MODEL

Deep networks are compositional models that are naturally spoken to as a collection of inter-connected layers that work on chunks of data. The network characterizes the entire model bottom-to-top from input data to loss. As data and derivatives stream through the network inside the forward and backward passes Caffe stores, communicates, and controls the data as blobs

Blobs

1. A Blob is a wrapper over the genuine data being processed and passed along by Caffe.

2. The blob is the standard cluster of array and bound together memory interface for the framework.

3. The subtle elements of blob portray how data is put away and communicated in and over layers and nets.

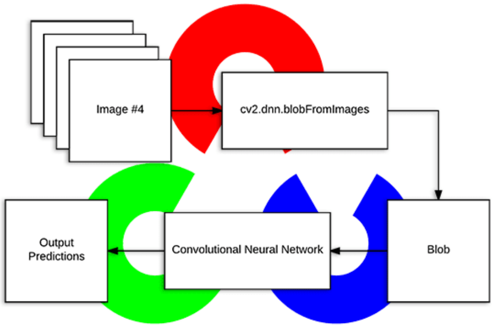
4. Blobs give a unified memory interface holding data; e.g., bunches of pictures, model parameters, and derivatives for optimization.

5. The conventional blob measurements for batches of picture data are number N x channel K x stature H x width W. Blob memory is row-major in format, so the final / furthest right measurement changes quickest. For case, in a 4D blob, the value at list (n, k, h, w) is physically located at index ((n * K + k) * H + h) * W + w

6. As being regularly inquisitive about the values as well as the gradients of the blob, a Blob stores two chunks of memories, data and diff. The past is the normal information that we pass along, and the last mentioned is the gradient computed by the network.

7. The reason for such design is that, a Blob uses a SyncedMem class to synchronize values between the CPU and GPU in order to hide the synchronization details and to play down data transfer.

8. In hone when GPUs are present, one loads information from the disk to a blob in CPU code, calls a device kernel to do GPU computation, and ships the blob off to the taking after layer, neglecting low-level details while keeping up a high level of performance



Layer

1. The layer comes taking after as the establishment of both model and computation.

2. The layer is the quintessence of a model and the fundamental unit of computation. Layers convolve channels, pool, take internal products, apply nonlinearities like rectified-linear and sigmoid and other element wise transformations, normalize, stack information, and compute losses like softmax and hinge .

3. A layer takes input through bottom connections and makes output through top connections.

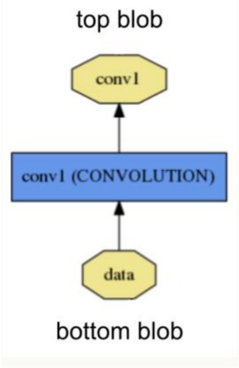
4. Each layer type defines three critical computations: setup, forward, and backward.

Setup: initialize the layer and its connections once at model initialization.

Forward: given input from bottom compute the output and send to the top.

Backward: given the gradient w.r.t. the top output compute the gradient w.r.t. to the input and send to the bottom. A layer with parameters computes the gradient w.r.t. to its parameters and stores it internally.

Layers have two key duties for the operation of the network as a entirety: a **forward pass** that takes the inputs and produces the outputs, and a **backward pass** that takes the gradient with regard to the output, and computes the gradients with respect to the parameters and to the inputs, which are in turn back-propagated to earlier layers. These passes are basically the composition of each layer’s forward and backward.



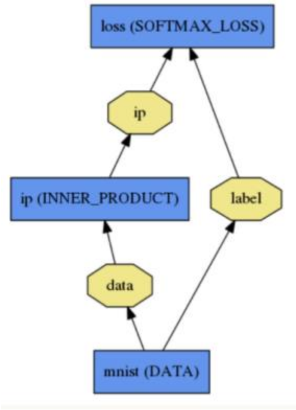
Net

1. The net takes after as the collection and connection of layers.

2. The net mutually characterizes a function and its gradient by composition and auto-differentiation. The composition of each layer's output computes the function to do a given task, and the composition of each layer's backward computes the gradient from the loss to memorize the task. Caffe models are end-to-end machine learning engines.

