**Age and Gender Classiﬁcation using Convolutional Neural Networks**

**1. INTRODUCTION**

Age and gender play fundamental roles in social interactions. Languages reserve different salutations and grammar rules for men or women, and very often different vocabularies are used when addressing elders compared to young people. Despite the basic roles these attributes play in our day-to-day lives, the ability to automatically estimate them accurately and reliably from face images is still far from meeting the needs of commercial applications. This is particularly perplexing when considering recent claims to super-human capabilities in the related task of face recognition . Past approaches to estimating or classifying these attributes from face images have relied on differences in facial feature dimensions or “tailored” face descriptors. Most have employed classiﬁcation schemes designed particularly for age or gender estimation tasks, including and others. Few of these past methods were designed to handle the many challenges of unconstrained imaging conditions. Moreover, the machine learning methods employed by these systems did not fully exploit the massive numbers of image examples and data available through the Internet in order to improve classiﬁcation capabilities. In this paper we attempt to close the gap between automatic face recognition capabilities and those of age and gender estimation methods. To this end, we follow the successful example laid down by recent face recognition systems: Face recognition techniques described in the last few years have shown that tremendous progress can be made by the use of deep convolutional neural networks (CNN). We demonstrate similar gains with a simple network architecture, designed by considering the rather limited availability of accurate age and gender labels in existing face data sets. We test our network on the newly released Adience benchmark for age and gender classiﬁcation of unﬁltered face images. We show that despite the very challenging nature of the images in the Adience set and the simplicity of our network design, our method outperforms existing state of the art by substantial margins. Although these results provide a remarkable baseline for deep-learning-based approaches, they leave room for improvements by more elaborate system designs, suggesting that the problem of accurately estimating age and gender in the unconstrained settings, as reﬂected by the Adience images, remains unsolved. In order to provide a foothold for the development of more effective future methods, we make our trained models and classiﬁcation system publicly available.

**1.1 Objective of the Project**

Age and gender play fundamental roles in social interactions. Languages reserve different salutations and grammar rules for men or women, and very often different vocabularies are used when addressing elders compared to young people. In this paper we show that by learning representations through the use of deep-convolutional neural networks (CNN), a signiﬁcant increase in performance can be obtained on these tasks. To this end, we propose a simple convolutional net architecture that can be used even when the amount of learning data is limited. We evaluate our method on the recent Adience benchmark for age and gender estimation and show it to dramatically outperform current state-of-the-art methods.

**2. LITERATURE SURVEY**

**Face description with local binary patterns: Application to face recognition.**

This paper presents a novel and efficient facial image representation based on local binary pattern (LBP) texture features. The face image is divided into several regions from which the LBP feature distributions are extracted and concatenated into an enhanced feature vector to be used as a face descriptor. The performance of the proposed method is assessed in the face recognition problem under different challenges. Other applications and several extensions are also discussed.

**Boosting sex identiﬁcation performance**

This paper presents a method based on AdaBoost to identify the sex of a person from a low resolution grayscale picture of their face. The method described here is implemented in a system that will process well over 109 images. The goal of this work is to create an efficient system that is both simple to implement and maintain; the methods described here are extremely fast and have straightforward implementations. We achieve 80% accuracy in sex identification with less than 10 pixel comparisons and 90% accuracy with less than 50 pixel comparisons. The best classifiers published to date use Support Vector Machines; we match their accuracies with as few as 500 comparison operations on a 20× 20 pixel image. The AdaBoost based classifiers presented here achieve over 93% accuracy; these match or surpass the accuracies of the SVM-based classifiers, and yield performance that is 50 times faster.

**Learning distance functions using equivalence relations**

We address the problem of learning distance metrics using side-information in the form of groups of "similar" points. We propose to use the RCA algorithm, which is a simple and efficient algorithm for learning a full ranked Mahalanobis metric (Shental et al., 2002). We first show that RCA obtains the solution to an interesting optimization problem, founded on an information theoretic basis. If the Mahalanobis matrix is allowed to be singular, we show that Fisher’s linear discriminant followed by RCA is the optimal dimensionality reduction algorithm under the same criterion. We then show how this optimization problem is related to the criterion optimized by another recent algorithm for metric learning (Xing et al., 2002), which uses the same kind of side information. We empirically demonstrate that learning a distance metric using the RCA algorithm significantly improves clustering performance, similarly to the alternative algorithm. Since the RCA algorithm is much more efficient and cost effective than the alternative, as it only uses closed form expressions of the data, it seems like a preferable choice for the learning of full rank Mahalanobis distances.

