

Elmar Wings

Artificial Intelligence with an Arduino Portenta H7

Realtime Object Detection with a Vision Shield

Version: 1765
April 11, 2024

Contents

Contents	3
List of Figures	11
List of Listings	15
I. Machine Learning	19
1. Artificial intelligence - When are machines intelligent?	21
1.1. What is Artificial Intelligence?	21
1.2. Machinelles Lernen	22
1.3. Deep Learning	23
1.4. Application	23
1.5. Limits of Artificial Intelligence	24
2. Processes for Knowledge Acquisition in Databases	27
2.1. Knowledge Discovery in Databases (KDD Process)	27
2.1.1. Application Areas	27
2.1.2. Process Model	27
2.1.3. KDD-Prozess based on Fayyad	27
2.2. Generic Process Model	30
2.2.1. Application Domain Understanding	30
2.2.2. Understanding the Data (Data Understanding)	30
2.2.3. Data Preparation and Identification of DM Technology	30
2.2.4. Data Mining	30
2.2.5. Evaluation	30
2.2.6. Knowledge Deployment (Knowledge Consolidation and Deployment)	30
2.3. CRISP-DM	30
2.4. Application-Oriented Process Models	31
2.5. Role of the User	32
2.6. KDD Process using the Example of Image Classification with an Edge Computer	32
2.6.1. Database	33
2.6.2. Data Selection	33
2.6.3. Data Preparation	33
2.6.4. Data Transformation	33
2.6.5. Data Mining	33
2.6.6. Evaluation/Verification	34
2.6.7. Model	34
2.7. Data Analysis and Data Mining Methods at a Glance	34
2.7.1. Definition Data Analysis	34
2.7.2. Statistical Data Analysis	34
2.7.3. Overview of Data Mining Methods and Procedures	36

3. Neural Networks	39
3.1. Mathematical Modelling	39
3.1.1. Weights	40
3.1.2. Bias	41
3.1.3. Activation and Output Function	41
3.2. Training	43
3.2.1. Propagation	43
3.2.2. Error	43
3.2.3. Delta Rule	43
3.2.4. Error Backpropagation	44
3.2.5. Problem of Overfitting	44
3.2.6. Epochs, Batches and Steps	44
3.2.7. Dropout	44
3.2.8. Correct Classification Rate and Loss Function	45
3.2.9. Training Data, Validation Data and Test Set	45
3.2.10. Transfer Learning	45
3.2.11. Weight Imprinting	45
3.3. Convolutional Neural Networks	47
3.3.1. Convolutional Layer	47
3.3.2. Pooling Layer	48
3.3.3. Fully Connected Layer	48
3.3.4. Hyperparameter	48
3.3.5. Fine-tuning	49
3.3.6. Feature Extraction	49
3.3.7. AlexNet	49
3.3.8. YOLO	50
3.4. Images and Image Classification with a Convolutional Neural Network (CNN)	51
3.4.1. Representation of Images	52
3.5. Convolution Neural Network	54
3.5.1. CNN Architectures	55
3.5.2. LeNet	55
3.5.3. AlexNet	55
3.5.4. InceptionNet	56
3.5.5. VGG	56
3.5.6. Residual Neural Network (ResNet)	56
3.5.7. MobileNet & MobileNetV2	56
3.5.8. EfficientNet	56
3.5.9. General Structure	57
3.5.10. Input and Output	57
3.5.11. Convolution Layer	58
3.5.12. Padding	60
3.5.13. Activation function	60
3.5.14. Pooling	61
3.5.15. Flattening	62
3.5.16. Dense Layer	62
3.5.17. Model used for a CIFAR 10 classification	62
4. Databases and Models for Machine Learning	65
4.1. Datenbanken	66
4.1.1. Data Set MNIST	66
4.1.2. CIFAR-10 and CIFAR-100	67
4.1.3. Fisher's Iris Data Set	69
4.1.4. Common Objects in Context - COCO	72
4.1.5. ImageNet	77

4.1.6. Visual Wake Words	77
4.2. Datensatz UTKFace	77
4.3. Models	78
4.4. FaceNet	78
5. Frameworks and Libraries	81
5.1. Development Environment	81
5.1.1. Jupyter-Notebook	81
5.2. Frameworks	81
5.2.1. TensorFlow	81
5.2.2. TensorFlow Lite	82
5.2.3. TensorRT	82
5.2.4. Keras	82
5.2.5. PyTorch	82
5.2.6. Caffe	83
5.2.7. Caffe2	83
5.2.8. Theano	83
5.2.9. Apache MXNet	83
5.2.10. CNTK	83
5.2.11. Deeplearning4J	83
5.2.12. Chainer	83
5.2.13. FastAI	83
5.3. General libraries	84
5.3.1. NumPy	84
5.3.2. SciPy	84
5.3.3. pandas	84
5.3.4. IPython	84
5.3.5. Matplotlib	84
5.3.6. scikit-learn	84
5.3.7. Scrapy	85
5.3.8. NLTK	85
5.3.9. Pattern	85
5.3.10. Seaborn	85
5.3.11. Bokeh	85
5.3.12. basemap	85
5.3.13. NetworkX	85
5.3.14. LightGBM	85
5.3.15. Eli5	86
5.3.16. Mlpy - Machine Learning	86
5.3.17. Statsmodels - Statistical Data Analysis	86
5.4. Specialised Libraries	86
5.4.1. OpenCV	86
5.4.2. OpenVINO	86
5.4.3. Compute Unified Device Architecture (CUDA)	86
5.4.4. OpenNN	87
5.5. Special Libraries	87
5.5.1. glob	87
5.5.2. os	87
5.5.3. PIL	87
5.5.4. LCC15	87
5.5.5. math	87
5.5.6. cv2	87
5.5.7. random	87
5.5.8. pickle	88
5.5.9. PyPI	88

6. Libraries, Modules, and Frameworks	89
6.1. TensorFlow Framework	89
6.1.1. TensorFlow Use Case	89
6.2. OpenCV	90
6.2.1. Application of OpenCV	90
6.2.2. Image Processing	90
6.2.3. How does a computer read an image	90
6.3. MediaPipe	91
6.4. MediaPipe Applications	91
6.4.1. Hand Landmarks Detection	91
6.4.2. Face Detection	92
7. Gesichtserkennung	93
7.0.1. Merkmalsbasierter Ansatz	93
7.0.2. Bildbasierter Ansatz	93
7.0.3. Herausforderungen	93
7.0.4. Lösungen	94
7.0.5. Anwendungen	94
II. Tools	95
8. Arduino IDE 1.8.x	97
8.1. Introduction	97
8.2. Arduino IDE Description	97
8.3. Installation	97
9. First Steps with the Portenta H7	101
9.1. Introduction	101
9.2. Configuration	101
9.2.1. Example Blink Sketch	101
9.2.2. Serial Blink Example for Dual Core Processing	103
10. Kmandozeileninterpreter (CLI)	107
10.1. Installation und Verwendung des CLI	107
10.2. Konfiguration des Arduino CLI	108
10.3. Funktionsübersicht	108
10.3.1. Grundfunktionen	110
10.4. Erste Schritte mit dem Arduino Nano 33 BLE Sense Lite mit Hilfe des CLI	111
10.4.1. Erkennung der angeschlossenen Boards	111
10.4.2. Erstellung eines neuen Sketches	111
10.4.3. Kompilieren eines Sketches	112
10.4.4. Hochladen eines Sketches	112
10.4.5. Installation von Bibliotheken	113
10.4.6. Beispiel „Hello World“	113
10.5. Beschreibung der Software auf dem PC	114
11. OpenMV IDE	119
11.0.1. OpenMV IDE Installation	119
11.0.2. OpenMV IDE Overview	122
12. First Steps with the Vision Shield	125
12.1. Introduction	125

