INTRODUCTION

1.1 ABOUT THE PROJECT

Age and Gender prediction are used extensively in the field of computer vision for surveillance. Advancement in computer vision makes this prediction even more practical and open to all, thus enables the world to come up with datasets, two of which we are using in this project. The IMDB-WIKI dataset for the prediction of gender and the UTK face dataset for the prediction of age. The IMDB-WIKI dataset has more than 2lakh faces divided into two classes, male and female. The UTK face dataset has faces more than 20000 and age ranging from 1 to 115.

In this project, I am using a Convolution Neural Network (CNN) with for gender classification and CNN with ResNet50 architecture to predict age. CNN is a Neural Network algorithm that extracts the deep features from the image and specifies the desired output at the final layers. Age prediction is approximately near to the real values with a five difference in both ways. Gender prediction is accurate in all the test data presented to the model. Validating with arguments shows no change in training and validation. Our model successfully executed with approximately 91% in gender classification and mean absolute error of approx 5 in age prediction that can be further advanced with pipelining with other classification models and much larger real-world datasets.

1.2 INTRODUCTION TO THE PROBLEM DOMAIN

As technology increases, the applications that combine the advanced fields of pattern recognition and image processing are used to find age and gender. In today's world, age plays a prominent role, when you appear for an interview, health check-ups. The information of age is used in many governments, private and advertising sector organization to find the culprits, employee eligible for the job, audience to be targeted for their publicity of product respectively. However, it's not that easy to find the age of a person, and there are constraints that restrict us from seeing the correct age among the set of images. Human face contains lot of information through their expressions. These expressions confuse us while finding the age, and as their expression change, the facial feature differs, resulting in either higher or low than the people's ideal age.

The age estimation plays a prominent role in the applications like biometric evaluation, 3d face marking, virtual makeup, and virtual try-on applications for jewellery and eye-ware by mapping the face according to the age found. Lenskart is such an application that gives the try-on option for their customers. Age estimation is a subfield of face recognition and face tracking which in combination can predict the health of the individual. Many health care applications use this mechanism to keep track of health by monitoring their daily activities. China uses this face detection technique in service driver identification and jaywalker identification. Some other countries use it for worshipper identification and advertising etc.

Finding the correct dataset for training the model is a crucial task. Since the real-time data is massive, the computation and the time to prepare the model are high. It's been a tough task after implementing several methods from machine learning, and the accuracy increases drastically. By eliminating the barrier of expressions, we have the possibility to find the best features leading to an accurate measure of gender and age. To predict the age and gender, we use a vital range of machine learning and deep learning algorithms. CNN (convolution neural network) is one of the most used techniques for age and gender detection. In this project, I am using keras, Ktrain with CNN to predict the age and gender of any given person's image.

1.3 APPLICATIONS

Some of the real world applications of gender and age prediction are:-

- Identity verification
- Biometrics
- Video surveillance
- Human computer interaction
- Electronic customer
- Advertisement
- Social understanding
- Item recommendation
- Movie recommendation
- Songs recommendation

This project aims to provide an application that will help different users for different purposes. The scope and applicability of this application make this system more efficient and more accurate.

REQUIREMENT ANALYSIS SPECIFICATION

2.1 INTRODUCTION

Requirement Analysis Document is the starting point of the software requirement activity. Requirement Analysis, also known as Requirement Engineering, defines user expectations for a new software being built or modified. In software engineering. Requirement analysis includes those tasks that determine the needs or conditions to meet, considering the possibly conflicting requirements, analyzing, documenting, validating, and managing software needs.

The Software Requirement Specification is a process of translating the clients' minds into a formal document. Thus, the step's result is a set of formally specified requirements, which hopefully are complete and consistent, while the input has none of these properties.

2.2 EXISTING SYSTEM

The existing systems are based on a single model for prediction of both age and gender, also have very less accuracy. To improve the accuracy of the model one has to use the dataset which will have both the features in it. The existing models can only be used in the notebook or from the command line of that particular system.

2.3 PROPOSED SYSTEM

The proposed system has two different models for gender and age prediction. So in the future it will be very easy to improve accuracy with the any available dataset. The accuracy achieved was 92% approx which can further be improved by using the weights and the architecture of the model. The mean square root error of age model is approx 5 which can be decreased further with real world datasets. The model prediction can be used in any system as well as from any mobile as we are hosting it for free in the internet.

2.4 FEASIBILITY STUDY

This study plan discusses the schedule for the project and contains the various phases of the project. The Various Phases of the Project**:**

S.NO	PHASES
1	Requirement Specification
2	Requirement document specification
3	Design analysis
4	Coding
5	Testing
6	Maintenance

2.5 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENT

Functional requirements (FR) can define specific behaviours, features, and use cases of the product. FRs can be broken down into high-level requirements (HLR) and low-level requirements (LLR). HLRs can be abstract statements defining an overall feature needed, and LLRs are more detailed descriptions of what the product needs in order to realize the HLR. The project will keep a log of changes as well as the failures of predicting the image or the framework. The project will have a website hosted in the localhost as well the internet. We just need to upload an image and the underlying models will predict and classify.

Non-functional requirements (NFR), on the other hand, assess system properties and constraints such as performance, scalability, and security. In short, FRs describes what the system should do and NFRs describe how the system should perform it. The interface between a user and the automated system shall have a maximum response time of 15 seconds unless noted by an exception. The project can predict from any type of image like jpg, jpeg, png, gif etc. Even a video feed of a single person is working. Anyone can use the project from their mobile and laptop as I will be hosting the project in the Amazon AWS.

NFRs are important because they pinpoint areas that affect the health and wellbeing of the system as a whole rather than just a single feature in a module. Fulfilling them requires not only careful design decisions in the beginning, but also continuous effort throughout the entire software development process.

2.6 SOFTWARE REQUIREMENTS

OPERATING SYSTEM	Windows 7 or more
FRONT END	Django
BACK END	Keras, Ktrain
IDE	Jupyter Notebook, Visual Studio
LIBRARIES	Pillow, Scipy, Matplotlib, Numpy

2.7 HARDWARE REQUIREMENTS

SYSTEM	Intel i5 or more
RAM	4GB or more
HARD DISK	150GB or more

SYSTEM DESIGN

3.1 ABOUT DATASET

FOR GENDER PREDICTION:-

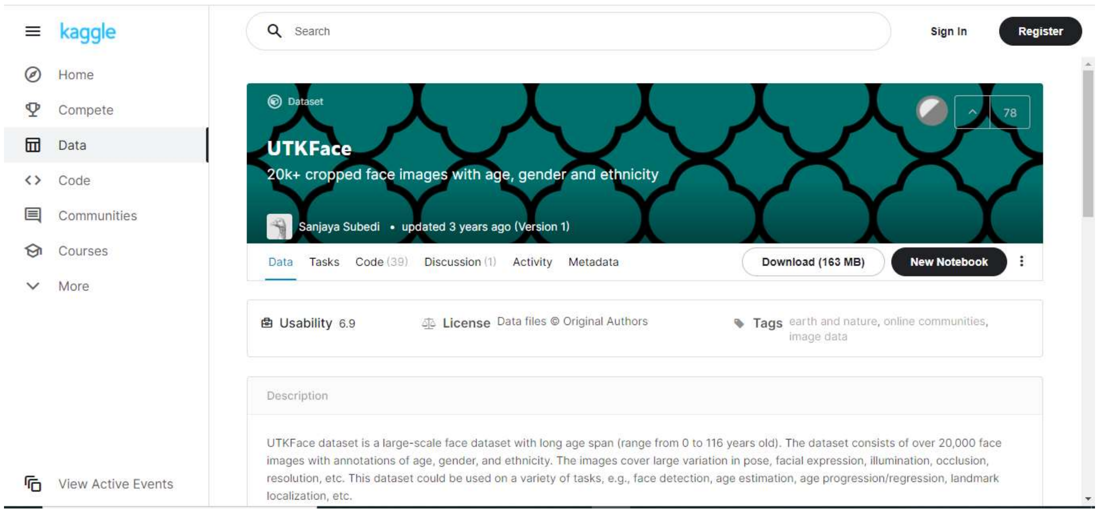
The IMDB-WIKI dataset is the largest dataset available in the internet. The dataset is divided into three labels. Train, test and validation. Each label has two classes- Male and Female. The dataset has more than 200000 images.