**Facial age estimation based on label-sensitive learning and age-oriented regression.**

In this paper, a new age estimation framework considering the intrinsic properties of human ages is proposed, which improves the dimensionality reduction techniques to learn the connections between facial features and aging labels. To enhance the performance of dimensionality reduction, a distance metric adjustment step is introduced in advance to achieve a suitable metric in the feature space. In addition, to further exploit the ordinal relationship of human ages, the “label-sensitive” concept is proposed, which regards the label similarity during the learning phase of distance metric and dimensionality reduction. Finally, an age-specific local regression algorithm is proposed to capture the complicated aging process for age determination. From the simulation results, the proposed framework achieves the lowest mean absolute error against the existing methods.

**Human age estimation with regression on discriminative aging manifold.**

Recently, extensive studies on human faces in the human-computer interaction (HCI) field reveal significant potentials for designing automatic age estimation systems via face image analysis. The success of such research may bring in many innovative HCI tools used for the applications of human-centered multimedia communication. Due to the temporal property of age progression, face images with aging features may display some sequential patterns with low-dimensional distributions. In this paper, we demonstrate that such aging patterns can be effectively extracted from a discriminant subspace learning algorithm and visualized as distinct manifold structures. Through the manifold method of analysis on face images, the dimensionality redundancy of the original image space can be significantly reduced with subspace learning. A multiple linear regression procedure, especially with a quadratic model function, can be facilitated by the low dimensionality to represent the manifold space embodying the discriminative property. Such a processing has been evaluated by extensive simulations and compared with the state-of-the-art methods. Experimental results on a large size aging database demonstrate the effectiveness and robustness of our proposed framework.

**Face Age Classification on Consumer Images with Gabor Feature and Fuzzy LDA Method**

As we all know, face age estimation task is not only challenging for computer, but even hard for human in some cases, however, coarse age classification such as classifying human face as baby, child, adult or elder people is much easier for human. In this paper, we try to dig out the potential age classification power of computer on faces from consumer images which are taken under various conditions. Gabor feature is extracted and used in LDA classifiers. In order to solve the intrinsic age ambiguity problem, a fuzzy version LDA is introduced through defining age membership functions. Systematic comparative experiment results show that the proposed method with Gabor feature and fuzzy LDA can achieve better age classification precision in consumer images.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

In existing systems, they process only low level images with limited accuracy. The performance of existing methods on real-world images is still signiﬁcantly lacking, especially when compared to the tremendous leaps in performance recently reported for the related task of face recognition.

.

**Disadvantages**:

1. Accuracy is less.

2. They process only low level images.

3. Architecture is complex to understand.

**3.2 Proposed System**

In this paper, we proposed a age and gender predicted framwork by using deep-convolutional neural networks (CNN). 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 labeled 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 here.

**Advantages**:

1. Accuracy is high.

2. Simple architecture.

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms *what* must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**Operating Environment**

Windows XP.

**HARDWARE REQUIREMENTS:**

# Processor - Pentium –IV

* Speed - 1.1 Ghz
* RAM - 256 MB(min)
* Hard Disk - 20 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows7/8
* Programming Language - Python (python 3.6.3)

