Face Mask Recognition using deep learning

A Micro Project Report

Submitted by

K.Chandu
Reg.no: 99220041232

B.Tech – CSE,AIML

Kalasalingam Academy of Research and Education

(Deemed to be University)

Anand Nagar, Krishnankoil - 626 126

February 2024

KALASALINGAM
ACADEMY OF RESEARCH AND EDUCATION
(DEEMED TO BE UNIVERSITY)
Under sec. 3 of UGC Act 1956. Accredited by NAAC with "A" Grade

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Bonafide record of the work done by K.Chandu – 99220041232 in partial fulfill-ment of the requirements for the award of the degree of Bachelor of Technology in Specialization of the Computer Science and Engineering, during the Academic Year Even Semester (2023-24)

Mrs.K.Deepa Lakshmi
Project Guide
[Designation]
CSE
Kalasalingam Academy of Research and Education
Krishnan kovil - 626126

Mrs.S.Reshni
Faculty In charge
[Designation]
CSE
Kalasalingam Academy of Research and Education
Krishnan kovil - 626126

Mr.R.Kalai Selvan
Evaluator
[Designation]
CSE
Kalasalingam Academy of Research and Education
Krishnan kovil - 626126

Abstract

Global pandemic COVID-19 circumstances emerged in an epidemic of dangerous disease in all over the world. Wearing a face mask will help prevent the spread of infection and prevent the individual from contracting any airborne infectious germs. Using Face Mask Detection System, one can monitor if the people are wearing masks or not. Here HAAR-CASACADE algorithm is used for image detection. Collating with other existing algorithms, this classifier produces a high recognition rate even with varying expressions, efficient feature selection and low assortment of false positive features. HAAR feature based cascade classifier system utilizes only 200 features out of 6000 features to yield a recognition rate of 85-95%. According to this motivation we demand mask detection as a unique and public health service system during the global pandemic COVID-19 epidemic. The model is trained by face mask image and non-face mask image.

Keywords: COVID-19 epidemic, HAAR-CASACADE algorithm, mask detection, face mask image, non-face mask image

Contents

List of Figures

Chapter 1

Introduction

Face mask detection refers to detect whether a person is wearing a mask or not. In fact, the problem is reverse engineering of face detection where the face is detected using different machine learning algorithms for the purpose of security, authentication and surveillance. Face detection is a key area in the field of Computer Vision and Pattern Recognition. A significant body of research has contributed sophisticated to algorithms for face detection in past. The primary research on face detection was done in 2001 using the design of handcraft feature and application of traditional machine learning algorithms to train effective classifiers for detection and recognition. The problems encountered with this approach include high complexity in feature design and low detection accuracy. In recent years, face detection methods based on deep convolutional neural networks (CNN) have been widely developed to improve detection performance. Although numerous researchers have committed efforts in designing efficient algorithms for face detection and recognition but there exists an essential difference between 8detection of the face under mask9 and 8detection of mask over face9. As per available literature, very little body of research is attempted to detect mask over face. Thus, our work aims to a develop technique that can accurately detect mask over the face in public areas (such as airports. railway stations, crowded markets, bus stops, etc.) to curtail the spread of Coronavirus and thereby contributing to public healthcare. Further, it is not easy to detect faces with/without a mask in public as the dataset available for detecting masks on human faces is relatively small leading to the hard training of the model. So, the concept of transfer learning is used here to transfer the learned kernels from networks trained for a similar face detection task on an extensive dataset. The dataset covers various face images including faces with masks, faces without masks, faces with and without masks in one image and confusing images without masks. With an extensive dataset containing 45,000 images, our technique achieves outstanding accuracy of 98.2%. The major contribution of the proposed work is given below:

1. Develop a novel object detection method that combines one stage and two-stage detectors for accurately detecting the object in real-time from video streams with transfer learning at the back end.
2. Develop a novel object detection method that combines one stage and two-stage detectors for accurately detecting the object in real-time from video streams with transfer learning at the back end.
3. Creation of unbiased facemask dataset with imbalance ratio equals to nearly one.
4. The proposed model requires less memory, making it easily deployable for embedded devices used for surveillance purposes.