- Train label –
  - Female - 92845 images
  - Male - 67155 images
- Validation label –
  - Female - 13778 images
  - Male - 8820 images
- Test label –
  - Female - 11542 images
  - Male - 8459 images

FOR AGE PREDICTION:-

UTKFace dataset is a large-scale face dataset with long age span (range from 0 to 116 years old). The dataset consists of over 20,000 face images with annotations of age, gender, and ethnicity. The images cover large variation in pose, facial expression, illumination, occlusion, resolution, etc. This dataset could be used on a variety of tasks, e.g., face detection, age estimation, age progression/regression, landmark localization, etc.

- Filenames are as - [age]_[gender]_[race]_[date].jpg
- Age is extracted using Python RegEx.



UTKFace
20k+ cropped face images with age, gender and ethnicity
Sanjaya Subedi • updated 3 years ago (Version 1)

Usability 6.9
License Data files © Original Authors
Tags earth and nature, online communities, image data

UTKFace dataset is a large-scale face dataset with long age span (range from 0 to 116 years old). The dataset consists of over 20,000 face images with annotations of age, gender, and ethnicity. The images cover large variation in pose, facial expression, illumination, occlusion, resolution, etc. This dataset could be used on a variety of tasks, e.g., face detection, age estimation, age progression/regression, landmark localization, etc.