3. The net could be a set of layers related in a computation chart – a directed acyclic graph (DAG) to be redress. Caffe does all the bookkeeping for any DAG of layers to guarantee rightness of the forward and backward passes. A standard net starts with a data layer that loads from disk and closes with a loss layer that computes the objective for a assignment such as classification or reconstruction.

4. The net is characterized as a set of layers and their affiliations in a plaintext modeling language.

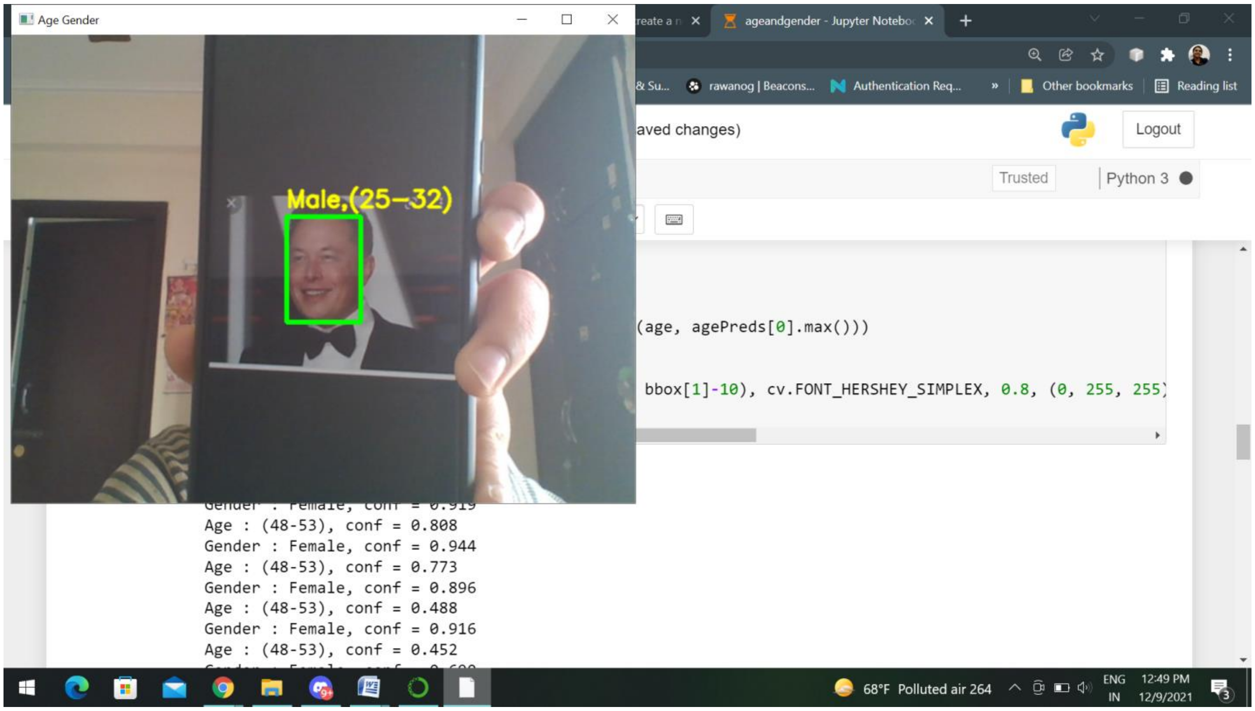


Model initialization is handled by Net::Init(). The initialization mainly does two things: scaffolding the by and large DAG by making the blobs and layers , and calls the layers' SetUp() function. It moreover does a set of other bookkeeping things, such as approving the rightness of the overall network architecture

Blobs and layers cover up execution details from the model definition. After development, the network is run on either CPU or GPU by setting a single switch characterized in Caffe::mode() and set by Caffe::set_mode(). Layers come with comparing CPU and