12.2. OpenMV Examples	125
12.2.1. Creating a basic face filter with OpenMV	125
12.2.2. Face tracking	126
12.2.3. Finding circles around the objects	126
12.2.4. QR Code information extraction	127
12.2.5. Blob-Erkennung mit OpenMV	128
12.2.6. Blob-Erkennung	128
12.2.7. Erstellen des MicroPython-Skripts	128
12.2.8. Festlegen der Farbschwellenwerte	129
12.2.9. Erkennung von Blobs	130
12.2.10. LEDs umschalten	132
12.2.11. Hochladen des Skripts	133
12.3. Edge Impulse	133
13. Google Colaboratory	137
13.1. Introduction	137
13.2. Tips for using Colab	137
13.3. Advantages of Colab	138
13.4. Selection of the Model: YOLO V4	138
13.5. Training the model	139
13.5.1. Architecture of YOLO V4	139
13.5.2. Enabling GPU in Google Colab	140
13.5.3. Building Darknet	140
13.5.4. Define Helper Functions	141
13.5.5. Moving the Datasets to Cloud VM	142
13.5.6. Configuring Files for Training	143
13.5.7. Downloading pretrained weights for the convolution layers	145
13.5.8. Training the custom object detector	145
13.5.9. Challenges	145
14. Training eines Gesichtserkennungsmodells mit Google Colaboratory	147
14.0.1. Google Colaboratory	147
14.1. Datenbank	148
14.1.1. Verbinden mit Google Drive	148
14.1.2. Einrichtung des Datensatzes	148
14.2. Vorverarbeitung von Daten	149
14.3. Training des Modells	149
14.4. Prüfung des Modells	150
14.5. Umwandlung des gespeicherten Modells in das tflite-Format	150
14.6. Konvertierung der tflite-Datei in eine Arduino-Header-Datei	151
14.7. Offene Fragen	151
14.7.1. Arduino Portenta H7-Board nicht kompatibel für Objekterkennung mit Arduino IDE	151
14.7.2. Einsatz des auf Edge Impulse trainierten Objekterkennungsmodells mit Arduino IDE nicht möglich	151
15. Quellcode	153
15.1. Face Detection using Edge Impulse and OpenMV	153
III. Portenta H7	155
16. Einleitung	157
17. Introduction	159

18. Hardware	161
18.1. Portenta H7 - Hardware	161
18.2. Tutorial	163
18.3. Hardware Installation	163
19. General Description of Softwares used in this project	165
19.1. Introduction	165
19.2. Arduino IDE Description	165
19.3. Installation	166
20. First Steps with the Portenta H7	169
20.1. Introduction	169
20.2. Configuration	169
20.2.1. Example Blink Sketch	169
20.2.2. Serial Blink Example for Dual Core Processing	171
21. First Steps with the Vision Shield	175
21.1. Introduction	175
21.2. Update Bootloader version	175
21.3. OpenMV Examples	176
21.3.1. Creating a basic face filter with OpenMV	176
21.3.2. Face tracking	177
21.3.3. Finding circles around the objects	178
21.3.4. QR Code information extraction	179
22. MicroPython & OpenMV	181
22.1. Introduction	181
22.2. MicroPython	181
22.3. Implementing MicroPython on Portenta H7 with OpenMV IDE	181
22.3.1. Example script for blinking builtin LED on the Arduino Portenta H7 with OpenMV	182
23. First Application: Object Detection	185
23.1. What is Face Detection	185
23.1.1. Challenges	185
23.1.2. Solutions	185
23.1.3. Applications	186
24. Training a Custom Machine Learning Model for Portenta H7	187
24.1. Introduction	187
24.2. How to train a face detection model	187
24.3. KDD Process	187
24.3.1. KDD Process for the object detection model	187
24.3.2. Database	187
24.3.3. Add Dataset in the OpenMV IDE	188
24.3.4. Export the Data to Edge Impulse platform	188
24.3.5. Data Selection	188
24.3.6. Data Transformation	188
24.3.7. Data training	190
24.3.8. Deploy to Arduino Porrtenta H7	192
24.3.9. Testing the trained model using Vision shield and OpenMV IDE	193
24.4. Face Detection using Edge Impulse	194
24.4.1. Database	195
24.4.2. Connecting the Arduino Portenta H7 with Edge Impulse	196
24.4.3. Data Labelling	198

Contents	9
-----------------	----------

24.5. Data transformation	199
24.5.1. Designing the Impulse	199
24.5.2. Configuring the Image processing block	199
24.6. Data training	200
24.7. Testing the model	201
25. Use TensorFlow Lite with Portenta H7	203
25.1. Introduction	203
25.2. Training a Face detection model using Google Colaboratory	203
25.2.1. Google Colaboratory	203
25.3. Database	203
25.3.1. Connecting to Google Drive	204
25.3.2. Setting up the Dataset	204
25.4. Data Preprocessing	204
25.5. Training the model	205
25.6. Testing the model	206
25.7. Converting the saved model to tflite format	206
25.8. Converting the tflite file to Arduino header file	206
25.9. Open Questions	207
25.9.1. Arduino Portenta H7 board not compatible for object detection using Arduino IDE	207
25.9.2. Unable to Deploy object detection model trained on Edge Impulse using Arduino IDE	207
26. Source Code	209
26.1. Serial Blink Example for Arduino Portenta H7	209
26.2. Example Script for blinking builtin RGB LED on the Arduino Portenta H7 with OpenMV	209
26.3. QR Code Information Extraction Program	209
26.4. Finding Circles around the Objects	210
26.5. Creating a basic face filter with OpenMV	211
26.6. Face Tracking	211
26.7. Face Detection using Edge Impulse and OpenMV	213
26.7.1. Beispiel für serielles Blinken bei Dual-Core-Verarbeitung	214
26.8. Arduino Portenta Vision Shield	216
26.8.1. Camera Module	217
26.8.2. Microphones	217
26.8.3. Micro SD card	217
26.8.4. Ethernet	217
26.8.5. LoRa Module	217
26.8.6. Power	217
26.9. Accessing IMU Data on Nicla Vision	218
26.9.1. Overview	218
26.9.2. Goals	218
26.9.3. Hardware & Software Needed	219
26.10. Proximity Detection with Arduino Nicla Vision	219
26.10.1. Overview	219
26.10.2. Goals	219
26.10.3. Instructions	219
26.10.4. Conclusion	220
27. Gesichtserkennung mit Edge Impulse	223
27.1. Datenbank	224
27.1.1. Kennzeichnung von Daten	225

27.2. Umwandlung von Daten	225
27.2.1. Die Gestaltung des Impulses	225
27.2.2. Konfigurieren des Bildverarbeitungsblocks	226
27.3. Data training	227
27.4. Prüfung des Modells	227
28. MicroPython & OpenMV	229
28.1. MicroPython	229
29. First Application: Object Detection	231
30. Training a Custom Machine Learning Model for Portenta H7	233
31. Use TensorFlow Lite with Portenta H7	235
IV. Anhang	237
32. Materialliste	239
33. Datenblätter	241
33.1. Datenübersicht	241
34. Datenblätter Arduino Vision Shield	291
Literaturverzeichnis	311
Stichwortverzeichnis	320
Index	321