Figure 1: UTK Face Dataset

3.2UMLDIAGRAMS

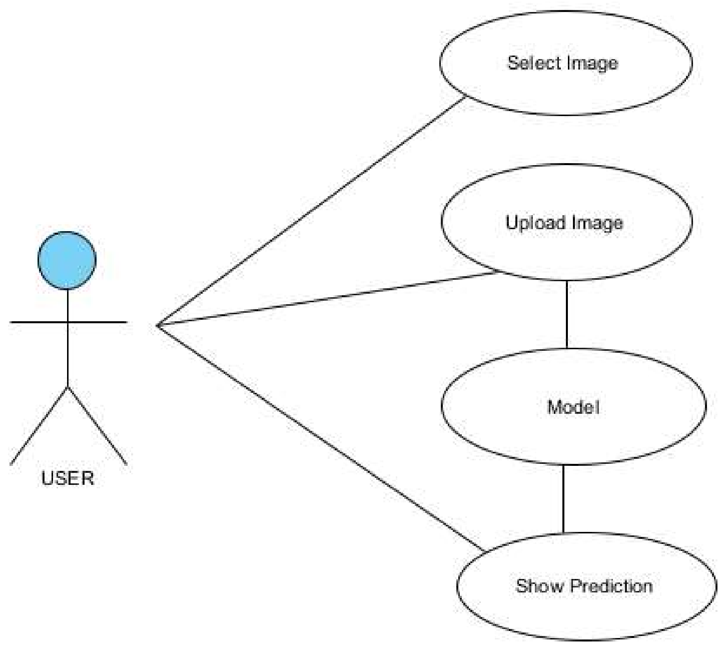


Figure 2: Use Case Diagram

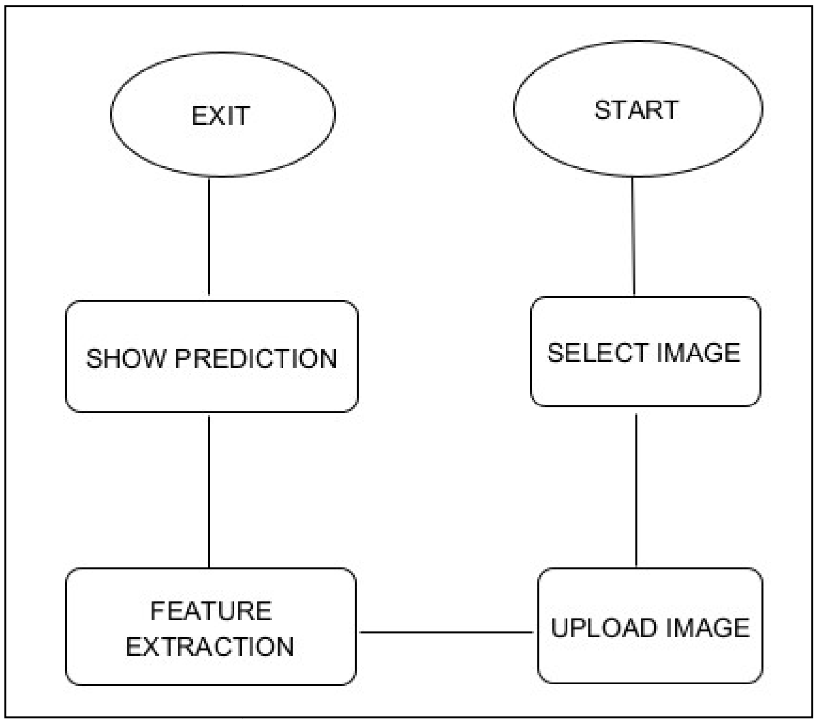


Figure 3: System Architecture

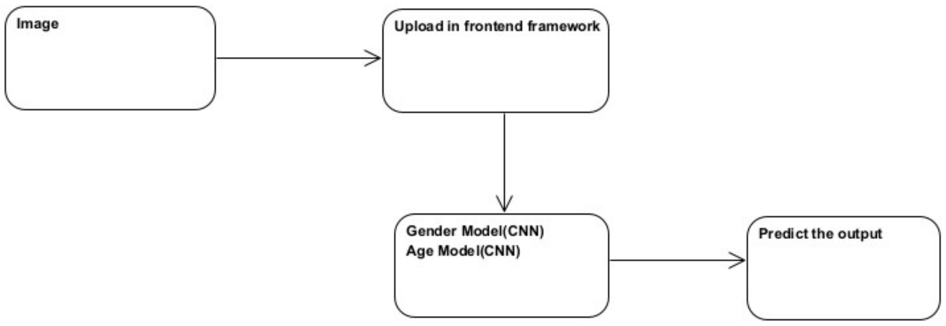


Figure 4: Activity Diagram

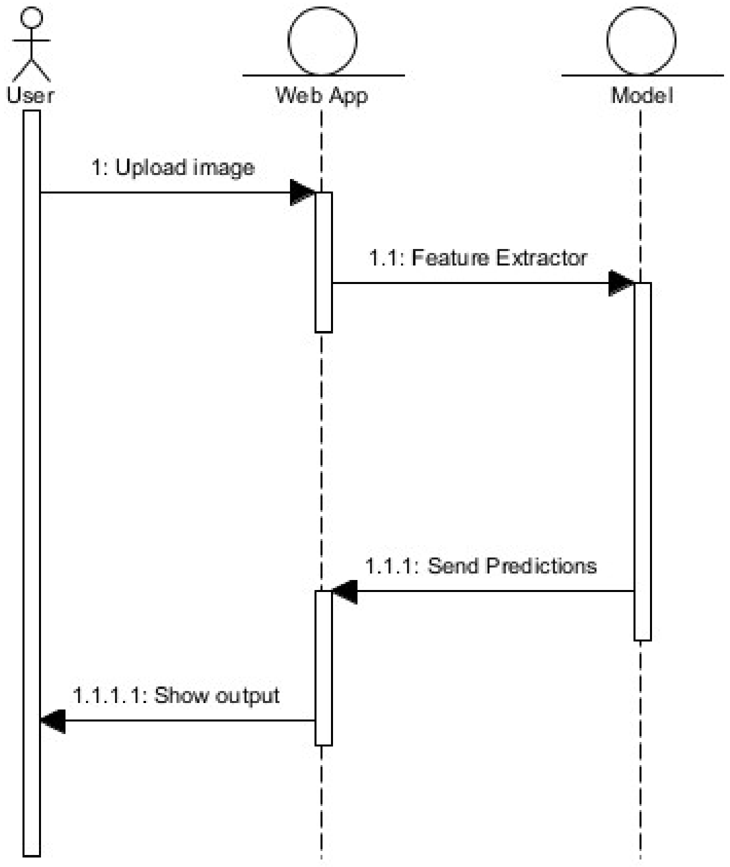


Figure 5: Sequence Diagram

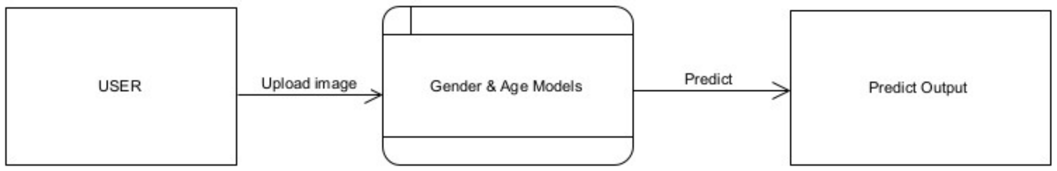


Figure 6: Data Flow Diagram

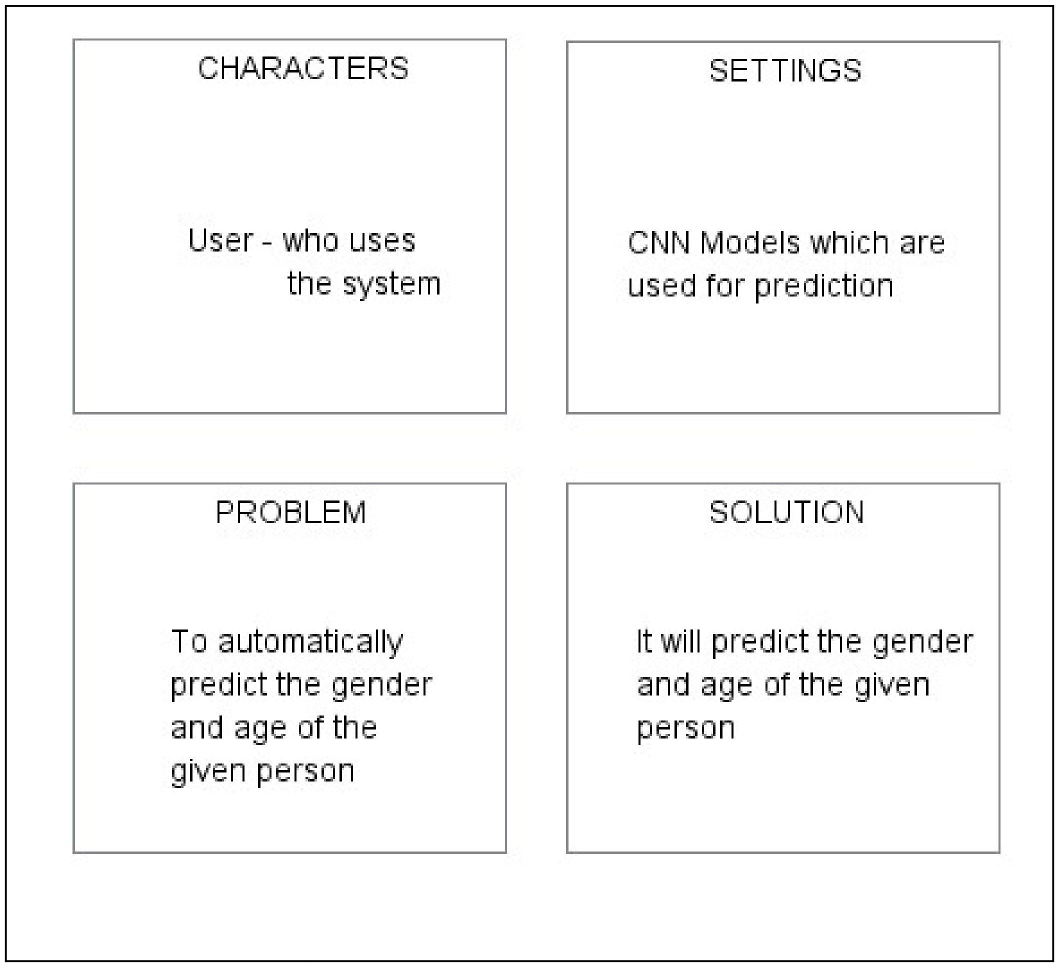

CHARACTERS
User - who uses the system
SETTINGS
CNN Models which are used for prediction
PROBLEM
To automatically predict the gender and age of the given person
SOLUTION
It will predict the gender and age of the given person

Figure 7: User Story

IMPLEMENTATION PLANNING AND DETAILS

4.1 INTRODUCTION

Django

Django is a Python-based free and open-source web framework that follows the model-template-views (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an American independent organization established as a 501(c)(3) non-profit. Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "plug ability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models. Some well known sites that use Django include PBS, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket and Nextdoor. The core features of Django are:-

- A lightweight and standalone web server for development and testing.
- A form serialization and validation system that can translate between HTML forms and values suitable for storage in the database.
- A template system that utilizes the concept of inheritance borrowed from object-oriented programming.
- A caching framework that can use any of several cache methods support for middleware classes that can intervene at various stages of request processing and carry out custom functions.

Python

Python is a high-level programming language object-oriented, interpreted, with dynamic semantics. Its high-level built-in data structures, linked with dynamic- typing and dynamic binding, make it very engaging for Rapid Application Development and use as scripting to connect existing components. Python's simple, easy-to-learnsyntax features readability and thereforedecreases the cost of program maintenance. Python provides modules andpackages, which encourages evenly program modularity and code re-use. The Python interpreter and the vast standard library are available in source or binary form without cost for all major platforms and freely distributed.

Machine Learning

Machine learning is a vast branch of artificial intelligence (AI) centered on building applications that learn from data and develop their accuracy over time without being programmed. In the data science field, an algorithm is a series of statistical processing levels. In the machine learning world, algorithms are 'trained' to find patterns and features in extensive quantities of data to make conclusions and predictions based on new data. The better the algorithm, the more accurate the findings.

Ktrain

ktrain is a lightweight wrapper for the deep learning library TensorFlow Keras (and other libraries) to help build, train, and deploy neural networks and other machine learning models. Inspired by ML framework extensions like fastai and ludwig, ktrain is designed to make deep learning and AI more accessible and easier to apply for both newcomers and experienced practitioners. With only a few lines of code, ktrain allows you to easily and quickly employ fast, accurate, and easy-to-use pre-canned models for text, vision, graph, and tabular data.

Keras

Keras is an open-source software library that acts as an interface for the TensorFlow library. Keras is a deep learning API written in Python, running onTensorFlow's machine learning platform. It was acquired with a focus on enabling fast experimentation.

ResNet

A residual neural network, ResNet, is an artificial neural network (ANN) built on constructs associated with pyramidal cells in the cerebral cortex. Residual neural networks do this by appropriating skip connections or shortcuts to take over any layers. Typical ResNet models are performed with double- or triple-layer skips containing nonlinearities and batch normalization in between. An added weight matrix may be used to learn the skip weights and residual neural networks' circumstances; a non-residual network may be described as a plain network.

Jupyter Notebook

The Jupyter Python Notebook is freely available, an open-source web application that enables you to create and share documents that combine live code, equations, visualizations, and narrative text. It provides data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning.

SciPy

SciPy is a free and open-source Python library used for scientific computing and technical computing.

NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

TensorFlow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. Tensorflow is a symbolic math library based on dataflow and differentiable programming.

Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits.

Convolutional Neural Networks

Sounds like a weird combination of biology and math with a little CS sprinkled in, but these networks have been some of the most influential innovations in the field of computer vision and image processing.

What Does a Computer See?