**4. SYSTEM DESIGN**

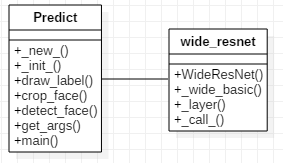
**UML Diagram:**

**Class Diagram:**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

**Class Diagram:**

****

**Use case Diagram:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

**Use case Diagram:**

Start Webcam

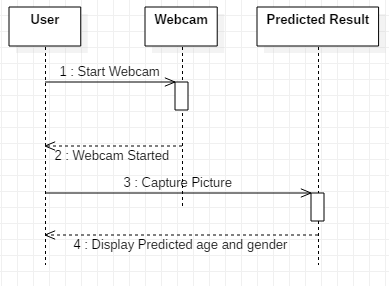
Get Predicted age and gender results

Exit

User

**Sequence diagram:**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.



**Collaboration diagram:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

User

Webcam,

Predicted

Results

1: Start Webcam

2: Webcam Started

3: Image Captured

4: Predict Age and Gender Results displyed

**Component Diagram:**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

User

Start Webcam

Get predicted

Age and gender

Exit

**Deployment Diagram:**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

User

Start webcam

Get age and

gender

Exit

**Activity Diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent

User

Start Webcam

Capture Image

Get Predicted Age and Gender Results

**Data Flow Diagram:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

User

2. webcam started

1. Start webcam 4. Predicted age and gender are displayed

3. Get Predicted results

5. Exit

**5. IMPLEMETATION**

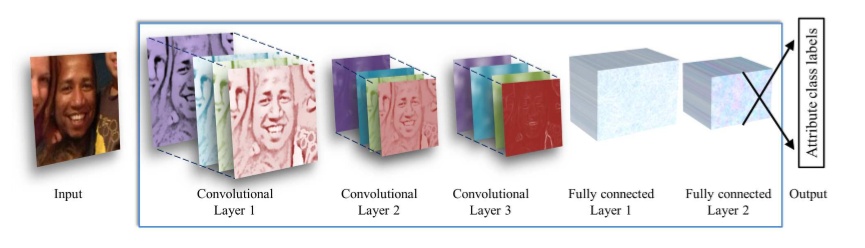


Illustration of our CNN architecture

**5.1 Python**

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

**History of Python:**

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

**Why Python was created?**

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

**Why the name Python?**

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

**Features of Python:**

**A simple language which is easier to learn**

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

If you are a newbie, it's a great choice to start your journey with Python.

**Free and open-source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute softwares written in it, you can even make changes to the Python's source code.

Python has a large community constantly improving it in each iteration.

**Portability**

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

**Extensible and Embeddable**

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

**A high-level, interpreted language**

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

**Large standard libraries to solve common tasks**

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

**Object-oriented**

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

**Applications of Python:**

**1. Simple Elegant Syntax**

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

**2. Not overly strict**

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

**3. Expressiveness of the language**

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

**4. Great Community and Support**

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

**5.2 Sample Code;**

**Predict.py:**

import cv2

import os

from time import sleep

import numpy as np

import argparse

from wide\_resnet import WideResNet

from keras.utils.data\_utils import get\_file

class FaceCV(object):

CASE\_PATH = ".\\models\\haarcascade\_frontalface\_alt.xml"

WRN\_WEIGHTS\_PATH = ".\\models\\weights.18-4.06.hdf5"

def \_\_new\_\_(cls, weight\_file=None, depth=16, width=8, face\_size=64):

if not hasattr(cls, 'instance'):

cls.instance = super(FaceCV, cls).\_\_new\_\_(cls)

return cls.instance

def \_\_init\_\_(self, depth=16, width=8, face\_size=64):

self.face\_size = face\_size

self.model = WideResNet(face\_size, depth=depth, k=width)()

model\_dir = os.path.join(os.getcwd(), "models").replace("//", "\\")

fpath = get\_file('weights.18-4.06.hdf5',

self.WRN\_WEIGHTS\_PATH,

cache\_subdir=model\_dir)

self.model.load\_weights(fpath)

@classmethod

def draw\_label(cls, image, point, label, font=cv2.FONT\_HERSHEY\_SIMPLEX,

font\_scale=1, thickness=2):

size = cv2.getTextSize(label, font, font\_scale, thickness)[0]

x, y = point

cv2.rectangle(image, (x, y - size[1]), (x + size[0], y), (255, 0, 0), cv2.FILLED)

cv2.putText(image, label, point, font, font\_scale, (255, 255, 255), thickness)

def crop\_face(self, imgarray, section, margin=40, size=64):

"""

:param imgarray: full image

:param section: face detected area (x, y, w, h)

:param margin: add some margin to the face detected area to include a full head

:param size: the result image resolution with be (size x size)

:return: resized image in numpy array with shape (size x size x 3)

"""

img\_h, img\_w, \_ = imgarray.shape

if section is None:

section = [0, 0, img\_w, img\_h]

(x, y, w, h) = section

margin = int(min(w,h) \* margin / 100)

x\_a = x - margin

y\_a = y - margin

x\_b = x + w + margin

y\_b = y + h + margin

if x\_a < 0:

x\_b = min(x\_b - x\_a, img\_w-1)

x\_a = 0

if y\_a < 0:

y\_b = min(y\_b - y\_a, img\_h-1)

y\_a = 0

if x\_b > img\_w:

x\_a = max(x\_a - (x\_b - img\_w), 0)

x\_b = img\_w

if y\_b > img\_h:

y\_a = max(y\_a - (y\_b - img\_h), 0)

y\_b = img\_h

cropped = imgarray[y\_a: y\_b, x\_a: x\_b]

resized\_img = cv2.resize(cropped, (size, size), interpolation=cv2.INTER\_AREA)

resized\_img = np.array(resized\_img)

return resized\_img, (x\_a, y\_a, x\_b - x\_a, y\_b - y\_a)

def detect\_face(self):

face\_cascade = cv2.CascadeClassifier(self.CASE\_PATH)