List of Figures

1.1.	relationship of artificial intelligence, machine learning and deep learning	22
1.2.	Example of a Künstliches Neuronales Netz (KNN)	23
1.3.	Overlay of the image of a panda with noise [GSS14]	24
1.4.	Manipulation of traffic signs [Sit+18]	24
1.5.	Optimal image for the identification of a toaster [Sit+18]	25
2.1.	KDD process according to [FPSS96] (own reduced representation based on [FPSS96])	28
2.2.	Life cycle of a data mining project in CRISP-DM according to [Cha+00]	31
2.3.	Modified model of the KDD process based on the model according to [FPSS96]	31
2.4.	KDD process in the industrial environment	32
3.1.	Biological (left) and abstracted (right) model of neural networks. Source:[Ert16]	39
3.2.	Representation of a neuron as a simple threshold element	40
3.3.	Graph with weighted edges and equivalent weight matrix	40
3.4.	ReLU function (left) and Leaky ReLU	42
3.5.	Sigmoid function	42
3.6.	Funktion Tangens Hyperbolicus	43
3.7.	Illustration of the weight imprinting process [QBL18]	46
3.8.	Illustration of imprinting in the normalized embedding space. (a) Before imprinting, the decision boundaries are determined by the trained weights. (b) With imprinting, the embedding of an example (the yellow point) from a novel class defines a new region. [QBL18]	46
3.9.	The YOLO system	50
3.10.	Original image, scaling, rotation and cropping of the image	51
3.11.	Structure of a black and white image with 10×10 pixels	52
3.12.	Image with 3 colour channels	52
3.13.	Greyscale with a value range from 0 to 255	53
3.14.	Black and white image, image with 4 grey levels and with 256 grey levels.	53
3.15.	Softened image with a 5×5 filter	54
3.16.	Edge detection on an image	54
3.17.	Struktur der CNN-Architektur gemäß LeCun et.al. [LeC+98]	55
3.18.	Definition of the convolution of two matrices	58
3.19.	Convolution layer with 3×3 kernel and stride $(1, 1)$	58
3.20.	Beispielanwendung für einen Filter und ein Ausschnitt	59
3.21.	Visualisation of filters of a trained network [LXW19; Sha+16]	60
3.22.	Detection of vertical and horizontal edges	60
3.23.	Representation of the ReLu activation function	61
3.24.	Application of an activation function with a bias $b \in \mathbb{R}$ to a matrix	61
3.25.	Application of max-pooling and average-pooling	62
3.26.	Application of flattening	62
3.27.	Structure of the neural network used	63
4.1.	Examples from the training set of the dataset MNIST [SSS19]	67
4.2.	Extract from ten illustrations of the data set Canadian Institute For Advanced Research (CIFAR)-10	68

4.3. Supercategories and categories in Canadian Institute For Advanced Research (CIFAR)-100 with the original designations	69
4.4. Header lines of Fisher's Iris Data Set	70
4.5. Output of the categories of Fisher's Iris Data Set	72
4.6. Names of the categories in Fisher's Iris Data Set	72
4.7. Beispielbilder aus dem Datensatz UTKFace	77
5.1. Tensorflow Datenflussdiagramm	82
6.1. Hand Landmarks	91
6.2. Face Detection Using Landmarks	92
8.1. Menu bar options	98
8.2. Menu buttons	98
8.3. Arduino Creat Agent Installation	99
8.4. Arduino Setup Installation options	99
8.5. Arduino Setup: Installation Folder	99
8.6. Arduino Sketch	100
9.2. Install packages for Arduino Portenta H7	101
9.1. Arduino Portenta H7 Connected to a laptop	102
9.3. Upload Basic Sketch	103
9.4. Compile Blink Sketch	103
9.5. Serial Blink Sketch	104
9.6. Serial Blink Sketch	104
9.7. Output LED color	105
10.1. Installation von Arduino-CLI	108
10.2. Standardeinstellungen der Konfigurationsdatei	109
10.3. Rückgabewerte der „board list“-Funktion	111
10.4. Installieren des Kerns für den Arduino Nano 33 BLE Sense Lite	111
10.5. Kompilieren eines Sketches mit Command Line Interface (CLI)	112
10.6. Für das Kompilieren des Blink-Sketches wird der Dateipfad zu dem Sketch mit angegeben (rot unterstrichen). Außerdem sollte der FQBN des Arduinos, für den der Sketch kompiliert wird, genannt werden.	113
10.7. Hochladen des kompilierten Sketches aus dem Temp-Ordner	113
10.8. Die orange LED des Arduinos beginnt zu blinken	114
11.1. OpenMV IDE Installation	119
11.2. OpenMV IDE Installation folder	120
11.3. OpenMV IDE Setup Install	120
11.4. OpenMV IDE	121
11.5. Installing latest firmware	121
11.6. firmware update complete	122
12.1. Face Filter output	125
12.2. Face tracking output	126
12.3. Find circles output	127
12.4. QR Code output	127
12.5. LAB-Schwellenwerte für einen Apfel im Schwellenwert-Editor	130
12.6. LAB-Farbhistogramm des Bildpuffers	130
12.7. LAB-Schwellenwerte für eine Banane im Schwellenwert-Editor	131
12.8. Visualisierung der Blobs in der Bildpuffervorschau	132
12.9. Edge Impulse process [Tea21b]	133
13.1. Screenshot of the Colab UI	137

13.2. Selecting GPU under Hardware accelerator	138
13.3. Architecture of a single shot detector [BWL20]	140
13.4. Setting GPU and OpenCV to 1 in Makefile	140
13.5. Verification of CUDA version	141
13.6. darknet file with no extensions created after successful execution of make command	141
13.7. Obj and Test folders containing training and validation datasets respectively	143
13.8. Contents of the obj.names file	144
13.9. Contents of the obj.data file	144
13.10 Training the model in Google Colab	145
13.11 Training Failed after 2000 iterations due to Run-time limitations of Google Colab	146
13.12 Unable to access the GPU	146
14.1. Beispield Bilder aus dem Blumendatensatz	148
14.2. Einrichtung des Datensatzes	148
14.3. Neuskalierung und Datenerweiterung	149
14.4. Training des Modells	150
14.5. Prüfung des Modells	150
18.1. Arduino Portenta Arduino Store	162
19.1. Menu bar options	165
19.3. Arduino Setup Installation options	166
19.2. Menu buttons	166
19.4. Arduino Setup: Installation Folder	167
19.5. Arduino Sketch	167
20.2. Install packages for Arduino Portenta H7	169
20.1. Arduino Portenta H7 Connected to a laptop	170
20.3. Upload Basic Sketch	171
20.4. Compile Blink Sketch	171
20.5. Serial Blink Sketch	172
20.6. Serial Blink Sketch	172
20.7. Output LED color	173
21.1. Update Bootloader	175
21.2. Upload Bootloader sketch	176
21.3. Updating Bootloader	176
21.4. Face Filter output	177
21.5. Face tracking output	178
21.6. Find circles output	178
21.7. QR Code output	179
22.1. Output RGB LED blinking red , blue and green	183
24.1. Add Dataset	188
24.2. Dataset class Folders	189
24.3. export to Edge Impulse	189
24.4. Data Acquisition in Edge Impulse	190
24.5. Impulse Design	190
24.6. Save Parameters	191
24.7. Generate Features	191
24.8. Feature Explorer	191
24.9. Edit Neural Network Program	192