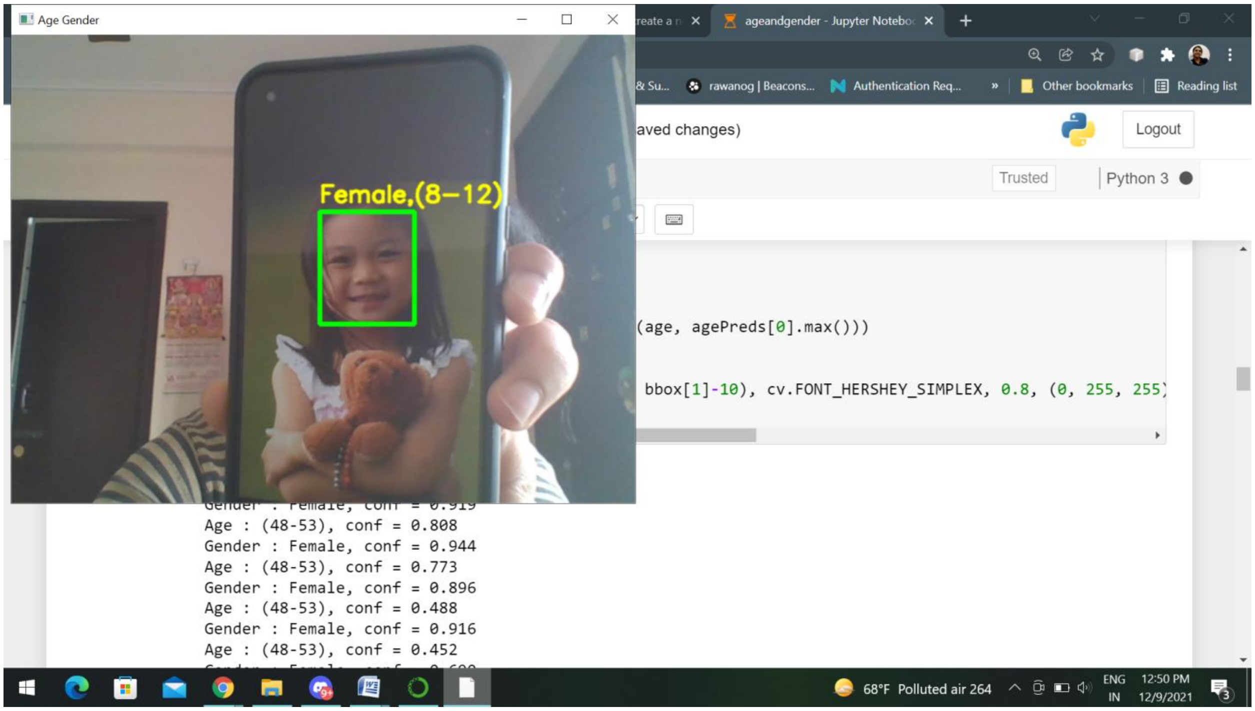
GPU schedules that create indistinguishable results . The CPU / GPU switch is consistent and free of the model definition. For investigate and deployment alike it is best to partition model and implementation.