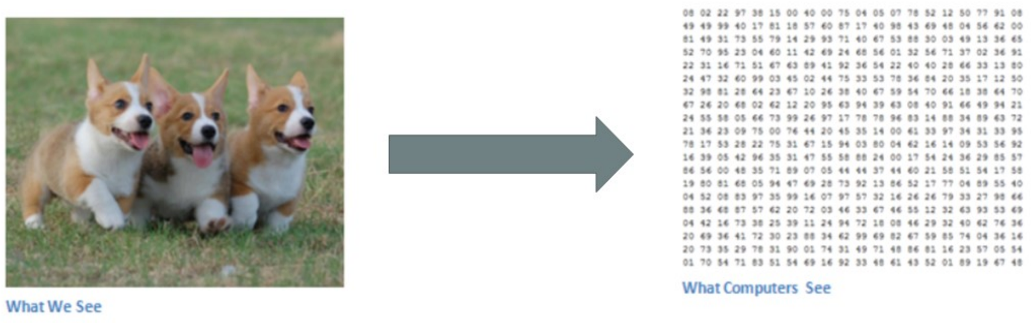


Figure 8: How computer reads a image?

Let’s say we have a colour image in JPG form and its size is 480x480. The representative array will be 480 x 480 x 3. Each of these numbers is given a value from 0 to 255 which describe the pixel intensity at that point. RGB intensity values of the image are visualized by the computer for processing.

The idea is that you give the computer this array of numbers and it will output numbers that describe the probability of the image being a certain class (.80 for a cat, .15 for a dog, .05 for a bird, etc.). It works similar to how our brain works. When we look at a picture of a dog, we can classify it as such if the picture has identifiable features such as paws or 4 legs. In a similar way, the computer is able to perform image classification by looking for low-level features such as edges and curves and then building up to more abstract concepts through a series of convolutional layers. The computer uses low-level features obtained at the initial levels to generate high-level features such as paws or eyes to identify the object.

Contents of CNN:

- Convolutional Layer
- Activation operation following each convolutional layer
- Pooling layer especially Max Pooling layer and also others based on the requirement Fully Connected Layer

Operations of CNN:

1. Input to a convolution layer - The image is resized to an optimal size and is fed as input to the convolutional layer.

2. There exists a filter or neuron or kernel which lays over some of the pixels of the input image depending on the dimensions of the Kernel size. Let the dimensions of the kernel of the filter be 5x5x3.

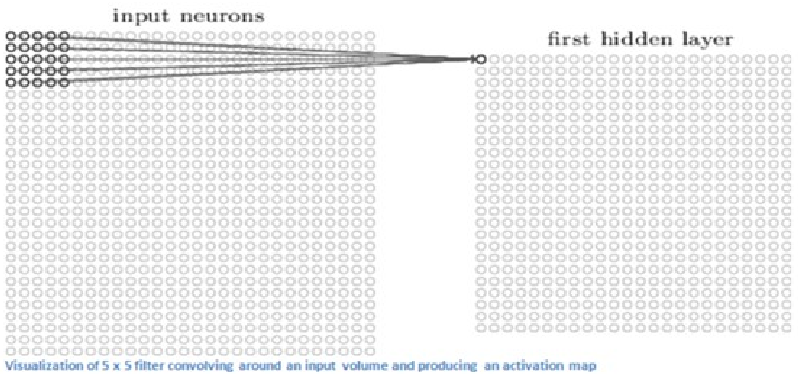
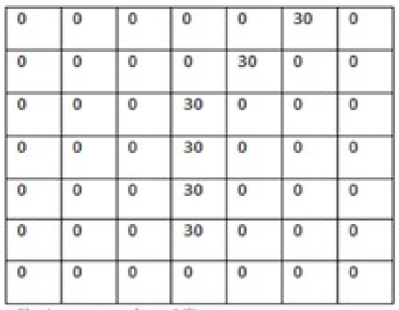

input neurons
first hidden layer
Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

Figure 9: Activation Map

3. The Kernel actually slides over the input image, thus it is multiplying the values in the filter with the original pixel values of the image (aka computing element-wise multiplications). The multiplications are summed up generating a single number for that particular receptive field and hence for sliding the kernel a total of 784 numbers are mapped to 28x28 arrays known as the feature map.

   *Let us take a kernel of size (7x7x3) for understanding. Each of the kernels is considered to be a feature identifier, hence say that our filter will be a curve detector.



0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

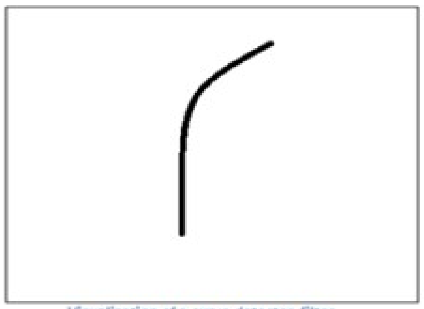
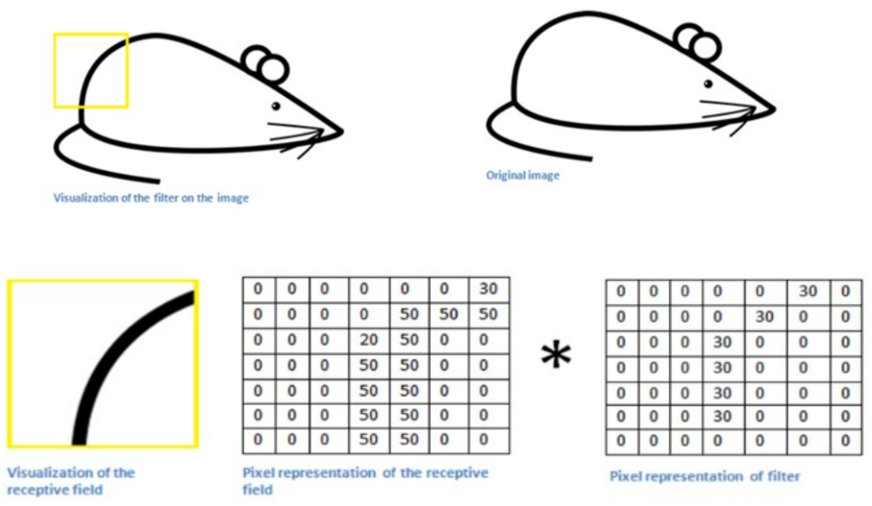


Figure 10: Pixel Representation of the curve

* The original image and the visualization of the kernel on the image.

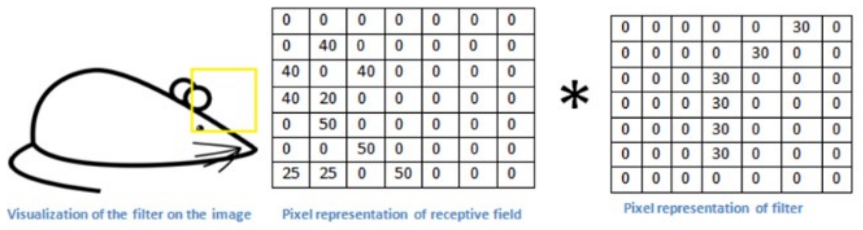

Visualization of the filter on the image
Original image
Visualization of the receptive field
Pixel representation of the receptive field
*
Pixel representation of filter

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Figure 11: Visualization of kernel

* Now when the kernel moves to the other part of the image.


Visualization of the filter on the image
Pixel representation of receptive field
*
Pixel representation of filter

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Figure 12: Visualization of kernel

Sequential Conv layers after the first layer of CNN:

1. When we go through another conv. layer, the output of the first conv. layer becomes the input of the 2nd conv. layer.
2. However, when we're talking about the 2nd conv. layer, the input is the activation map(s) that result from the first layer. So each layer of the input is basically describing the locations in the original image for where certain low-level features appear.
3. Now when you apply a set of filters on top of that (pass it through the 2nd conv. layer), the output will be activations that represent higher-level features. Types of these features could be semicircles (a combination of a curve and straight edge) or squares (a combination of several straight

edges). As you go through the network and go through more conv. layers, you get activation maps that represent more and more complex features.

4. By the end of the network, you may have some filters that activate when there is handwriting in the image, filters that activate when they see pink objects, etc.