24.10 Model output	192
24.11 Deployment to the Arduino Portenta H7	193
24.12 Face detection output	193
24.13 Pen detection	194
24.14 Flowchart of the overall process	195
24.15 Sample Images from UTKFace Dataset	196
24.16 Arduino-CLI Installation	197
24.17 Node.js Installation	197
24.18 Connecting Arduino Portenta H7 with Edge Impulse	198
24.19 Connected Devices in the Edge Impulse	198
24.20 Labelling an Image	199
24.21 Create Impulse	199
24.22 Save parameters	200
24.23 Generate Features	200
24.24 Completed training	201
24.25 Testing the model	201
24.26 Live classification by using vision shield camera	202
 25.1. Sample Images from flowers dataset	204
25.2. Setting up the dataset	204
25.3. Rescaling and Data Augmentation	205
25.4. Training the model	205
25.5. Testing the model	206
 26.1. Optionen der Menüleiste	214
26.2. Serieller Blink-Sketch I	215
26.3. Serieller Blink-Sketch II	215
26.4. Ausgang LED-Farbe	216
26.5. Arduinos Vision Shield für den Portenta H7 Arduino Store	217
26.6. Arduinos Vision Shield für den Portenta H7 Arduino Store	218
26.7. Arduino Nicla Vision - Time of Flight sensor	219
 27.1. Flussdiagramm des Gesamtprozesses	224
27.2. Beschriftung eines Bildes	225
27.3. Impulse schaffen	226
27.4. Parameter speichern	226
27.5. Merkmale generieren	227
27.6. Abgeschlossenes Training	227
27.7. Prüfung des Modells	228
27.8. Live-Klassifizierung mit Hilfe der Vision-Shield-Kamera	228

List of Listings

3.1.	Building a Convolutional Neural Network (CNN) with Keras	63
4.1.	Loading a dataset with TensorFlow	66
4.2.	Load and normalise the data set Canadian Institute For Advanced Research (CIFAR)-10	68
4.3.	Load and normalise the data set CIFAR-100	68
4.4.	Loading the Fisher's Iris Data Set	70
4.5.	Description of Fisher's Iris Data Set	71
4.6.	Information of the data set Common Objects in Context (COCO) . .	73
4.7.	Metainformation of the dataset COCO	73
4.8.	Licence information of the data set COCO	74
4.9.	Image information of the dataset COCO	74
4.10.	Class information of the dataset COCO	75
4.11.	Substance information of the dataset COCO	75
4.12.	Annotations of the dataset COCO	76
12.1.	Skript zur Erstellung der Datei <code>train.txt</code>	129
12.2.	Einen Schnappschuss mit der Kamera machen	129
12.3.	Detektierung von Blobs	131
12.4.	Anzeigen von Blobs	131
12.5.	Initialisierung der LEDs	132
12.6.	Initialisierung der LEDs	132
12.7.	Initialisierung der LEDs	134
13.1.	Useful Functions to show images, upload or download the images from Cloud VM	142
13.2.	Script to generate train.txt file with paths to the training images . .	144
13.3.	Script to generate test.txt file with paths to the testing images . .	145
26.1.	Proximity-Sketch	221

Acronyms

AP Average Precision

API Application Programming Interface

CIFAR Canadian Institute For Advanced Research

CLI Command Line Interface

CNN Convolutional Neural Network

COCO Common Objects in Context

CPU Central Processing Unit

CSP Cross Stage Partial Network

CUDA Compute Unified Device Architecture

FP16 Floating Point 16-bit

FP32 Floating Point 32-bit

FPGA Field-Programmable Gate Array

fps frames per second

GPU Graphics Processing Unit

INT8 Integer 8-bit

IoT Internet of Things

KNN Künstliches Neuronales Netz

LED Light Emitting Diode

NN Neuronales Netz

PAN Path Aggregation Network

ReLU Rectified Linear Unit

ResNet Residual Neural Network

RLE Run-Length Encoding

SD-1 NIST Special Database 1

SD-3 NIST Special Database 3

SP Streaming Processor

VM Virtual Machine

VPU Video Processing Unit

YOLO You Only Look Once

Part I.

Machine Learning

1. Artificial intelligence - When are machines intelligent?

Answering the question is not easy. First, the term intelligence must be defined. Its definition is not unambiguous. For example, it is not defined at what point a person is considered intelligent. This raises the question of what belongs to intelligence. In psychology, intelligence includes, among other things, the ability to adapt to unknown and situations, but also to solve new problems.[FVP98]

1.1. What is Artificial Intelligence?

Consequently, the term artificial intelligence is also not clearly defined. Historically, the term was introduced by John McCarthy [McC+06]:

„The aim of AI is to develop machines that behave as if they had intelligence.“ - John McCarthy, 1955

According to this definition, simple machines and sensors are intelligent. They act in accordance with a programme. This means that all reactions of the system have been determined and fixed beforehand. If individual characteristics of humans are considered, then, according to machines, a great many machines can be considered intelligent; especially since machines have taken over activities that are wholly or partially performed by humans. A simple example is computer programs. They are many times more efficient in this area than humans. Hence the following definition [Ric83]:

„AI is the study of how to make computers do things which, at the moment, people do better.“ - Elaine Rich, 1983

This approach is clever in that it does not circumvent the definition of intelligence. On the other hand, it points out that it is a snapshot in time. This takes into account, that the human being is very adaptable. This adaptability distinguishes him in particular. Man is able to grasp new external influences and adapt to them. He can learn independently.

In the decade from 2010 to 202, many impressive results were achieved in the field of artificial intelligence. In 2011, the computer Watson defeated the human champion in the very popular guessing show „Jeopardy“ [Fer12]. Professional players of the board game Go were beaten by System AlphaGo for the first time in 2015 and 2016 [Wan+16]. Even bluffing in poker was successfully applied by an artificial intelligence called Libratus in 2017 [BS18]. In 2018, a computer program from the Chinese corporation Alibaba shows better reading comprehension than humans. The software, GPT-3, writes 2020 non-fiction text and poetry that looks like it was created by humans. Artificial intelligences have also become active in the art world; new songs by the Beatles have been composed and paintings created in the style of famous painters. These successes tempt one to conclude that computer software is superior to human intelligence. To clarify the difference, the terms weak and strong intelligence are introduced. Strong artificial intelligence is for all-encompassing software that can

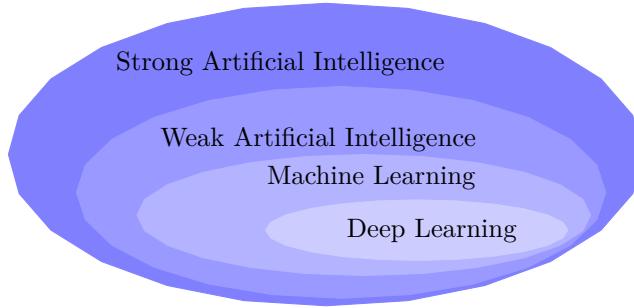


Figure 1.1.: relationship of artificial intelligence, machine learning and deep learning

adapt to different or new requirements. In contrast, weak artificial intelligence means that an algorithm has been developed to solve a specific task. They can solve complex and demanding tasks from very different fields as well as humans or better. The above list shows that weak AIs deliver stunning results. But strong artificial intelligence does not exist any more than a time machine. In the graphic 1.1 the relationships between strong, weak Artificial Intelligence, Machine Learning and Deep Learning are shown; according to this, Deep Learning is a subfield of Machine Learning and this in turn is a subfield of Artificial Intelligence.

1.2. Machinelles Lernen

Machine learning represents the ability of a machine to extract knowledge from available data and recognise patterns. If necessary, this pattern can then be applied to subsequent data sets. The way in which the learning process takes place can be divided into three categories:

Supervised learning uses training and test data for the learning process. The training data includes both input data (e.g. object metrics) and the desired result (e.g. classification of objects). The algorithm should then use the training data to find a function that maps the input data to the result. Here, the function is adapted independently by the algorithm during the learning process. Once a certain success rate has been achieved for the training data, the learning process is verified with the help of the test data. An example of this would be a clustering procedure in which the clusters are already known before the learning process begins.