OUTPUT OF THE REAL TIME AGE AND GENDER PREDICTION MODEL-



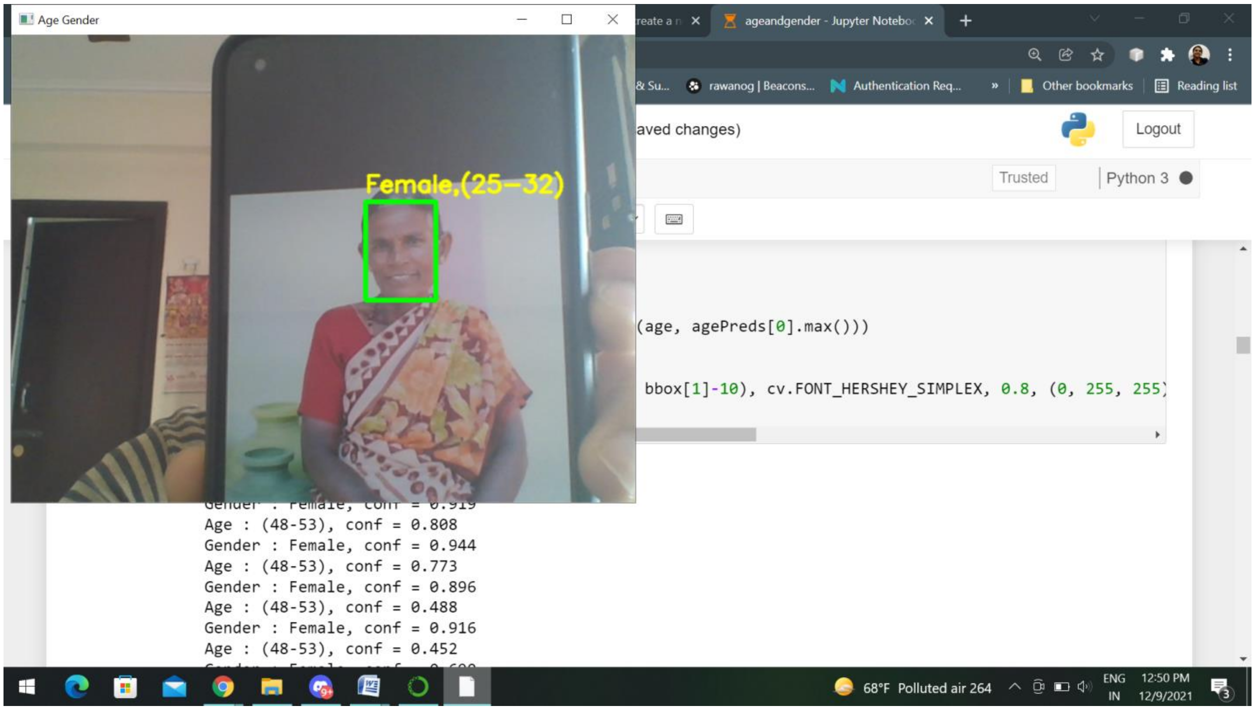
```
(age, agePreds[0].max()))

 bbox[1]-10), cv.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255)
```

```
Gender : Female, conf = 0.919
Age : (48-53), conf = 0.808
Gender : Female, conf = 0.944
Age : (48-53), conf = 0.773
Gender : Female, conf = 0.896
Age : (48-53), conf = 0.488
Gender : Female, conf = 0.916
Age : (48-53), conf = 0.452
Gender : Female, conf = 0.690
```