# 0 means the default video capture device in OS

video\_capture = cv2.VideoCapture(0)

# infinite loop, break by key ESC

while True:

if not video\_capture.isOpened():

sleep(5)

# Capture frame-by-frame

ret, frame = video\_capture.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(

gray,

scaleFactor=1.2,

minNeighbors=10,

minSize=(self.face\_size, self.face\_size)

)

# placeholder for cropped faces

face\_imgs = np.empty((len(faces), self.face\_size, self.face\_size, 3))

for i, face in enumerate(faces):

face\_img, cropped = self.crop\_face(frame, face, margin=40, size=self.face\_size)

(x, y, w, h) = cropped

cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 200, 0), 2)

face\_imgs[i,:,:,:] = face\_img

if len(face\_imgs) > 0:

# predict ages and genders of the detected faces

results = self.model.predict(face\_imgs)

predicted\_genders = results[0]

ages = np.arange(0, 101).reshape(101, 1)

predicted\_ages = results[1].dot(ages).flatten()

# draw results

for i, face in enumerate(faces):

label = "{}, {}".format(int(predicted\_ages[i]),

"F" if predicted\_genders[i][0] > 0.5 else "M")

self.draw\_label(frame, (face[0], face[1]), label)

cv2.imshow('Predicted Faces', frame)

if cv2.waitKey(5) == 27: # ESC key press

break

# When everything is done, release the capture

video\_capture.release()

cv2.destroyAllWindows()

def get\_args():

parser = argparse.ArgumentParser(description=""

"estimates age and gender for the detected faces.",

formatter\_class=argparse.ArgumentDefaultsHelpFormatter)

parser.add\_argument("--depth", type=int, default=16,

help="depth of network")

parser.add\_argument("--width", type=int, default=8,

help="width of network")

args = parser.parse\_args()

return args

def main():

args = get\_args()

depth = args.depth

width = args.width

face = FaceCV(depth=depth, width=width)

face.detect\_face()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**wide\_resnet.py:**

# This code is imported from the following project: https://github.com/asmith26/wide\_resnets\_keras

import logging

import sys

import numpy as np

from keras.models import Model

from keras.layers import Input, Activation, add, Dense, Flatten, Dropout

from keras.layers.convolutional import Conv2D, AveragePooling2D

from keras.layers.normalization import BatchNormalization

from keras.regularizers import l2

from keras import backend as K

sys.setrecursionlimit(2 \*\* 20)

np.random.seed(2 \*\* 10)

class WideResNet:

def \_\_init\_\_(self, image\_size, depth=16, k=8):

self.\_depth = depth

self.\_k = k

self.\_dropout\_probability = 0

self.\_weight\_decay = 0.0005

self.\_use\_bias = False

self.\_weight\_init = "he\_normal"

if K.image\_data\_format() == "th":

logging.debug("image\_dim\_ordering = 'th'")

self.\_channel\_axis = 1

self.\_input\_shape = (3, image\_size, image\_size)

else:

logging.debug("image\_dim\_ordering = 'tf'")

self.\_channel\_axis = -1

self.\_input\_shape = (image\_size, image\_size, 3)

# Wide residual network http://arxiv.org/abs/1605.07146

def \_wide\_basic(self, n\_input\_plane, n\_output\_plane, stride):

def f(net):