Unsupervised learning uses only input data for the learning process, where the outcome is not yet known. Based on the characteristics of the input data, patterns are to be recognised in this. One application of unsupervised learning is the clustering of data where the individual clusters are not yet defined before the learning process.

Reinforcement learning is based on the reward principle for actions that have taken place. It starts in an initial state without information about the environment or about the effects of actions. An action then leads to a new state and provides a reward (positive or negative). This is done until a final condition occurs. The learning process can then be repeated to maximise the reward.

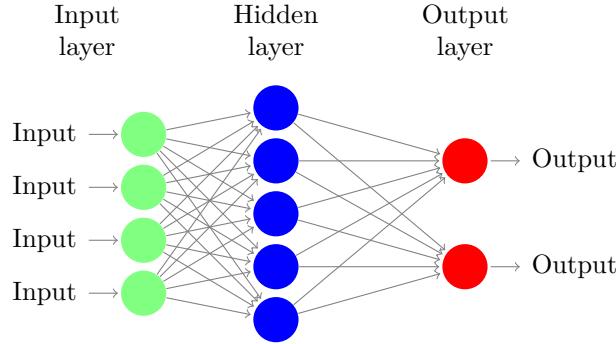


Figure 1.2.: Example of a KNN

1.3. Deep Learning

Deep Learning represents a sub-field of machine learning and uses a Neuronales Netz (NN) for the learning process. These represent a model of the human brain and the neuronal processes. The Künstliches Neuronales Netz (KNN) consists of nodes representing neurons. A distinction is made between three categories of neurons.

input neurons are the neurons that receive signals from the outside world. Here there is one neuron for each type of input (feature).

hidden neurons are neurons that represent the actual learning process.

output neurons are the neurons that give signals to the outside world. There is one neuron for each type of output (feature).

All neurons of a category are combined into a layer. Thus, in each neural network there is an input layer, see figure 1.2 green neurons, a hidden layer, in the figure 1.2 the blue neurons, and an output layer with the red neurons in the figure 1.2. If a KNN contains more than one hidden layer, it is called a deep neural network. The connections between the neurons in each layer are called synapses. These contain a weighting, which is multiplied by the signal of the start neuron. Thus, the individual signals are weighted. The weights are in turn adjusted during the learning process, based on functions.

1.4. Application

Due to the adaptability of artificial intelligences, they can be used in many different ways. In its study „Smartening up with Artificial Intelligence (AI)“, McKinsey described eight application scenarios in which artificial intelligence has particular potential [Mck].

- Autonomous Vehicles
- Predictive maintenance through better forecasting
- Collaborative robotics with perception of the environment (machine-machine interaction & human-machine interaction)
- Quality improvement through autonomous adaptation of machines to products to be processed

Published as a conference paper at ICLR 2015

$$\begin{array}{ccc}
 \text{Image of a panda} & + .007 \times & \text{Image of a gibbon} \\
 x & \text{sign}(\nabla_x J(\theta, x, y)) & = \\
 \text{"panda"} & \text{"nematode"} & \text{esign}(\nabla_x J(\theta, x, y)) \\
 & & \text{"gibbon"}
 \end{array}$$

Figure 1.3.: Overlay of the image of a panda with noise [GSS14]

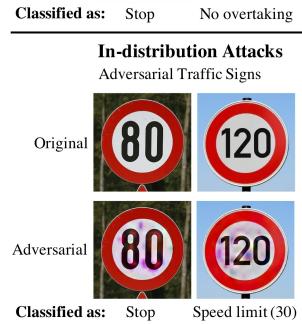


Figure 1.4.: Manipulation of traffic signs [Sit+18]

- Automated quality inspection Supply chain management through more accurate forecasting Research and development
- Automated support processes

1.5. Limits of Artificial Intelligence

Image recognition has made huge progress in recent years []. Nevertheless, limitations are also becoming apparent. In this context, the image of a panda from a paper by Ian Goodfellow and other researchers is shown [GSS14]. An algorithm identifies the panda with a probability of 57,7%. If the image is overlaid with a second image that a human perceives as noise, a gibbon is identified with a probability of 99,3, even though in a human's eyes the image has not changed. The situation is illustrated in Figure 1.3.

The example with the Panda is harmless. In contrast, the effects in road traffic can be devastating. Research shows that by simply manipulating road signs, artificial intelligence can make false interpretations. [Sit+18] Well-defined manipulations can be carried out by selectively altering an image. In the image 1.4, a sign marking a speed limit is transformed into a stop sign by adding spots.

Conversely, an optimal image can be determined from the neuronal networks that delivers a result with a probability close to 100%. This also produces images that

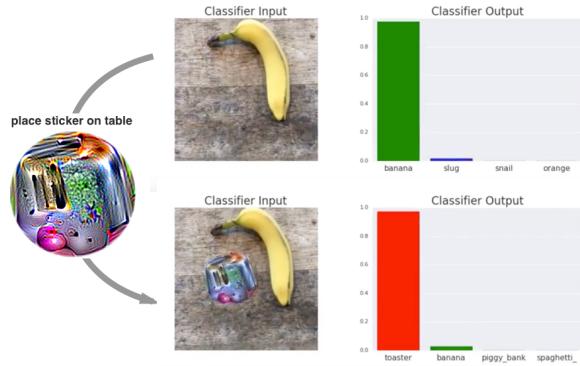


Figure 1.5.: Optimal image for the identification of a toaster [Sit+18]

the human eye could not recognise as such. A classic example is the toaster in the image 1.5.[Bro+17]

2. Processes for Knowledge Acquisition in Databases

2.1. Knowledge Discovery in Databases (KDD Process)

Knowledge discovery in databases, or KDD for short, is the decisive step in the examination of data. There are various possibilities for knowledge discovery. However, in order to achieve the goal efficiently and systematically, a suitable process is of immanent importance. In this section, the KDD process is considered in detail. In this process, knowledge discovery is divided into several steps. [Düs00]

2.1.1. Application Areas

The KDD process can basically be applied whenever large amounts of data are available from which knowledge can be extracted. The KDD process was presented in the article by Fayyad. [FPSS96] There, astronomy, investment, marketing, fraud detection, manufacturing processes and telecommunications are given as examples on areas of application. But the KDD process is not limited to these areas. Thus, [MR10] presents various applications in the fields of finance, medicine, detection of attacks on computer systems or customer relationship management. Although these examples exploit all elements of the KDD process in detail, application in these areas demonstrates the possible use of the entire KDD process with data mining as a core element. [PUC02] describes the application of the KDD process to data sets on tourist destinations, which can lead to insights into possibilities of increasing the attractiveness of individual destinations.

The literature mentioned shows the versatility of the KDD process. Basically, it can be used everywhere where sufficient data is available and a generalisation of certain structures is permissible.

2.1.2. Process Model

One difficulty regarding the KDD process is that „no general process model of Knowledge Discovery in Databases is currently available“ [Düs00]. There are several models that show the process steps starting from the data to the extracted knowledge. One of the first and according to [KM06] the most cited model is the one according to Fayyad [FPSS96]. Therefore, it will be presented in the following. According to [KM06], it is mainly the KDD process according to Fayyad and CRISP-DM that are also applied in the technical/engineering field, while many of the other models are rather aimed at areas in and around marketing. Therefore, both (Fayyad and CRISP-DM) will be described. For this purpose, a generic model and such a model designed for application with edge computers will be presented.

2.1.3. KDD-Prozess based on Fayyad

On an abstract level, the field of KDD is concerned with the development of methods and techniques to „make sense of data“ [FPSS96]. This requires going through several steps. The iterative nature of the process should be noted: Again and again, one can and must return to the previous steps and make corrections based on the knowledge

gained. It may be necessary to obtain further data or introduce other pre-processing steps. [Wro98]

The basic flow of the KDD process and the iteration loops included can be seen in Figure 2.1.