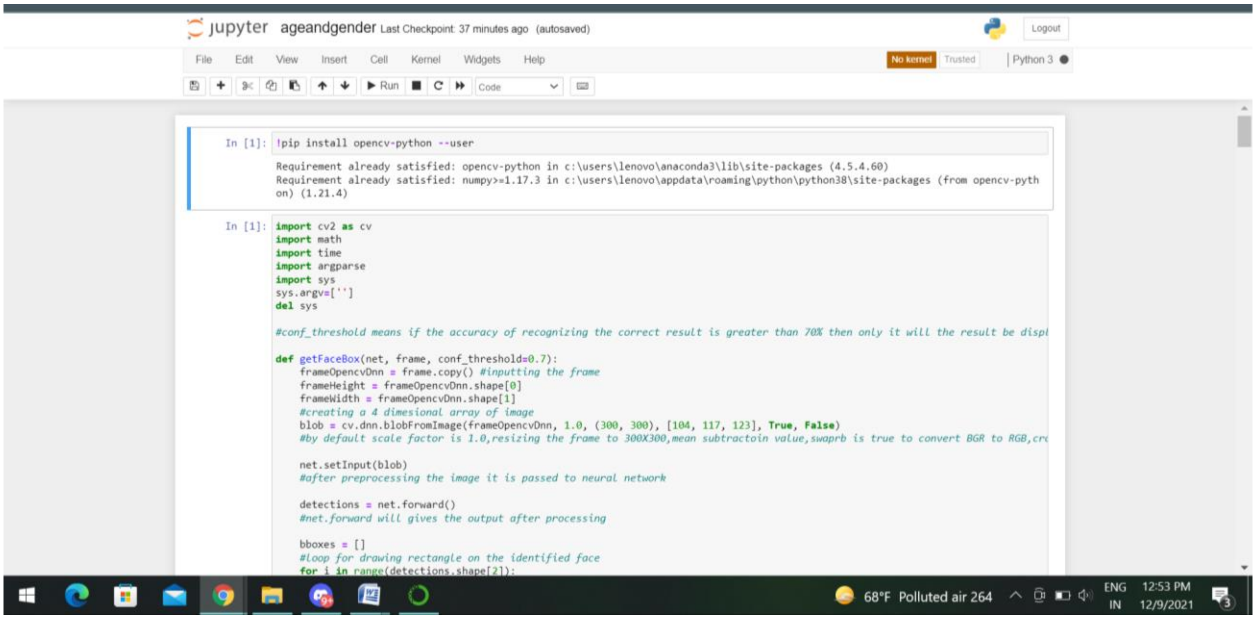
```
(age, agePreds[0].max()))

 bbox[1]-10), cv.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255)
```

```
Gender : Female, conf = 0.919
Age : (48-53), conf = 0.808
Gender : Female, conf = 0.944
Age : (48-53), conf = 0.773
Gender : Female, conf = 0.896
Age : (48-53), conf = 0.488
Gender : Female, conf = 0.916
Age : (48-53), conf = 0.452
Gender : Female, conf = 0.699
```



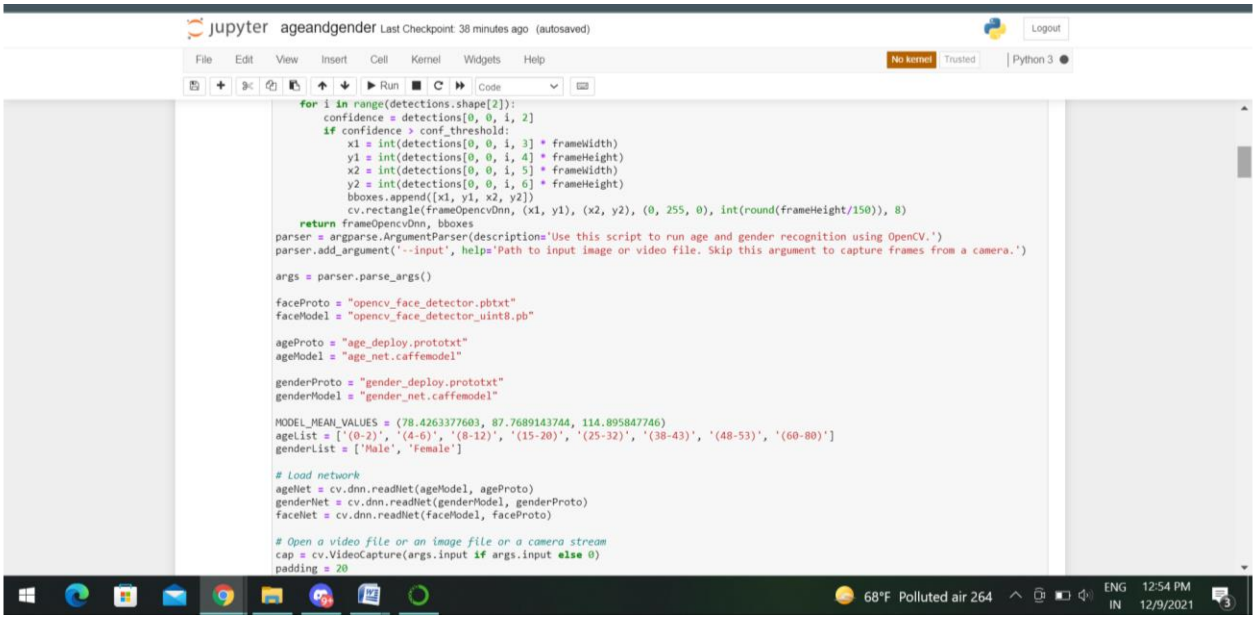
```
(age, agePreds[0].max()))

 bbox[1]-10), cv.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255)
```

```
Gender : Female, conf = 0.919
Age : (48-53), conf = 0.808
Gender : Female, conf = 0.944
Age : (48-53), conf = 0.773
Gender : Female, conf = 0.896
Age : (48-53), conf = 0.488
Gender : Female, conf = 0.916
Age : (48-53), conf = 0.452
Gender : Female, conf = 0.699
```