1.1 Deep Learning Approaches for Face Mask Recognition

These deep learning approaches provide a foundation for developing accurate and efficient face mask recognition systems capable of detecting masked faces in various real-world scenarios, contributing to public health and safety initiatives. Evaluate the effectiveness of transfer learning compared to training CNNs from scratch, considering factors such as accuracy, training time, and model complexity.

1.2 Dataset Acquisition and Preprocessing

Effective dataset acquisition and preprocessing are crucial steps in developing a robust face mask recognition system. By carefully curating and preparing the dataset, researchers can improve model performance, enhance generalization capabilities, and facilitate the development of accurate and reliable mask detection models for real-world applications. Address potential class imbalances in the dataset by applying techniques such as oversampling, undersampling, or class-weighted loss functions to ensure equal representation of mask and non-mask instances during model training.

1.2.1 Transfer Learning Techniques in Mask Recognition Models

Transfer learning techniques in mask recognition models involve leveraging pre-trained deep learning models that were initially trained on large-scale datasets for generic object recognition tasks, and fine-tuning them for the specific task of mask detection. This approach offers several advantages, including reduced training time, improved performance with limited labeled data, and the ability to capture complex features relevant to mask detection.

1. **Selection of Pre-trained Models**: Evaluate the suitability of different pre-trained models for the mask recognition task based on factors like model complexity, accuracy, and resource requirements.

2. **Fine-tuning Strategies**: Explore methods for setting learning rates, batch sizes, and other hyperparameters during fine-tuning to achieve optimal performance without overfitting.

1.2.2 Data Augmentation and Cleaning Strategies for Improved Model Generalization

Introduce random rotations of images by certain degrees to simulate variations in pose and orientation. Introduce random rotations of images by certain degrees to simulate variations in pose and orientation. Resize images to different scales or crop them to varying sizes to introduce variations in object sizes and positions. Inject Gaussian noise into images to mimic real-world noise and improve model robustness to environmental variations. Randomly adjust brightness, contrast, saturation, and hue of images to simulate changes in lighting conditions. Apply elastic deformations to images to mimic distortions caused by facial movements or occlusions.

1. **Preprocessing Steps**:
    - Resizing and Normalization: Resize images to a fixed resolution and normalize pixel values to a standard range to ensure consistency across the dataset.
    - Grayscale Conversion: Convert images to grayscale to reduce computational complexity and remove color-based variations that are irrelevant to mask detection.

2. **Class Imbalance Handling**:
    - Oversampling: Duplicate samples from minority classes to balance class distribution and prevent bias towards the majority class.
    - Undersampling: Randomly remove samples from the majority class to achieve a balanced dataset, reducing computational overhead and preventing overfitting.

Chapter 2

Tool and Technology Used

2.1 Deep Learning Frameworks and Libraries

Deep learning frameworks and libraries serve as essential tools for researchers, developers, and practitioners in the field of artificial intelligence, enabling them to build, train, and deploy neural network models efficiently. TensorFlow and Keras stand out as prominent players in this domain. TensorFlow, developed by Google, offers a comprehensive ecosystem with robust support for building and training deep neural networks. Its flexibility allows for seamless deployment across various platforms, including mobile devices and distributed systems, making it a popular choice for both research and production environments. Keras, on the other hand, provides a user-friendly interface built on top of TensorFlow.

2.1.1 TensorFlow and Keras:

- **TensorFlow**: TensorFlow is an open-source deep learning framework developed by Google that provides comprehensive tools and libraries for building and training neural networks. It offers flexibility and scalability, allowing developers to deploy models in various environments, including mobile devices and cloud servers.