Pooling Operation:

A pooling layer is another building block of a CNN. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature map independently. The most common approach used in pooling is max pooling.

Example of max pooling-

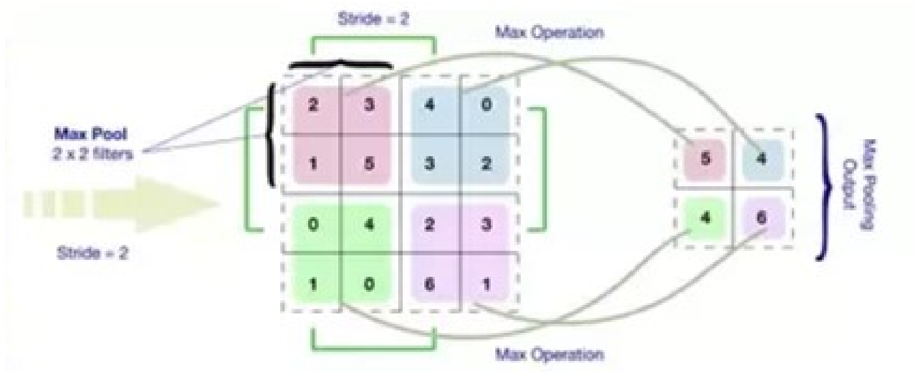


Figure 13: Max Pooling Example

Fully Connected Layer:

1. The way this fully connected layer works is that it looks at the output of the previous layer (which as we remember should represent the activation maps of high-level features) and the number of classes N (10 for digit classification).
2. For example, if the program is predicting that some image is a dog, it will have high values in the activation maps that represent high-level features like a paw or 4 legs, etc. Basically, an FC layer looks at what high level features most strongly correlate to a particular class and has particular weights so that when you compute the products between the weights and the previous layer, you get the correct probabilities for the different classes.
3. The output of a fully connected layer is as follows [0 .1 .1 .75 0 0 0 0 0 .05], then this represents a 10% probability that the image is a 1, a 10% probability

that the image is a 2, a 75% probability that the image is a 3, and a 5% probability that the image is a 9 (Softmax approach) for digit classification.

Training:

The training process is also known as back propagation, which is further separated into 4 distinct sections or processes.

1. Forward Pass
2. Loss Function
3. Backward Pass
4. Weight Update

Forward Pass

For the first epoch or iteration of the training the initial kernels of the first conv. layer is initialized with random values. Thus after the first iteration output will be something like [.1.1.1.1.1.1.1.1.1.1], which does not give preference to any class as the kernels don't have specific weights.

Loss Function

The training involves images along with labels, hence the label for the digit 3 will be [0 0 0 1 0 0 0 0 0 0], whereas the output after a first epoch is very different, hence we will calculate loss (MSE — Mean Squared Error). The objective is to minimize the loss, which is an optimization problem in calculus. It involves trying to adjust the weights to reduce the loss.

Backward Pass

It involves determining which weights contributed most to the loss and finding ways to adjust them so that the loss decreases. It is computed using dL/dW, where L is the loss and the W is the weights of the corresponding kernel.

Weight Update

This is where the weights of the kernel are updated using the following equation.

$$w = w_i - \eta \frac{dL}{dW}$$

w = Weight
w_i = Initial Weight
η = Learning Rate

Here the Learning Rate is chosen by the programmer. Larger value of the learning rate indicates much larger steps towards optimization of steps and larger time to convolve to an optimized weight.

Batch Normalization

In CNN we use Batch normalization. Batch Norm is a normalization technique done between the layers of a Neural Network instead of in the raw data. It is done along mini-batches instead of the full data set. It serves to speed up training and use higher learning rates, making learning easier.
In convolutions, we have shared filters that go along the feature maps of the input. These filters are the same on every feature map. It is then reasonable to normalize the output, in the same way, sharing it over the feature maps.

Testing

Finally, to see whether or not our CNN works, we have a different set of images and labels (can’t double dip between training and test) and pass the images through the CNN. We compare the outputs to the ground truth and see if our network works.

4.2 PROGRAM MODULES SPECIFICATION

There are three modules used in this system. They are the following:

- IMPORTING MODULES
- MODEL CREATION MODULE
- PREDICTION MODULE

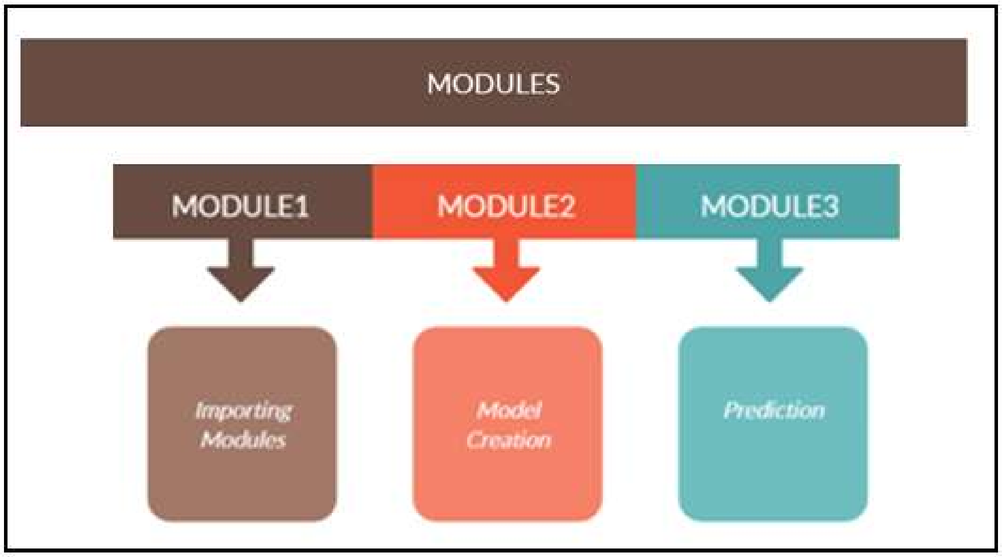


Figure 14: Modules

Importing Modules: Importing all the packages that I am using in this project.

Model Creation: Creating a Model, having feature extractors.

Prediction Module: Predicts the age and gender.

4.3 PROCESS INVOLVED

The involved processes are:-

- Data Collections
- Image Pre-processing
- Extracting Age from images.
- Training the model.
- Validating the model.
- Testing the model.
- Saving the model.
- Prediction of images.

SAMPLE CODE

For Gender Prediction

```
import os
from tensorflow.keras import layers
from tensorflow.keras import Model
from tensorflow.keras.preprocessing.image
import ImageDataGenerator
import tensorflow as tf

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')

test_datagen = ImageDataGenerator( rescale = 1.0/255)
train_generator=train_datagen.flow_from_directory("C:/users/Dataset/Train/",
                                    batch_size =256 ,
                                    class_mode = 'binary',
                                     target_size = (64, 64))

validation_generator=test_datagen.flow_from_directory("C:/user/Dataset/Valid/",
                                     batch_size  = 256,
                                     class_mode  = 'binary'
                                     target_size = (64, 64))

from keras.optimizers import Adam
model = tf.keras.models.Sequential([
# 1st conv
tf.keras.layers.Conv2D(96, (11,11),strides=(4,4), activation='relu',input_shape=(64,
64, 3)),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.MaxPooling2D(2, strides=(2,2)),
# 2nd conv
tf.keras.layers.Conv2D(256, (11,11),strides=(1,1), activation='relu',padding="same"),
tf.keras.layers.BatchNormalization(),
# 3rd conv
tf.keras.layers.Conv2D(384, (3,3),strides=(1,1), activation='relu',padding="same"),
tf.keras.layers.BatchNormalization(),
# 4th conv tf.keras.layers.Conv2D(384, (3,3),strides=(1,1),
activation='relu',padding="same"),
 tf.keras.layers.BatchNormalization(),
# 5th Conv tf.keras.layers.Conv2D(256, (3, 3), strides=(1, 1),
activation='relu',padding="same"),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.MaxPooling2D(2, strides=(2, 2)),
# To Flatten layer
 tf.keras.layers.Flatten(),
# To FC layer 1
tf.keras.layers.Dense(4096, activation='relu'),
tf.keras.layers.Dropout(0.5),
#To FC layer 2
tf.keras.layers.Dense(4096, activation='relu'),
tf.keras.layers.Dropout(0.5),
tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile( optimizer=Adam(lr=0.001),
loss='binary_crossentropy',
metrics=['accuracy'] )

 hist = model.fit_generator(generator=train_generator,
validation_data=validation_generator,
steps_per_epoch=256,
validation_steps=256,
epochs=50)

from keras.models import model_from_json
model_json = model.to_json()
with open("model.json", "w") as json_file: json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model.h5")
```