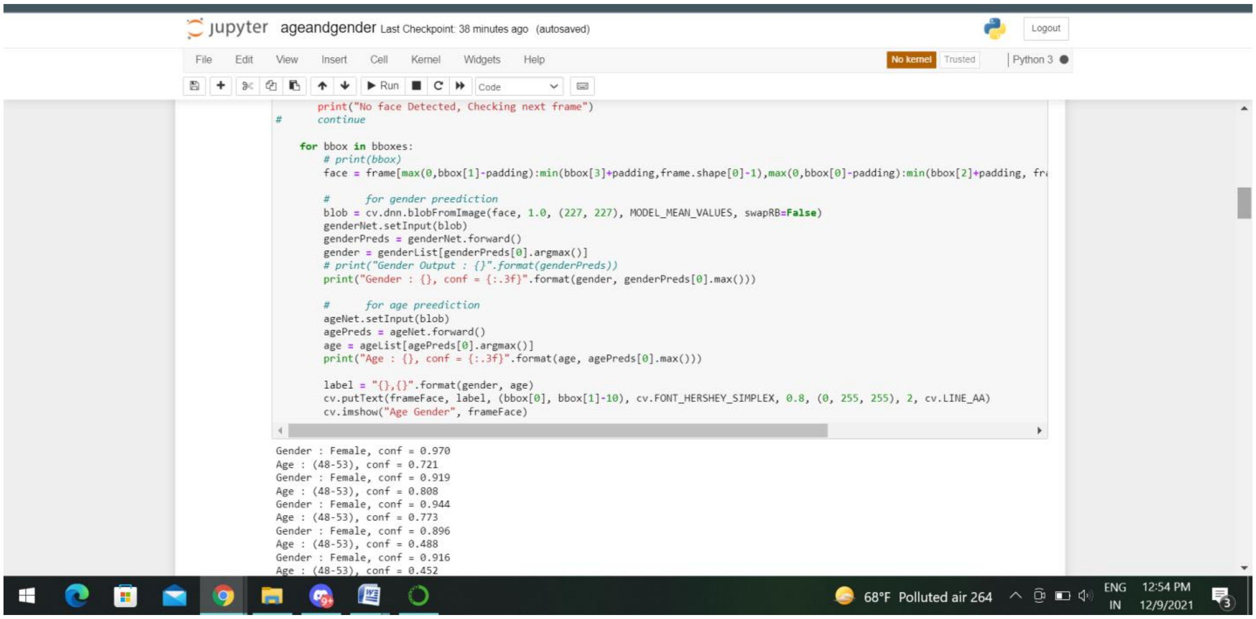

jupyter ageandgender Last Checkpoint: 37 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
No kernel
Trusted
Python 3
Run
Code
68°F Polluted air 264
ENG IN
12:53 PM 12/9/2021

```
In [1]: !pip install opencv-python --user

        Requirement already satisfied: opencv-python in c:\users\lenovo\anaconda3\lib\site-packages (4.5.4.60)
        Requirement already satisfied: numpy>=1.17.3 in c:\users\lenovo\appdata\roaming\python\python38\site-packages (from opencv-pyth
        on) (1.21.4)
```

```
In [1]: import cv2 as cv
        import math
        import time
        import argparse
        import sys
        sys.argv=['']
        del sys

        #conf_threshold means if the accuracy of recognizing the correct result is greater than 70% then only it will the result be displ

        def getFaceBox(net, frame, conf_threshold=0.7):
            frameOpencvDnn = frame.copy() #inputting the frame
            frameHeight = frameOpencvDnn.shape[0]
            frameWidth = frameOpencvDnn.shape[1]
            #creating a 4 dimesional array of image
            blob = cv.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117, 123], True, False)
            #by default scale factor is 1.0,resizing the frame to 300X300,mean subtractoin value,swaprb is true to convert BGR to RGB,cro

            net.setInput(blob)
            #after preprocessing the image it is passed to neural network

            detections = net.forward()
            #net.forward will gives the output after processing

            bboxes = []
            #loop for drawing rectangle on the identified face
            for i in range(detections.shape[2]):
```