- **Keras:** Keras is a high-level neural networks API written in Python that serves as an interface for TensorFlow, allowing for fast experimentation and prototyping of deep learning models. It provides a user-friendly interface for building and training neural networks with minimal code, making it ideal for beginners and researchers.

2.1.2 Python :

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

2.2.1 Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Create publication quality plots. Make interactive figures that can zoom, pan, update. Customize visual style and layout. Export to many file formats .Embed in Jupyter Lab and Graphical User Interfaces. Use a rich array of third-party packages built on Matplotlib.

2.2.2 Numpy and Scipy

1) **Numpy** : NumPy (pronounced /ˈnʌmpaɪ/ (NUM-py) or sometimes /ˈnʌmpi/ (NUM-pee)) is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Num array into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project.

2) **Scipy :** Scipy provides algorithms for optimization, integration, interpolation, eigenvalue problems, algebraic equations, differential equations, statistics and many other classes of problems. SciPy's high level syntax makes it accessible and productive for programmers from any background or experience level. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

2.3 Development Environment and Collaboration Tools

Development environments and collaboration tools play a crucial role in facilitating efficient software development workflows, particularly in the realm of deep learning and AI. PyCharm is a standout development environment renowned for its robust features tailored specifically for Python programming. Its integrated interface encompasses a rich set of tools such as code editor, debugger, and version control integration, streamlining the development process and enhancing productivity. With PyCharm, developers can write, debug, and manage Python projects seamlessly, ensuring code quality and consistency.

2.3.1 PyCharm

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. It provides a productive environment for writing, debugging, and managing Python projects, suitable for individual developers and teams.

2.3.2 Version Control and Collaboration Tools

1. **Git:** Git is a distributed version control system used for tracking changes in code and coordinating work among multiple developers. It allows for efficient collaboration, code review, and project management, enabling teams to maintain a history of changes and easily revert to previous versions if needed.

2. **GitHub:** GitHub is a web-based hosting service for version control using Git, providing features for code hosting, collaboration, and project management. It offers tools for code review, issue tracking, and continuous integration, facilitating collaborative software development workflows.