For Age Prediction

```
!pip install ktrain

%matplotlib inline
import os
import ktrain
from ktrain import vision as vs

import re
pattern1=r'([^/]+)_\d+_\d+_\d+.jpg$'
p=re.compile(pattern1)
r=p.search('15_0_0_201770112020.jpg')
print(r.group(1))

DATADIR='/content/Image-Dataset-for-Age-Prediction/images'

data_aug = vs.get_data_aug(horizontal_flip=True)

(train_data, val_data, preproc) = vis.images_from_fname(DATADIR,
                                                         pattern = pattern1,
                                                        data_aug = data_aug,
                                                         is_regression=True,
                                                         random_state=42)

vs.print_image_regression_models()
    pretrained_resnet50: 50-layer Residual Network (pretrained)
    resnet50: 50-layer Resididual Network (randomly initialized)
```

```
model = vs.image_regression_model('pretrained_resnet50', train_data, val_data)

learner=ktrain.get_learner(model=model, train_data=train_data,
                                                         val_data=val_data,
                                                          batch_size=64)


#learning rate is 1e-4
#epoch is 2
learner.fit_onecycle(1e-4, 25)


learner.freeze(15)
learner.fit_onecycle(1e-4, 17)


predictor=ktrain.get_predictor(learner.model,preproc)


def show_prediction(fname):

        fname=DATADIR +'/' + fname

        pred=round(predictor.predict_filename(fname)[0])

        actual=int(p.search(fname).group(1))

        vs.show_image(fname)

        print("Predicted age :%s | Actual age :%s" %(pred,actual))


predictor.save('content/age_prediction')


!zip -r /content/age_prediction.zip /content/content/age_prediction


model_json = learner.model.to_json()
with open("model2.json", "w") as json_file: json_file.write(model_json)
learner.model.save_weights("model2.h5")


def real_prediction(fname):

        pred=round(predictor.predict_filename(fname)[0])

        vs.show_image(fname)

        print("Predicted age :%s " %(pred))
```

DJANGO

Forms.py

```
from PIL import Image
from django import forms
from django.core.files.uploadedfile import SimpleUploadedFile
from .models import *
class ImageUploadForm(forms.ModelForm):
    class Meta:
        model= ExampleModel
        fields=['image']
```

Models.py

```
from django.db import models
class ExampleModel(models.Model):
 image = models.ImageField(upload_to="images/")
```

Urls.py

```
from django.contrib import admin
from django.urls import path,include
from .import views
urlpatterns = [
    path('',views.home,name='home'),
    path('imageprocess',views.imageprocess,name='imageprocess'),]
```

Views.py

```
from django.shortcuts import render
from django import forms
from .forms import ImageUploadForm
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input,
decode_predictions
import numpy as np
from keras.models import model_from_json
import json
from django.http import HttpResponse
import ktrain
from ktrain import vision as vs
from .models import *
from django.views.generic import TemplateView


def handle_uploaded_file(f):
        with open('media/img.jpg', 'wb+') as destination:
         for chunk in f.chunks():
                destination.write(chunk)


def home(request):
        return render(request,'home.html')

def imageprocess(request):
        if request.method == 'POST':
                form=ImageUploadForm(request.POST,request.FILES)
                if form.is_valid():
                        handle_uploaded_file(request.FILES['image'])
                        image_field = form.cleaned_data['image']
                        form.save()


 json_file=open("./models/model.json","r")
loaded_json=json_file.read()
 json_file.close()
```

```
    model=model_from_json(loaded_json)
    model.load_weights("./models/model.h5")
    MODEL=model
    reloaded_predictor = ktrain.load_predictor('./models/age_prediction')

    img_path='media/img.jpg'
    img = image.load_img(img_path, target_size=(64, 64))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    images = np.vstack([x])
    classes = model.predict(images, batch_size=1)
    print(classes[0])
    z=reloaded_predictor.predict_filename(img_path)
    if classes[0]>0.5:
        res1="Male"
        res2=int(z)
        res3=res2-(res2%5)
        res2=res3+5
        return render(request,'result.html',{'res1':res1,'res2':res2,'res3':res3})
    else:
        res1="Female"
        res2=int(z)
        res3=res2-(res2%5)
        res2=res3+5
        return render(request,'result.html',{'res1':res1,'res2':res2,'res3':res3})
else:
    form = ImageUploadForm()
img=ExampleModel.objects.all()
```

Admin urls.py

```
from django.contrib import admin
from django.conf import settings
from django.conf.urls.static import static
from django.urls import include, path
urlpatterns = [
            path('', include('agegender.urls')),
            path('admin/', admin.site.urls),
            ]
if settings.DEBUG:
      urlpatterns += static(settings.MEDIA_URL,
                                    document_root=settings.MEDIA_ROOT)
```

Admin settings.py

```
import os
from pathlib import Path
from keras.models import load_model
DEBUG = True
ALLOWED_HOSTS = ['*']
INSTALLED_APPS = [
       'django.contrib.admin',
      'django.contrib.auth',
       'django.contrib.contenttypes',
       'django.contrib.sessions',
      'django.contrib.messages',
      'django.contrib.staticfiles',
       'agegender',
      ]
MIDDLEWARE = [
       'django.middleware.security.SecurityMiddleware',
            'django.contrib.sessions.middleware.SessionMiddleware',
            'django.middleware.common.CommonMiddleware',
            'django.middleware.csrf.CsrfViewMiddleware',
             'django.contrib.auth.middleware.AuthenticationMiddleware',
             'django.contrib.messages.middleware.MessageMiddleware',
             'django.middleware.clickjacking.XFrameOptionsMiddleware',
          ]
ROOT_URLCONF = 'my.urls'
TEMPLATES = [{
            'BACKEND': 'django.template.backends.django.DjangoTemplates',
           'DIRS': ['templates'],
           'APP_DIRS': True,
            'OPTIONS': {
                       'context_processors': [
                       'django.template.context_processors.debug',
                       'django.template.context_processors.request',
                      'django.contrib.auth.context_processors.auth',
                       'django.contrib.messages.context_processors.messages',
                       ],
                 },
            },
         ]

WSGI_APPLICATION = 'my.wsgi.application'
DATABASES = { 'default': {
                       'ENGINE': 'django.db.backends.sqlite3',
                      'NAME': BASE_DIR / 'db.sqlite3',
                      }
                  }
LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True
STATIC_URL = '/static/'
STATICFILES_DIRS = (
        os.path.join(BASE_DIR, 'static'),
        )
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

Result Page