# format of conv\_params:

# [ [kernel\_size=("kernel width", "kernel height"),

# strides="(stride\_vertical,stride\_horizontal)",

# padding="same" or "valid"] ]

# B(3,3): orignal <<basic>> block

conv\_params = [[3, 3, stride, "same"],

[3, 3, (1, 1), "same"]]

n\_bottleneck\_plane = n\_output\_plane

# Residual block

for i, v in enumerate(conv\_params):

if i == 0:

if n\_input\_plane != n\_output\_plane:

net = BatchNormalization(axis=self.\_channel\_axis)(net)

net = Activation("relu")(net)

convs = net

else:

convs = BatchNormalization(axis=self.\_channel\_axis)(net)

convs = Activation("relu")(convs)

convs = Conv2D(n\_bottleneck\_plane, kernel\_size=(v[0], v[1]),

strides=v[2],

padding=v[3],

kernel\_initializer=self.\_weight\_init,

kernel\_regularizer=l2(self.\_weight\_decay),

use\_bias=self.\_use\_bias)(convs)

else:

convs = BatchNormalization(axis=self.\_channel\_axis)(convs)

convs = Activation("relu")(convs)

if self.\_dropout\_probability > 0:

convs = Dropout(self.\_dropout\_probability)(convs)

convs = Conv2D(n\_bottleneck\_plane, kernel\_size=(v[0], v[1]),

strides=v[2],

padding=v[3],

kernel\_initializer=self.\_weight\_init,

kernel\_regularizer=l2(self.\_weight\_decay),

use\_bias=self.\_use\_bias)(convs)

# Shortcut Connection: identity function or 1x1 convolutional

# (depends on difference between input & output shape - this

# corresponds to whether we are using the first block in each

# group; see \_layer() ).

if n\_input\_plane != n\_output\_plane:

shortcut = Conv2D(n\_output\_plane, kernel\_size=(1, 1),

strides=stride,

padding="same",

kernel\_initializer=self.\_weight\_init,

kernel\_regularizer=l2(self.\_weight\_decay),

use\_bias=self.\_use\_bias)(net)

else:

shortcut = net

return add([convs, shortcut])

return f

# "Stacking Residual Units on the same stage"

def \_layer(self, block, n\_input\_plane, n\_output\_plane, count, stride):

def f(net):

net = block(n\_input\_plane, n\_output\_plane, stride)(net)

for i in range(2, int(count + 1)):

net = block(n\_output\_plane, n\_output\_plane, stride=(1, 1))(net)

return net

return f

# def create\_model(self):

def \_\_call\_\_(self):

logging.debug("Creating model...")

assert ((self.\_depth - 4) % 6 == 0)

n = (self.\_depth - 4) / 6

inputs = Input(shape=self.\_input\_shape)

n\_stages = [16, 16 \* self.\_k, 32 \* self.\_k, 64 \* self.\_k]

conv1 = Conv2D(filters=n\_stages[0], kernel\_size=(3, 3),

strides=(1, 1),

padding="same",

kernel\_initializer=self.\_weight\_init,

kernel\_regularizer=l2(self.\_weight\_decay),

use\_bias=self.\_use\_bias)(inputs) # "One conv at the beginning (spatial size: 32x32)"

# Add wide residual blocks

block\_fn = self.\_wide\_basic

conv2 = self.\_layer(block\_fn, n\_input\_plane=n\_stages[0], n\_output\_plane=n\_stages[1], count=n, stride=(1, 1))(conv1)

conv3 = self.\_layer(block\_fn, n\_input\_plane=n\_stages[1], n\_output\_plane=n\_stages[2], count=n, stride=(2, 2))(conv2)

conv4 = self.\_layer(block\_fn, n\_input\_plane=n\_stages[2], n\_output\_plane=n\_stages[3], count=n, stride=(2, 2))(conv3)

batch\_norm = BatchNormalization(axis=self.\_channel\_axis)(conv4)

relu = Activation("relu")(batch\_norm)

# Classifier block

pool = AveragePooling2D(pool\_size=(8, 8), strides=(1, 1), padding="same")(relu)

flatten = Flatten()(pool)

predictions\_g = Dense(units=2, kernel\_initializer=self.\_weight\_init, use\_bias=self.\_use\_bias,

kernel\_regularizer=l2(self.\_weight\_decay), activation="softmax")(flatten)

predictions\_a = Dense(units=101, kernel\_initializer=self.\_weight\_init, use\_bias=self.\_use\_bias,

kernel\_regularizer=l2(self.\_weight\_decay), activation="softmax")(flatten)

model = Model(inputs=inputs, outputs=[predictions\_g, predictions\_a])

return model

**6. TESTING**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## Implementation

## The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

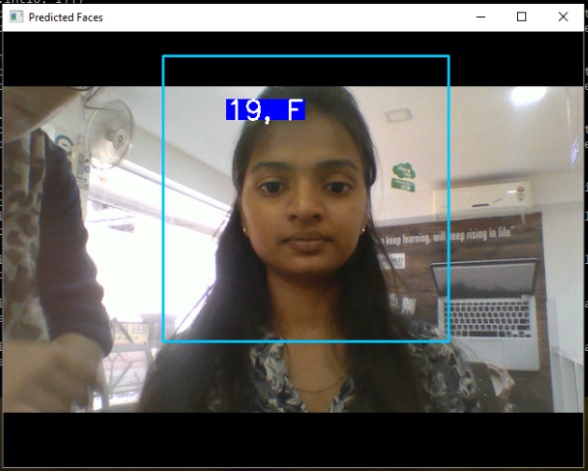
After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