Chapter 3

System Development Overview

3.1 Design

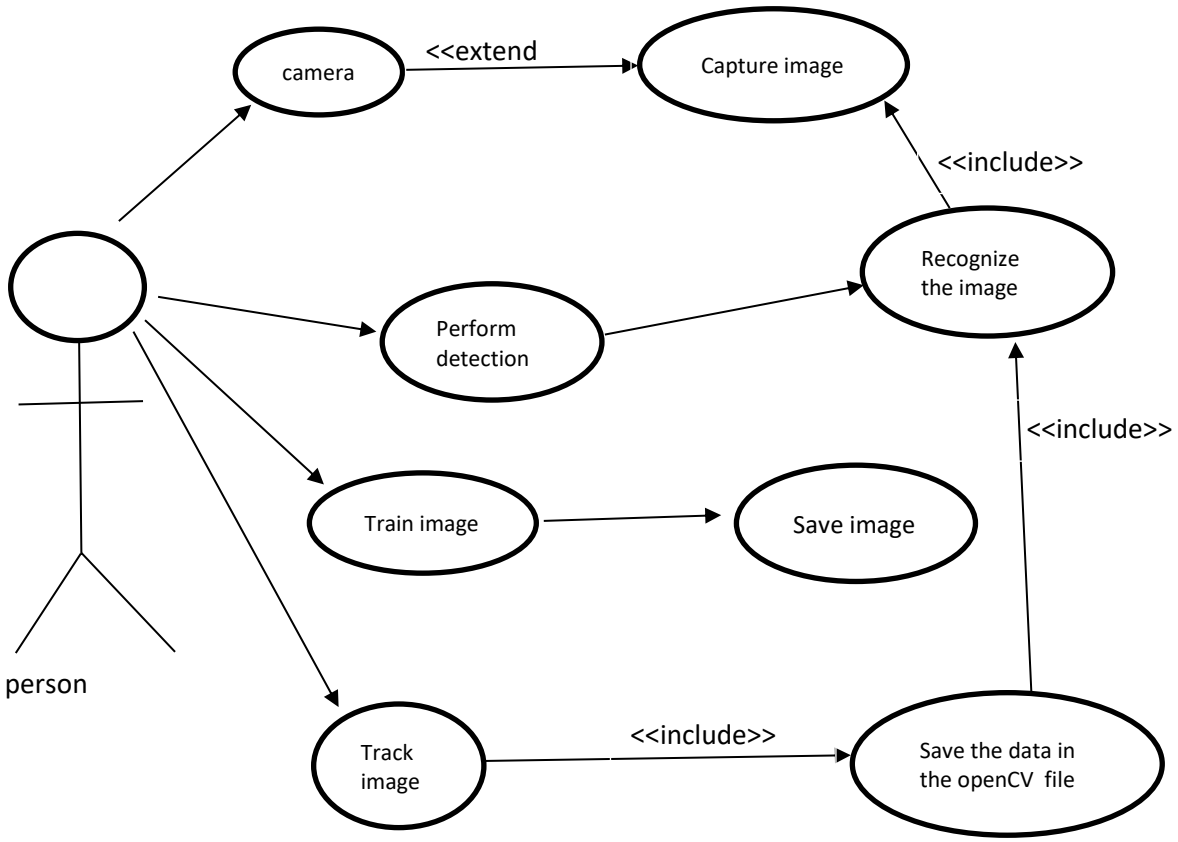
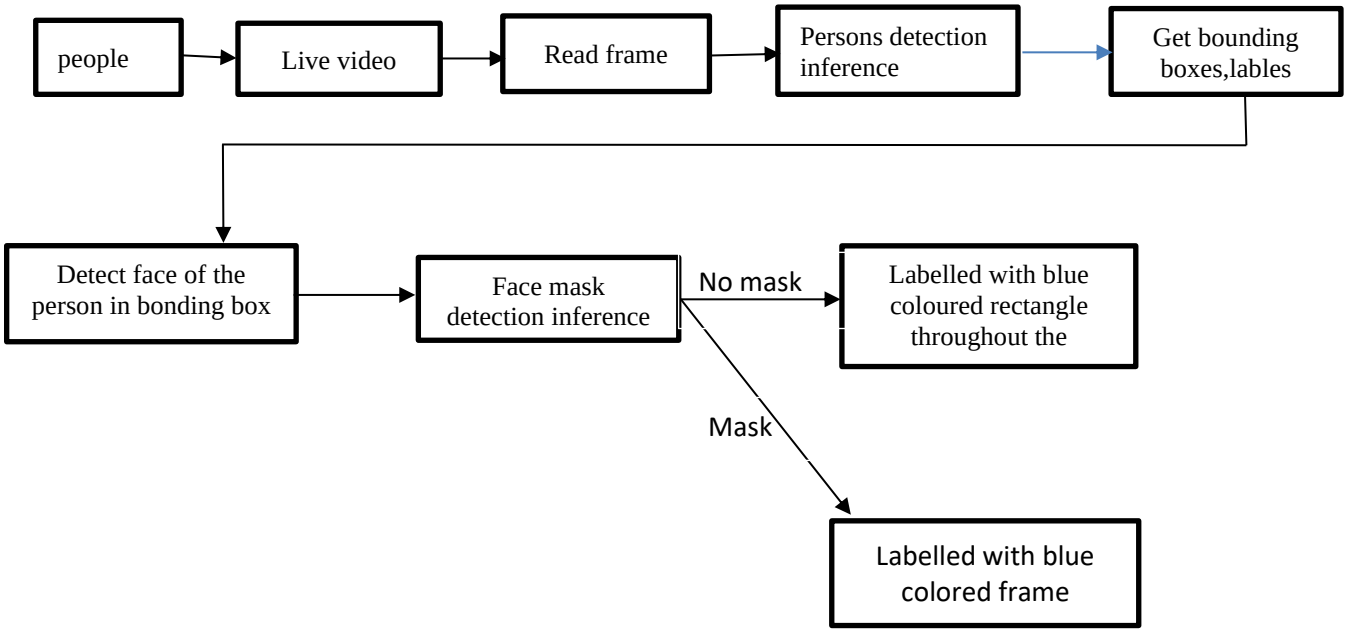