```
{% extends 'home.html' %}
 {% block content %}
{% load static %}
<div class="text-center ">
 <h3>Predictions</h3>
</div>
<div class="text-center">
<form class="cta-btn">
<img src="media/img.jpg" alt="image" height=20% width=20%/>
<h2>Gender:- {{res1}}</h2>
 <h2>Age:- [{{res3}} - {{res2}}]</h2>
</form>
 </div>
 <div class="text-center">
 <a class="cta-btn text-center" href="./#cta">Submit another Image</a>
 </div>
{% endblock content %}
```

TESTING

6.1 INTRODUCTION

System testing is the implementation stage to ensure that the system works accurately and efficiently before live operation begins. Testing is vital to the fulfillment of the system. System testing makes logical assumptions that if all the system parts are correct, the system will successfully achieve it. A series of testing is performed for the proposed plan before the system is ready for user acceptance testing.

The following are the types of testing:

- Unit Testing
- Integration Testing
- Validation Testing

6.2 TEST CASES AND TEST SCENARIOS

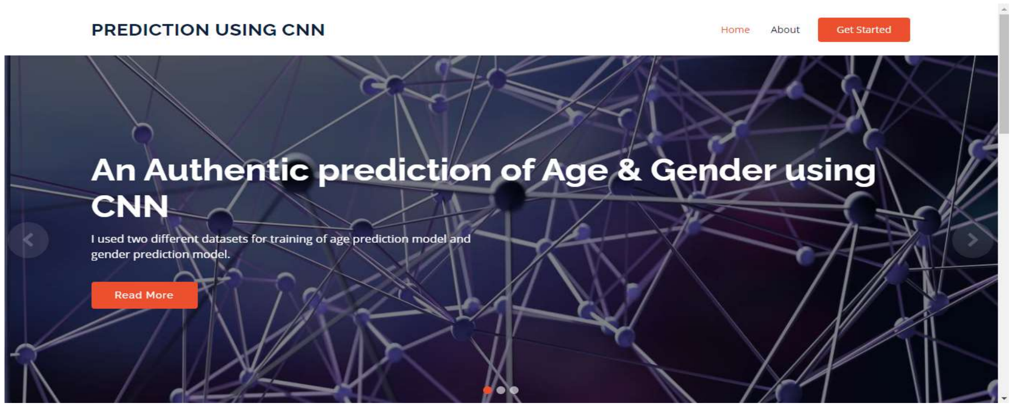
Test Case ID	ML_01	**Test Case Description**	Test the prediction of age & gender
Created By	M.L.Sai Kumar	**Reviewed By**	Ms. K. Yasudha
Tester's Name	M.L.Sai Kumar	**Date Tested**	April 21, 2021

S #	**Prerequisites:**	**S #**	**Test Data**	**Test Case (Pass/Fail/Not Executed)**
1	Access to Browser	1	Images	Pass
2	Images to test			

Test Scenario	Verify on selecting the Images and predicting the output

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Navigate to Site	Site should open	As Expected	Pass
2	Images should be selected	Images can be selected	As Expected	Pass
3	Click Predict button	Navigated to next page	As Expected	Pass
4	Show Result	Result is displayed	As Expected	Pass
5	Go back to upload new image	Home page should be redirected	As Expected	Pass

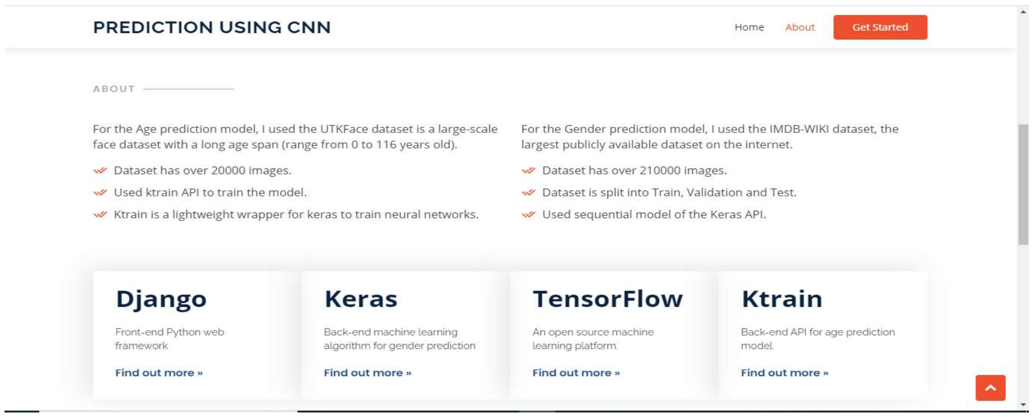
SCREENSHOTS



An Authentic prediction of Age & Gender using CNN

I used two different datasets for training of age prediction model and gender prediction model.

Figure 15: Home Page



For the Age prediction model, I used the UTKFace dataset is a large-scale face dataset with a long age span (range from 0 to 116 years old).

- Dataset has over 20000 images.
- Used ktrain API to train the model.
- Ktrain is a lightweight wrapper for keras to train neural networks.

For the Gender prediction model, I used the IMDB-WIKI dataset, the largest publicly available dataset on the internet.