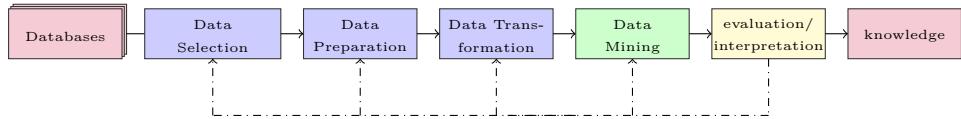


Figure 2.1.: KDD process according to [FPSS96] (own reduced representation based on [FPSS96])

At the beginning of the KDD process there is always the available database from which the knowledge is to be gained. From this, the data to be used must be selected from one or more databases and the data must then be processed and prepared. Only then can the data be used for knowledge extraction with the help of data mining methods. After evaluation and possibly several runs through the various steps, the knowledge gained can finally be extracted and used in the application.

Fayyad further divides this process into nine sub-steps, some of which are not directly represented in Figure 2.1. These nine steps are described below.

1. Domain Knowledge (Developing an Understanding of the Application Domain)

First, the domain of the application must be defined and understood. This includes the domain knowledge needed for the application to be able to perform the analysis in a way that leads to real added value. In addition, the precise objectives of the KDD process must be defined for the use case and the specific end user of the knowledge [FPSS96; KM06]. This step forms the basis for making the decisions necessary in the next steps. necessary decisions in the next steps [MR10].

2. Creating a Target Data Set

Next, a data set or a subset of a data set is selected. Single or multiple variables can be selected to narrow down the data from which knowledge is to be extracted. [FPSS96] This involves determining what data is available and possibly obtaining any data that is still missing. It is important that all necessary information is included. Deciding which variables are necessary must be based on domain knowledge. After selecting the variables, as many representatives of the variables as possible should become available to the algorithm for a good model. On the other hand, the collection, storage and processing of larger data sets is more costly and many variables can also lead to results that are difficult to interpret. Therefore, the relevance of returning to steps that have already been carried out when gaining knowledge should already be seen in this step.[MR10]

3. Data Cleaning and Preprocessing

This step aims to improve data reliability. This includes removing noise and outliers, but also dealing with missing data. If an attribute is not reliable enough, a data mining method can already be used at this point to improve the data.[MR10] Time series information and known changes must also be taken into account. [KM06] Data cleaning and preparation is the most time-consuming step in the KDD process. Its share of the time to be spent on the KDD process is estimated by many sources to be about 60 %. [KM06]

4. Data Reduction and Transformation (Data Reduction and Projection)

To further improve the data, useful features are sought to represent the data [FPSS96]. Possible methods include dimensionality reduction and transformation. The implementation of this step will vary greatly depending on the application and domain. As in the previous steps, the most effective transformation may emerge during the KDD process [MR10]. The importance of this step is made clearer by the application example in section 2.6.

5. Choosing the DM Task

After the preparation of the data has been completed, a suitable data mining method is selected. Which method is suitable depends largely on the objectives defined in the first step [FPSS96]. A basic distinction is made between descriptive methods, which aim at interpreting and understanding the data, and predictive methods, which provide a behavioural model that is able to predict certain variables for previously unseen data. For this purpose, regressions can be used on the one hand, and classification methods such as decision trees, support vector machines or neural networks on the other. Some of these models can also be used for descriptive statements. [MR10]

6. Choosing the DM Algorithm

In this step, the specific data mining algorithm to be used is selected. If, for example, it was decided in the previous step that a predictive method should be chosen, a choice must now be made between, for example, neural networks and decision trees, whereby the former provide more precise descriptions of the correlations, but the latter are easier to interpret. [MR10]

7. Data Mining

In the seventh step, data mining is finally applied. Often data mining and KDD are used synonymously, but in fact data mining is a sub-step of the KDD process. [FPSS96] This step may also have to be carried out several times, possibly even immediately one after the other, to optimise the hyperparameters. [MR10]

8. Interpreting the Results (Interpreting Mined Patterns)

After the data mining has been successfully completed, the results are evaluated and interpreted. The visualisation of found patterns or models can be helpful for this. In this step, the influence of the data preparation steps is reflected on and possibly returned to. It must also be evaluated whether the model found is plausible and usable. [FPSS96; MR10]

9. Applying the Results (Acting on Discovered Knowledge)

The discovered knowledge can now be used. The product of this can be documentation or reports that record the knowledge obtained so that it can be used by interested parties. Furthermore, the extracted knowledge can be integrated into another system, e.g. in the form of a prediction model. By applying it under real conditions, e.g. with dynamic instead of static data, deficits can become apparent that still need to be reworked. [FPSS96; MR10]

2.2. Generic Process Model

Based on the models of Fayyad et al, Cabena et al, Cios et al and Anand & Buchner and the CRISP-DM model, [KM06] describes a generic KDD model. This is structured very similarly to Fayyad's model described above and also runs through the steps iteratively. However, the steps are summarised differently, so that the model gets by with six instead of nine steps.

2.2.1. Application Domain Understanding

As with Fayyad, the first step is to understand the application and objective as a prerequisite for the following steps.

2.2.2. Understanding the Data (Data Understanding)

The general understanding of the data corresponds roughly to the second step according to Fayyad. It involves looking at the available data set and identifying opportunities and problems.

2.2.3. Data Preparation and Identification of DM Technology

This step combines steps three to six according to Fayyad. This makes sense insofar as the type of data preparation and the data mining method to be applied influence each other and a clear separation is therefore difficult.

2.2.4. Data Mining

The data mining step as the core of the KDD process remains unchanged.

2.2.5. Evaluation

The generic model also emphasises the iterative nature of the KDD process. Therefore, an evaluation of the obtained results is indispensable.

2.2.6. Knowledge Deployment (Knowledge Consolidation and Deployment)

This step also hardly differs from the last process step according to Fayyad. The different models mainly differ in the way the discovered knowledge is used, which is due to the different application domains. While the models geared towards business applications mainly focus on reports and visualisation, the technical applications have a stronger emphasis on dynamic uses of created models.

2.3. CRISP-DM

While the first KDD models seem to be strongly oriented towards business applications, the CRoss-Industry Standard Process for Data Mining (CRISP-DM) is intended to be applicable to all industries and thus also to technical problems. CRISP-DM was developed in 1996 as part of a merger of the companies DaimlerChrysler, SPSS and NCR. The aim was to embed data mining in an industry-, tool- and application-independent process. The methodology is described with a hierarchical process model consisting of four abstraction levels: phases, generic tasks, specialised tasks and process instance. While the phases are always run through in the same way and the generic tasks are the same, which are described in more detail by the specialised tasks, the last level deals with the individual scope of application. [Cha+00]

The CRISP-DM process model contains the life cycle of a data mining project as shown in Figure 2.2. The phases shown are largely consistent with the phases described in the generic model. Ultimately, the KDD process is embedded in CRISP-DM. As mentioned, individual generic tasks are assigned to each phase, which must be processed in accordance with the specific task. In the phase „Data preparation“ this includes data selection, cleansing, construction, integration and formatting. [Cha+00]