Figure 1.1 use case diagram

3.1.1 Block diagram



code

```
# USAGE
# python detect_mask_video.py

# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2
import os

def detect_and_predict_mask(frame, faceNet, maskNet):
	# grab the dimensions of the frame and then construct a blob
	# from it
	(h, w) = frame.shape[:2]
	blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),
		(104.0, 177.0, 123.0))

	# pass the blob through the network and obtain the face detections
	faceNet.setInput(blob)
	detections = faceNet.forward()
```

```
	# initialize our list of faces, their corresponding locations,
	# and the list of predictions from our face mask network
	faces = []
	locs = []
	preds = []

	# loop over the detections
	for i in range(0, detections.shape[2]):
		# extract the confidence (i.e., probability) associated with
		# the detection
		confidence = detections[0, 0, i, 2]

		# filter out weak detections by ensuring the confidence is
		# greater than the minimum confidence
		if confidence > args["confidence"]:
			# compute the (x, y)-coordinates of the bounding box for
			# the object
			box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
			(startX, startY, endX, endY) = box.astype("int")

			# ensure the bounding boxes fall within the dimensions of
			# the frame
			(startX, startY) = (max(0, startX), max(0, startY))
			(endX, endY) = (min(w - 1, endX), min(h - 1, endY))

			# extract the face ROI, convert it from BGR to RGB channel
			# ordering, resize it to 224x224, and preprocess it
			face = frame[startY:endY, startX:endX]
			if face.any():
				face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
				face = cv2.resize(face, (224, 224))
				face = img_to_array(face)
				face = preprocess_input(face)

				# add the face and bounding boxes to their respective
				# lists
				faces.append(face)
				locs.append((startX, startY, endX, endY))

	# only make a predictions if at least one face was detected
	if len(faces) > 0:
		# for faster inference we'll make batch predictions on *all*
		# faces at the same time rather than one-by-one predictions
		# in the above `for` loop
		faces = np.array(faces, dtype="float32")
		preds = maskNet.predict(faces, batch_size=32)

	# return a 2-tuple of the face locations and their corresponding
	# locations
	return (locs, preds)

# construct the argument parser and parse the arguments
```

```
ap = argparse.ArgumentParser()
ap.add_argument("-f", "--face", type=str,
	default="face_detector",
	help="path to face detector model directory")
ap.add_argument("-m", "--model", type=str,
	default="mask_detector.model",
	help="path to trained face mask detector model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
	help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# load our serialized face detector model from disk
print("[INFO] loading face detector model...")
prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"],
	"res10_300x300_ssd_iter_140000.caffemodel"])