jupyter ageandgender Last Checkpoint: 38 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
No kernel
Trusted
Python 3
Run
Code
68°F Polluted air 264
ENG IN
12:54 PM 12/9/2021

```
    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > conf_threshold:
            x1 = int(detections[0, 0, i, 3] * frameWidth)
            y1 = int(detections[0, 0, i, 4] * frameHeight)
            x2 = int(detections[0, 0, i, 5] * frameWidth)
            y2 = int(detections[0, 0, i, 6] * frameHeight)
            bboxes.append([x1, y1, x2, y2])
            cv.rectangle(frameOpencvDnn, (x1, y1), (x2, y2), (0, 255, 0), int(round(frameHeight/150)), 8)
    return frameOpencvDnn, bboxes
parser = argparse.ArgumentParser(description='Use this script to run age and gender recognition using OpenCV.')
parser.add_argument('--input', help='Path to input image or video file. Skip this argument to capture frames from a camera.')

args = parser.parse_args()

faceProto = "opencv_face_detector.pbtxt"
faceModel = "opencv_face_detector_uint8.pb"

ageProto = "age_deploy.prototxt"
ageModel = "age_net.caffemodel"

genderProto = "gender_deploy.prototxt"
genderModel = "gender_net.caffemodel"

MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)
ageList = ['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-53)', '(60-80)']
genderList = ['Male', 'Female']

# Load network
ageNet = cv.dnn.readNet(ageModel, ageProto)
genderNet = cv.dnn.readNet(genderModel, genderProto)
faceNet = cv.dnn.readNet(faceModel, faceProto)

# Open a video file or an image file or a camera stream
cap = cv.VideoCapture(args.input if args.input else 0)
padding = 20
```



```
        print("No face Detected, Checking next frame")
#       continue

    for bbox in bboxes:
        # print(bbox)
        face = frame[max(0,bbox[1]-padding):min(bbox[3]+padding,frame.shape[0]-1),max(0,bbox[0]-padding):min(bbox[2]+padding, fr

        #       for gender preediction
        blob = cv.dnn.blobFromImage(face, 1.0, (227, 227), MODEL_MEAN_VALUES, swapRB=False)
        genderNet.setInput(blob)
        genderPreds = genderNet.forward()
        gender = genderList[genderPreds[0].argmax()]
        # print("Gender Output : {}".format(genderPreds))
        print("Gender : {}, conf = {:.3f}".format(gender, genderPreds[0].max()))

        #       for age preediction
        ageNet.setInput(blob)
        agePreds = ageNet.forward()
        age = ageList[agePreds[0].argmax()]
        print("Age : {}, conf = {:.3f}".format(age, agePreds[0].max()))

        label = "{},{}".format(gender, age)
        cv.putText(frameFace, label, (bbox[0], bbox[1]-10), cv.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2, cv.LINE_AA)
        cv.imshow("Age Gender", frameFace)
```

```
Gender : Female, conf = 0.970
Age : (48-53), conf = 0.721
Gender : Female, conf = 0.919
Age : (48-53), conf = 0.808
Gender : Female, conf = 0.944
Age : (48-53), conf = 0.773
Gender : Female, conf = 0.896
Age : (48-53), conf = 0.488
Gender : Female, conf = 0.916
Age : (48-53), conf = 0.452
```