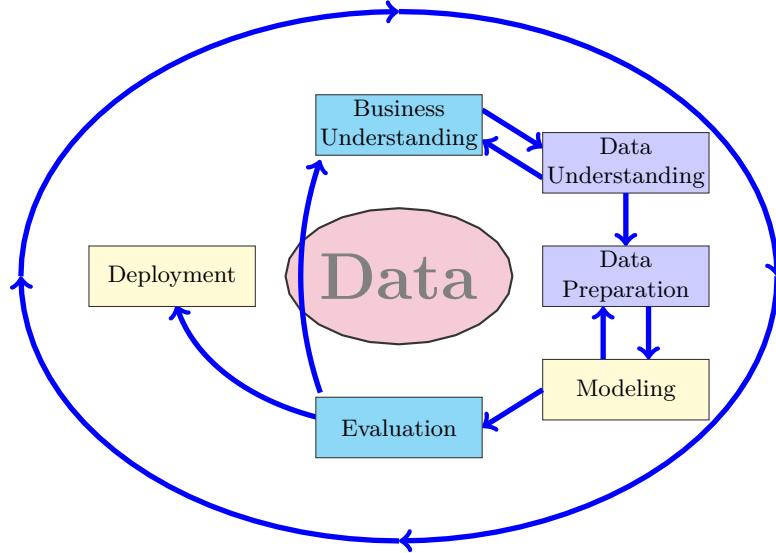


Figure 2.2.: Life cycle of a data mining project in CRISP-DM according to [Cha+00]

This model also emphasises the importance of returning to past phases during the process. In Figure 2.2 only the most important and frequent connections between the phases are shown. The outer circle represents the cyclical nature of the data mining process itself. [Cha+00]

2.4. Application-Oriented Process Models

The models shown remain very abstract and little related to actual technical applications in a technical environment. In order to represent the actual processes when using different systems for knowledge generation and for use in the field, the following representations can be helpful.

For the description of artificial intelligence projects with the Jetson Nano, a modified KDD process model is assumed, which largely corresponds to the model according to Fayyad, but still contains some minor changes. This modified process model can be seen in Figure 2.3.

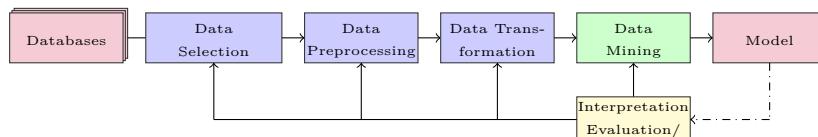


Figure 2.3.: Modified model of the KDD process based on the model according to [FPSS96]

The lines marking the iterative loops are solid and not dashed to emphasise the absolute necessity of these iterative correction steps. However, the feedback from the

model is shown dashed because in this process model it is assumed that the knowledge gained is used in the form of a model on an edge computer, here the Jetson Nano. This complicates direct feedback and corrections.

In Figure 2.4 the distribution of the steps on different levels as well as localities and systems becomes clear. This illustration refers to the application in a production environment. It would be conceivable, for example, to create a system for process control that detects defective workpieces with the help of a camera. The database would be collected with this camera. The use of the data, including the steps explained in the previous sections, takes place in an office outside of production. However, the results found are used again in the control system in production, where they are exposed to new data. This new data can be added to the database(s) and possibly a feedback between the results in the field to a review and correction of the models taking place in the office can also be realised.

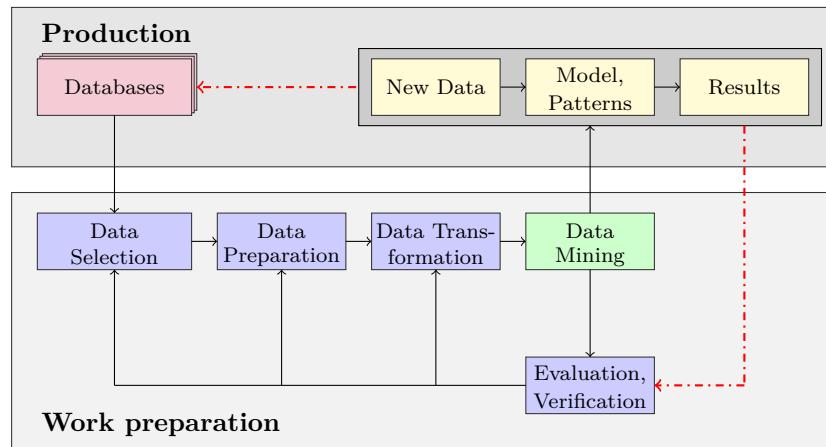


Figure 2.4.: KDD process in the industrial environment

2.5. Role of the User

Brachman and Anand [BA94] criticise that the usual definitions of KDD are too focused on the resulting information and do not adequately reflect the complexity of the process. According to them, the role of the human being must be emphasised more, even if in the long run there is a move towards fully autonomous integrations of the KDD process. Front-end requirements must be recognised and reconciled, and data preparation, analysis and evaluation, as well as monitoring of the individual steps through visualisation, also require domain knowledge on the part of the user and his or her intervention. To support this, the authors recommend a corresponding working environment (Knowledge Discovery Support Environment) that allows easy data exchange between the different work steps as well as facilitates the reuse of already created structures. In addition, a system could support the user in the selection of suitable methodologies.

2.6. KDD Process using the Example of Image Classification with an Edge Computer

To look at the process in relation to a tangible application, the development of an image classification with an Edge Computer as a KDD process will be highlighted as an example.

2.6.1. Database

As a data basis for an image classification, a sufficient number of images including their correct classification in terms of the application, the label, must be available. There are various image databases that are freely accessible, including ImageNet, MS-COCO and CIFAR-10, for example. Of course, only such databases come into question that contain objects and features that the model is to recognise later. [Den+09; Den12; SW16; Aga18a]

File formats and colour depth as well as the number of colour channels should also already be taken into account when searching for suitable databases, as a (lossless) transformation to the required file formats is not possible in every case or the database might contain less information than desired. Which file formats, colour depths and colour channels to use depends on the desired application, the training platform used and the algorithm used.

2.6.2. Data Selection

The images to be used can now be selected from the databases that have been identified, are available and can be used for the application. For example, a database could be used in its entirety, but only a part of such a database can be used or several databases can be combined. It is extremely important that all classes that are to be recognised are also present in the selected data. How many images are selected depends on the planned scope of the training and the test, which has a significant influence on the training time, but possibly also on the performance of the model obtained.

2.6.3. Data Preparation

Data preprocessing is usually primarily concerned with data cleaning. This mainly includes the removal of outliers. In terms of images, these outliers are mainly noisy or poorly resolved images. These should be removed or processed with suitable algorithms. When using images from the well-known open-source image databases, such as ImageNet [Den+09], image quality should not be a problem.

Furthermore, the images must be converted into the format supported by the respective platform. It should be taken into account that compressed file formats cannot always be decompressed without loss.

2.6.4. Data Transformation

Finally, the images must be transformed into a representation that can be used in data mining. It may be necessary to adjust the colour depth per channel or the number of channels. An RGB image may need to be transformed into a greyscale image. It is also common to normalise the images, for example to mean 0 and standard deviation 1.

The algorithm chosen for the data mining phase also determines the input format of the data. Applied to images, this means which file format to use. In addition to the file format, however, the colour resolution determines the number of pixels. In this phase the images are adjusted accordingly.

2.6.5. Data Mining

In the data mining step, the actual model building takes place. First, the algorithm must be chosen or the architecture defined and the model built accordingly. If deep learning algorithms are used, the architecture must be defined. With the definition of the architecture, the hyperparameters are determined. Then the algorithm can be trained with the selected images. During training, the weights are optimised. Their

adjustment is done on the basis of the training images. After a given number of training images, the validation images are used to test the accuracy of the current model. After several epochs, i.e. several runs of all training images, the model can be tested.