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
print("[INFO] loading face mask detector model...")
maskNet = load_model(args["model"])

# initialize the video stream and allow the camera sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)

# loop over the frames from the video stream
while True:
	# grab the frame from the threaded video stream and resize it
	# to have a maximum width of 400 pixels
	frame = vs.read()
	frame = imutils.resize(frame, width=400)

	# detect faces in the frame and determine if they are wearing a
	# face mask or not
	(locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

	# loop over the detected face locations and their corresponding
	# locations
	for (box, pred) in zip(locs, preds):
		# unpack the bounding box and predictions
		(startX, startY, endX, endY) = box
		(mask, withoutMask) = pred

		# determine the class label and color we'll use to draw
		# the bounding box and text
		label = "Mask" if mask > withoutMask else "No Mask"
		color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

		# include the probability in the label
		label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
```

```
                # display the label and bounding box rectangle on the output
                # frame
                cv2.putText(frame, label, (startX, startY - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
                cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

        # show the output frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
                break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

3.1.2 Functional requirements & Non-Functional requirements

1. **FUNCTIONAL REQUIREMENTS**

   The primary purpose of computer results is to deliver processing results to users. They are also employed to maintain a permanent record of the results for future use. In general, the following are many types of results :

   - External results are those that are exported outside the company.
   - Internal results, which are the main user and computer display and have a place within the organization.
   - Operating results used only by the computer department
   - User-interface results that allow the user to communicate directly with the system.
   - Understanding the user's preferences, the level of technology and the needs of his or her business through a friendly questionnaire.

2. **NON-FUNCTIONAL REQUIREMENTS**

   SYSTEM CONFIGURATION

   This project can run on commodity hardware. We ran entire project on an AMD Ryzen5 processor with 16 GB Ram, 4 GB Nvidia Graphic Processor, It also has 6 cores which runs at 1.7 GHz, 2.1 GHz respectively. First part of the is training phase which takes 10-15 mins of time and the second part is testing part which only takes few seconds to make predictions and calculate accuracy.

   HARDWARE REQUIREMENTS

   • RAM: 4 GB
   • Storage: 500 GB
   • CPU: 2 GHz or faster
   • Architecture: 32-bit or 64-bit

   SOFTWARE REQUIREMENTS