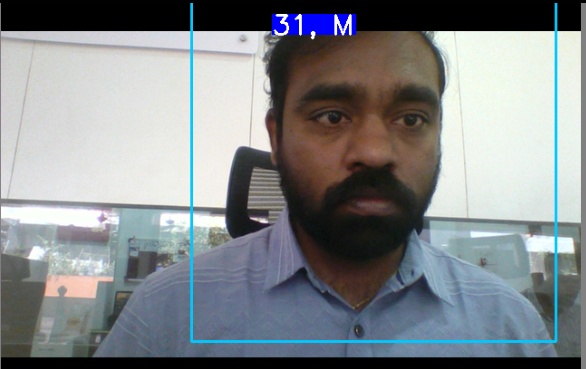
**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Start webcam | Test whether the webcam is started or not | If the webcam may not started | We cannot do the further operations | The webcam started successfully | High | High |
| 02 | Predict results | Verify the predicted results displayed or not | Without capturing the user image | We cannot get the age and gender predicted results | This displays age and gender successfully | High | High |

**7. SCREENSHOTS**





**8. 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 labeled for supervised learning. Taking example from the related problem of face recognition we explore how well deep CNN perform on these tasks using Internet data. We provide results with a lean deep-learning architecture designed to avoid overﬁtting due to the limitation of limited labeled data. Our network is “shallow” compared to some of the recent network architectures, thereby reducing the number of its parameters and the chance for overﬁtting. We further inﬂate the size of the training data by artiﬁcially adding cropped versions of the images in our training set. The resulting system was tested on the Adience benchmark of unﬁltered images and shown to signiﬁcantly outperform recent state of the art. 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 labeled 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 here.

**REFERENCES**

[1] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. Trans. Pattern Anal. Mach. Intell., 28(12):2037–2041, 2006. 2

[2] S. Baluja and H. A. Rowley. Boosting sex identiﬁcation performance. Int. J. Comput. Vision, 71(1):111–119, 2007. 2

[3] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In Int. Conf. Mach. Learning, volume 3, pages 11–18, 2003. 2

[4] W.-L. Chao, J.-Z. Liu, and J.-J. Ding. Facial age estimation based on label-sensitive learning and age-oriented regression. Pattern Recognition, 46(3):628–641, 2013. 1, 2

[5] K. Chatﬁeld, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531, 2014. 3

[6] J. Chen, S. Shan, C. He, G. Zhao, M. Pietikainen, X. Chen, and W. Gao. Wld: A robust local image descriptor. Trans. Pattern Anal. Mach. Intell., 32(9):1705–1720, 2010. 2

[7] S. E. Choi, Y. J. Lee, S. J. Lee, K. R. Park, and J. Kim. Age estimation using a hierarchical classiﬁer based on global and local facial features. Pattern Recognition, 44(6):1262–1281, 2011. 2

[8] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In European Conf. Comput. Vision, pages 484–498. Springer, 1998. 2

[9] C. Cortes and V. Vapnik. Support-vector networks. Machine learning, 20(3):273–297, 1995. 2

[10] E. Eidinger, R. Enbar, and T. Hassner. Age and gender estimation of unﬁltered faces. Trans. on Inform. Forensics and Security, 9(12), 2014. 1, 2, 5, 6

[11] Y. Fu, G. Guo, and T. S. Huang. Age synthesis and estimation via faces: A survey. Trans. Pattern Anal. Mach. Intell., 32(11):1955–1976, 2010. 2

[12] Y. Fu and T. S. Huang. Human age estimation with regression on discriminative aging manifold. Int. Conf. Multimedia, 10(4):578–584, 2008. 2

[13] K. Fukunaga. Introduction to statistical pattern recognition. Academic press, 1991. 2