- Dataset has over 210000 images.
- Dataset is split into Train, Validation and Test.
- Used sequential model of the Keras API.

Django — Front-end Python web framework — Find out more »

Keras — Back-end machine learning algorithm for gender prediction — Find out more »

TensorFlow — An open source machine learning platform. — Find out more »

Ktrain — Back-end API for age prediction model — Find out more »

Figure 16: About Page

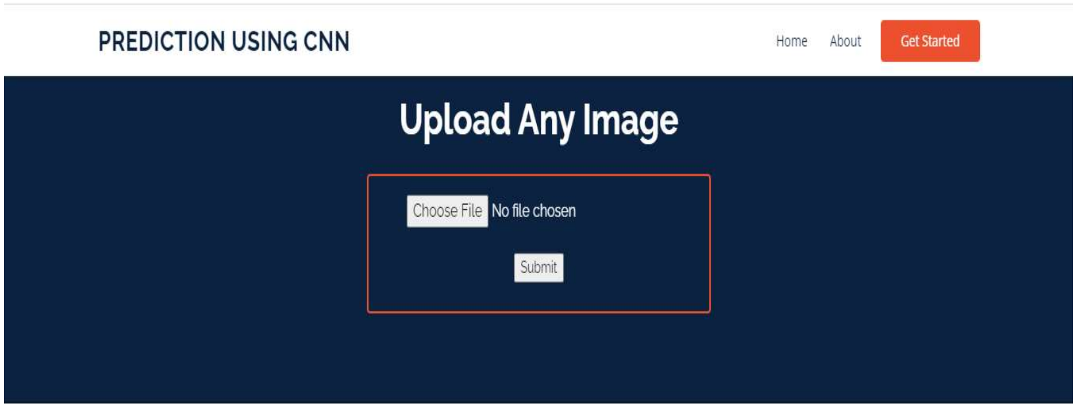


Figure 17: Upload Image Page

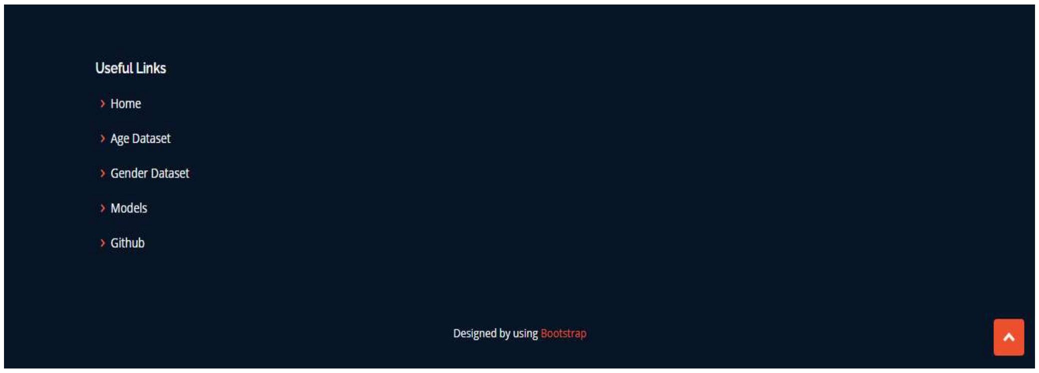


Figure 18: Footer

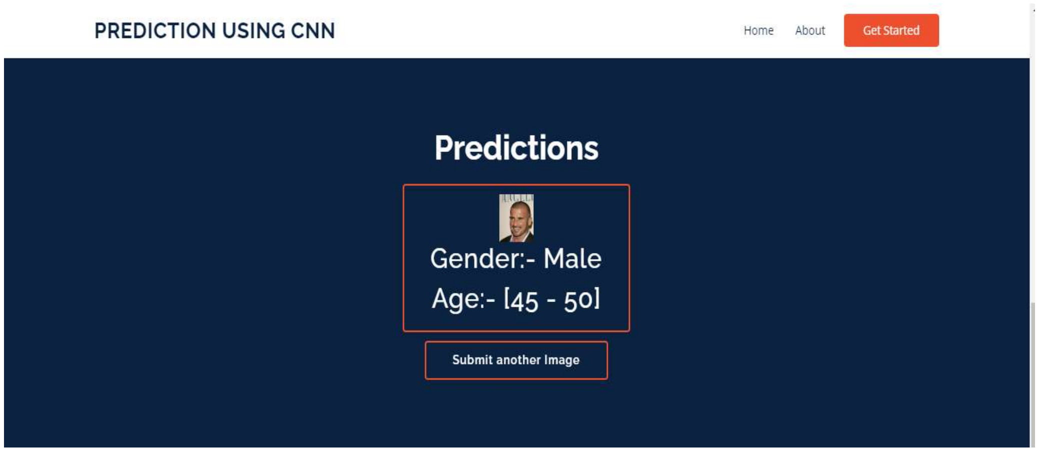


Figure 191: Output1

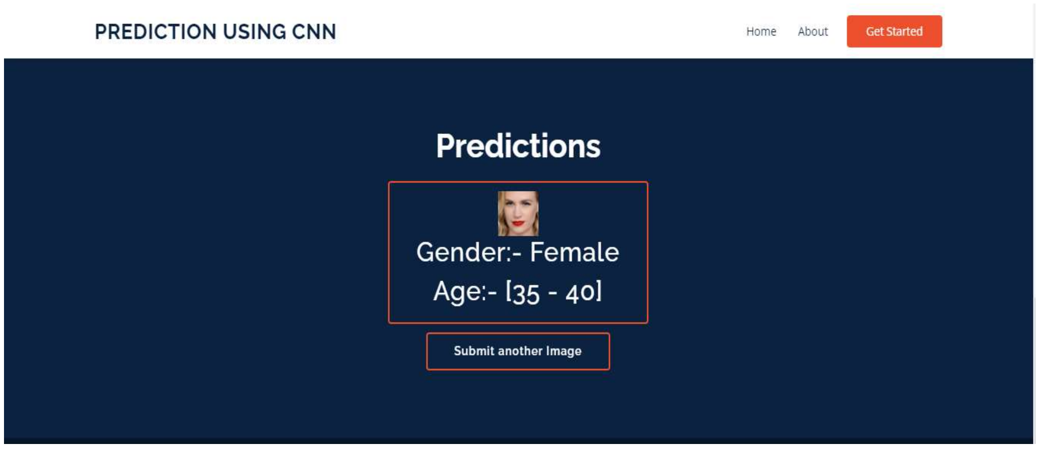
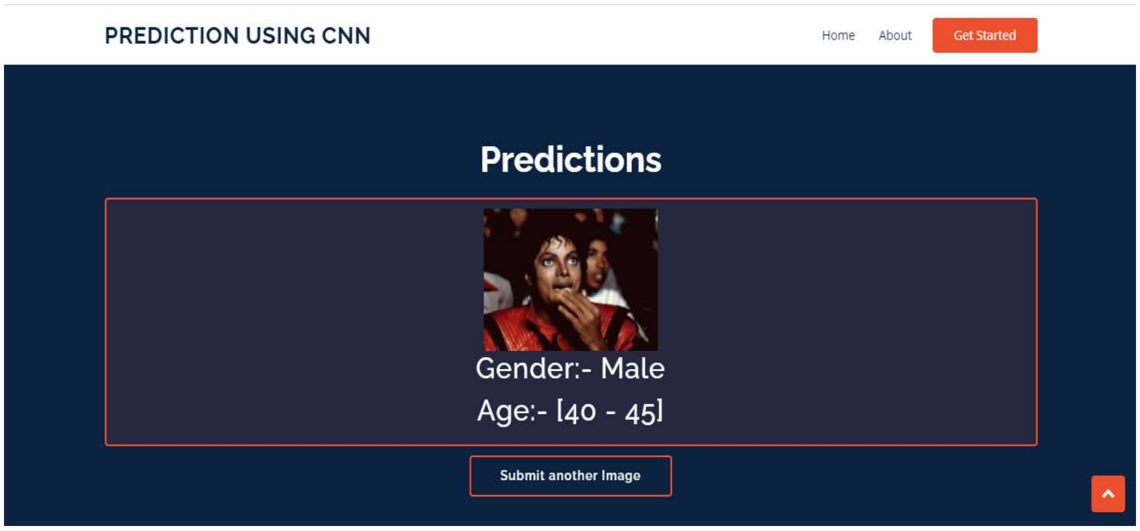


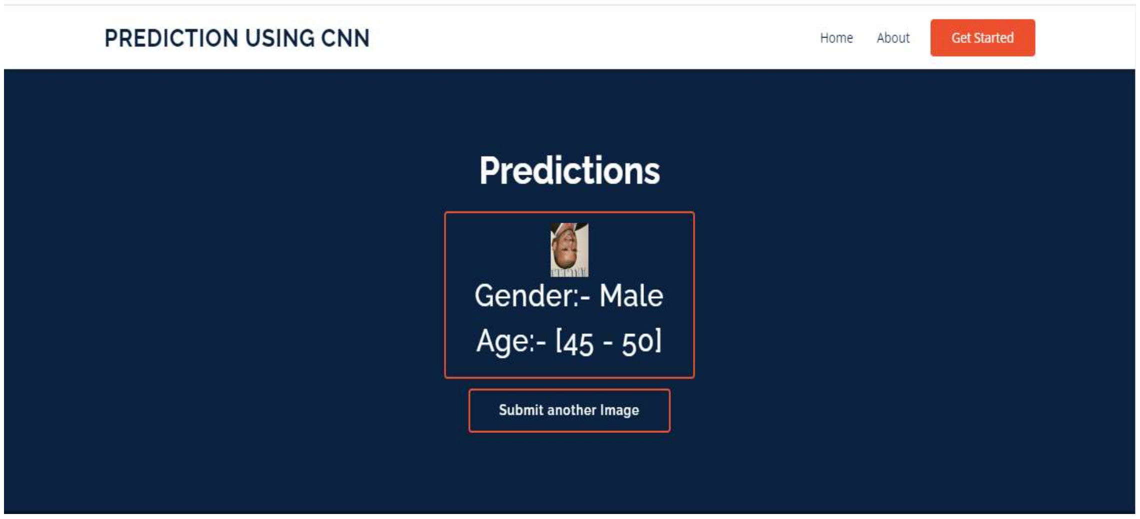
Figure20: Output2



Gender:- Male

Age:- [40 - 45]

Figure21: Output3



Gender:- Male

Age:- [45 - 50]

Figure22: Output4

CONCLUSION

Though many previous methods have addressed the problems of age and gender classification, until recently, much of this work has focused on constrained images taken in lab settings. Such settings do not adequately reflect appearance variations common to the real-world images in social websites and online repositories. Internet images, however, are not simply more challenging: they are also abundant. The easy availability of huge image collections provides modern machine learning based systems with effectively endless training data, though this data is not always suitably labelled for supervised learning.

Two important conclusions can be made from our results. First, CNN can be used to provide improved age and gender classification results, even considering the much smaller size of contemporary unconstrained image sets labelled for age and gender. Second, the simplicity of our model implies that more elaborate systems using more training data may well be capable of substantially improving results beyond those reported in this project.

BIBLIOGRAPHY

- http://keras.io/
- https://www.tensorflow.org/guide/keras/sequential_model
- https://github.com/amaiya/ktrain
- http://docs.djangoproject.com/en/3.2/
- https://colab.research.google.com/
- https://code.visualstudio.com/
- https://aws.amazon.com/getting-started/hands-on/deploy-python-application/
- https://www.vitoshacademy.com/python-django-how-to-add-css-to-django-application/
- https://vegibit.com/django-static-files-and-images/
- https://www.geeksforgeeks.org/how-to-deploy-django-application-on-heroku/