   • Python 3.5 in PyCharm is used for data pre-processing, model training and prediction
   • Operating System: windows 7 and above or Linux based OS or MAC OS
   • Coding Language: Python.
   • Nvidia Graphic Processor

Chapter 4

Conclusion and Future Work

As the technology are blooming with emerging trends the availability so we have novel face mask detector which can possibly contribute to public healthcare. The architecture consists of Mobile Net as the backbone it can be used for high and low computation scenarios. In order to extract more robust features, we utilize transfer learning to adopt weights from a similar task face detection, which is trained on a very large dataset. We used OpenCV, tensor flow, and NN to detect whether people were wearing face masks or not. The models were tested with images and real-time video streams. The accuracy of the model is achieved and, the optimization of the model is a continuous process and we are building a highly accurate solution by tuning the hyper parameters. This specific model could be used as a use case for edge analytics. Furthermore, the proposed method achieves state-of the-art results on a public face mask dataset. By the development of face mask-detection we can detect if the person is wearing a face mask and allow their entry would be of great help to the society.

4.1 Future enhancements

Jingdong's recognition accuracy is stronger than 99 percent. We created the MFDD, RMFRD, and SMFRD datasets, as well as a cutting-edge algorithm based on them. The algorithm will provide contactless face authentication in settings such as community access, campus governance, and enterprise resumption. Our research has given the world more scientific and technological strength.

4.2 Result

INPUT AND OUTPUT

Chapter 5

References

1. Militante, S. V., & Dionisio, N. V. (2020). Real-Time Face Mask Recognition with Alarm System using Deep Learning. 2020 11th IEEE Control and System Graduate Research Colloquium (ICSGRC), Shah Alam, Malaysia. https://doi.org/10.1109/ ICSGRC49013.2020.9232610 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero.Sed cursus ante dapibus diam. Sed nisi.

2. Guillermo, M., Pascua, A. R. A., Billones, R. K., Sybingco, E., Fillone, A., & Dadios, E. (2020). COVID19 Risk Assessment through Multiple Face Mask Detection using MobileNetV2 DNN. The 9th International Symposium on Computational Intelligence and Industrial Applications (ISCIIA2020), Beijing, China. https:// isciia2020.bit.edu.cn/docs/20201114082420135149. PdfLorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero.Sed cursus ante dapibus diam. Sed nisi.

3. Boyko, N., Basystiuk, O., & Shakhovska, N. (2018). Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library. 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine. https://doi.org/10.1109/DSMP.2018.8478556

4. Cousera

Chapter 6

Certification

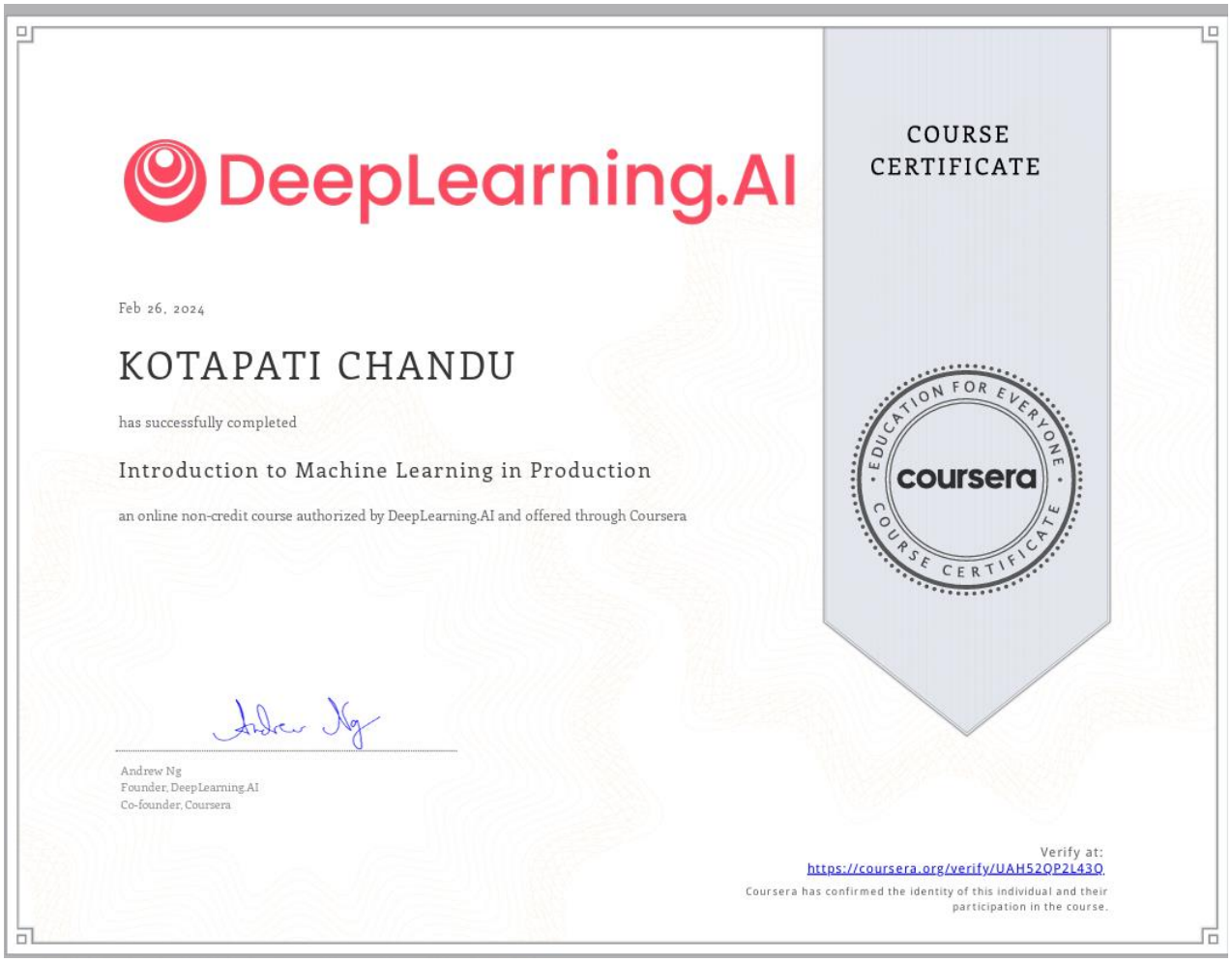


Figure 6.1: Certification details