[14] A. C. Gallagher and T. Chen. Understanding images of groups of people. In Proc. Conf. Comput. Vision Pattern Recognition, pages 256–263. IEEE, 2009. 2, 5

[15] F. Gao and H. Ai. Face age classiﬁcation on consumer images with gabor feature and fuzzy lda method. In Advances in biometrics, pages 132–141. Springer, 2009. 1, 2

[16] X. Geng, Z.-H. Zhou, and K. Smith-Miles. Automatic age estimation based on facial aging patterns. Trans. Pattern Anal. Mach. Intell., 29(12):2234–2240, 2007. 2

[17] B. A. Golomb, D. T. Lawrence, and T. J. Sejnowski. Sexnet: A neural network identiﬁes sex from human faces. In Neural Inform. Process. Syst., pages 572–579, 1990. 2

[18] A. Graves, A.-R. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 6645–6649. IEEE, 2013. 3

[19] G. Guo, Y. Fu, C. R. Dyer, and T. S. Huang. Imagebased human age estimation by manifold learning and locally adjusted robust regression. Trans. Image Processing, 17(7):1178–1188, 2008. 2

40

[20] G. Guo, G. Mu, Y. Fu, C. Dyer, and T. Huang. A study on automatic age estimation using a large database. In Proc. Int. Conf. Comput. Vision, pages 1986–1991. IEEE, 2009. 2 [21] H. Han, C. Otto, and A. K. Jain. Age estimation from face images: Human vs. machine performance. In Biometrics (ICB), 2013 International Conference on. IEEE, 2013. 2

[22] T. Hassner. Viewing real-world faces in 3d. In Proc. Int. Conf. Comput. Vision, pages 3607–3614. IEEE, 2013. 6 [23] T. Hassner, S. Harel, E. Paz, and R. Enbar. Effective face frontalization in unconstrained images. Proc. Conf. Comput. Vision Pattern Recognition, 2015. 5, 6

[24] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012. 5

[25] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007. 3, 5

[26] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093, 2014. 5

[27] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classiﬁcation with convolutional neural networks. In Proc. Conf. Comput. Vision Pattern Recognition, pages 1725–1732. IEEE, 2014. 3

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classiﬁcation with deep convolutional neural networks. In Neural Inform. Process. Syst., pages 1097–1105, 2012. 3, 4

[29] Y. H. Kwon and N. da Vitoria Lobo. Age classiﬁcation from facial images. In Proc. Conf. Comput. Vision Pattern Recognition, pages 762–767. IEEE, 1994. 1, 2

[30] A. Lanitis. The FG-NET aging database, 2002. Available: www-prima.inrialpes.fr/FGnet/html/ benchmarks.html. 2

[31] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989. 1, 3

[32] C. Liu and H. Wechsler. Gabor feature based classiﬁcation using the enhanced ﬁsher linear discriminant model for face recognition. Trans. Image Processing, 11(4):467–476, 2002. 1, 2

[33] P. Luo, X. Wang, and X. Tang. Hierarchical face parsing via deep learning. In Proc. Conf. Comput. Vision Pattern Recognition, pages 2480–2487. IEEE, 2012. 3

[34] E. Makinen and R. Raisamo. Evaluation of gender classiﬁcation methods with automatically detected and aligned faces. Trans. Pattern Anal. Mach. Intell., 30(3):541–547, 2008. 2

[35] B. Moghaddam and M.-H. Yang. Learning gender with support faces. Trans. Pattern Anal. Mach. Intell., 24(5):707– 711, 2002. 2 [36] X. Niyogi. Locality preserving projections. In Neural Inform. Process. Syst., volume 16, page 153. MIT, 2004. 2

[37] A. J. O’toole, T. Vetter, N. F. Troje, H. H. B¨ulthoff, et al. Sex classiﬁcation is better with three-dimensional head structure than with image intensity information. Perception, 26:75–84, 1997. 2

[38] C. Perez, J. Tapia, P. Est´evez, and C. Held. Gender classiﬁcation from face images using mutual information and feature fusion. International Journal of Optomechatronics, 6(1):92– 119, 2012. 2

[39] P. J. Phillips, H. Wechsler, J. Huang, and P. J. Rauss. The feret database and evaluation procedure for face-recognition algorithms. Image and vision computing, 16(5):295–306, 1998. 2, 3

[40] L. Rabiner and B.-H. Juang. An introduction to hidden markov models. ASSP Magazine, IEEE, 3(1):4–16, 1986. 2