2.6.6. Evaluation/Verification

In the first instance, the interpretation and evaluation of the trained model is done repeatedly during the training with the help of the validation data set. In the second instance, the model is tested directly after training with the help of the test images that are still unknown to the model. If the results are satisfactory, the model can be integrated into the application. In this instance, too, it is important to test the performance of the model in order to be able to draw conclusions and make corrections. In the dynamic application, other framework conditions prevail, so that drawing conclusions about necessary corrections in the previous steps may require great domain knowledge.

The interpretation of the found model is not trivial for neural networks, especially for complex networks like CNN. It may not be possible to extract direct correlations that could be transferred to other applications.

2.6.7. Model

The trained model can finally be integrated into the intended application. While the data preparation and training take place on a computer that is generally equipped with high computing power, the model is intended to be used on an edge computer. For example, real-time object recognition could be performed with a camera connected to the Edge Computer based on the training with the static images. Transferring the model to the edge computer may require converting the model to a different format.

2.7. Data Analysis and Data Mining Methods at a Glance

The following is an overview of some data mining techniques.

2.7.1. Definition Data Analysis

Data analysis attempts to extract information from large amounts of data with the help of statistical methods and to visualise it afterwards. The complex field of data analysis can be divided into the areas of descriptive, inferential, exploratory and confirmatory data analysis. In descriptive data analysis, the information of the individual data, which was taken from a total survey, for example, is condensed and presented in such a way that the essentials become clear. If only the data of a sample survey (partial survey) is the basis, the focus of the data analysis rests on the transfer of the sample findings to the population. This is referred to as inferential data analysis. Explorative data analysis is concerned with processing the available amount of data in such a way that structures in the data as well as correlations between them can be revealed and highlighted to a particular degree. In contrast, the aim of confirmatory data analysis is to verify correlations.

2.7.2. Statistical Data Analysis

Statistical data analysis usually begins with calculations of the mean and standard deviation (or variance). It also involves checking the data for normal distribution.

Hypothesis Test

For the application of the hypothesis test, two hypotheses are always formulated: a null hypothesis H_0 , which mostly implies that the presumed data structure does not exist, and the alternative hypothesis H_1 or H_A , which mostly implies that the presumed data structure exists. The hypothesis test refers to the assumption of a true null hypothesis [ABF11]. It must be proven that the null hypothesis is false. First, values such as the T-value ("The T-value is the quotient of the arithmetic mean of the differences between two variables being compared and the estimate of the standard error for this mean in the population." [ABF11]) or the F-value is calculated. These values are compared with a critical value adjusted to the situation. If this calculated value is smaller than the critical value, the null hypothesis is considered true. In this case, the P-value, i.e. the probability with which the result related to the null hypothesis occurs, is also only slightly smaller than 0.05. If, on the other hand, the P-value is very small, the null hypothesis is rejected. It is considered relatively certain that the null hypothesis is false; however, it is not known which hypothesis is correct. Conversely, if the null hypothesis is not rejected, it cannot be concluded that it is correct [ABF11]. In this case, the result cannot be interpreted.

Normal Distribution, P-Test, T-Test

The P-value (significance value) indicates how extreme the value of the test statistic calculated on the basis of the data collected is. It also indicates how likely it is to obtain such a sample result or an even more extreme one if the null hypothesis is true. Since the value is a probability value, it takes on values between zero and one. The P-value therefore indicates how extreme the result is: the smaller the P-value, the more the result speaks against the null hypothesis. Usually, a significance level α is set before the test and the null hypothesis is then rejected if the P-value is less than or equal to α [ABF11]. In order to be able to make decisions whether the null hypothesis should be rejected or retained, fixed limits have been established at about 5 %, 1 % or 0.1%. If the null hypothesis is rejected, the result is called statistically significant. The size of the P-value does not indicate the size of the true effect.

The T-test refers to a group of hypothesis tests with a T-distributed test size [ABF11], but is often differentiated into one-sample T-test and two-sample T-test. The prerequisite for carrying out the T-test is the normal distribution of the population of data to be examined. Furthermore, there must be a sufficiently large sample size. If the data are not normally distributed, the T-test cannot be used and the principle of the U-test is applied. The one-sample T-test uses the mean value of a sample to test whether the mean value of the population differs from a given target value. The classic T-test (two-sample T-test), on the other hand, tests whether the mean values of two independent samples behave like the mean values of two populations. It is assumed that both samples come from populations with the same variance. The variance is the square of the standard deviation. The greater the variance, the flatter the normal distribution curve. If two sample sizes are compared, the weighted variance must also be determined. Here, the larger sample has the more decisive influence on the result.

ANOVA (Analysis of Variance)

„Analysis of variance, usually called ANOVA in German, primarily looks for differences between groups and tests whether dividing the data into different groups reduces unexplained variability.“[Dor13] Prerequisites for analysis of variance are normally distributed values in each group, approximate equality of standard deviations and independence of the measured values from each other. It is tested whether the mean values of several groups differ. In the simplest case, the null hypothesis is: The mean values of all groups are equal. Then the following alternative hypothesis results: Not

all mean values are equal. The test variables of the procedure are used to test whether the variance between the groups is greater than the variance within the groups. This makes it possible to determine whether the grouping is meaningful or not, or whether the groups differ significantly or not. In its simplest form, the analysis of variance is an alternative to the T-test. The result is the same as with the T-test, „because the results of a one-way analysis of variance (one-way ANOVA) and a T-test are identical if the two samples have the same variance.“ [Dor13]

2.7.3. Overview of Data Mining Methods and Procedures

Typical methods of data mining are:

- cluster analysis: grouping of objects based on similarities,
- classification: elements are assigned to existing classes,
- association analysis: identification of correlations and dependencies in the data,
- regression analysis: identification of relationships between variables,
- outlier detection: identification of unusual data sets,
- correlation analysis: examines the relationship between two variables,
- summary: transformation of the data set into a more compact description without significant loss of information.

Here, outlier detection as well as cluster analysis belong to the observation problems; classification and regression analysis belong to the prediction problems.

Cluster Analysis & Classification

Cluster analysis is used to reveal similarity structures in large data sets with the aim of identifying new groups in the data. The similarity groups found can be graph-theoretical, hierarchical, partitioning or optimising. The objects assigned to a cluster should be as homogeneous as possible, the objects assigned to different clusters should be very heterogeneous [JL07]. In addition, several characteristics can be used in parallel when forming clusters. The cluster analysis is carried out in the following steps.

At the beginning, each object is considered as a single cluster. Then, the two individual objects that are most similar to each other are merged. The union reduces the number of clusters by one. Afterwards, all distances of the individual objects are calculated again and the two objects with the smallest distance are combined into a new cluster. This could theoretically be repeated until all objects are united in a single cluster, a so-called megacluster. However, for the analysis of the data, it is much more significant to determine the clustering that seems to make the most sense. The cluster number is determined by looking at the variance within and between the groups. It is determined which clustering seems to make the most sense, as no termination condition is specified for the function itself. As a rule, different cluster divisions are advantageous.

Prerequisites for the application of cluster analysis are metrically scaled characteristics, which can simply be included in the analysis; ordinally and nominally scaled characteristics must be scaled as dummy variables [JL07]. Characteristics that are scaled in different dimensions can lead to a bias in the results. These values must be standardised before carrying out a cluster analysis, for example by a Z-transformation [JL07]. Furthermore, the characteristics should not correlate with each other.

The distance between two individual objects is determined by the distance measure. The larger the measure, the more dissimilar the objects are. The various cluster

methods [JL07] are used to determine the distance between two clusters or a cluster and a single object. Classification, on the other hand, assigns data to already existing groups.

Association Analysis & Regression Analysis

„Association analysis generates rules that describe the relationships between the elements (items) that occur in the records of a dataset.“[GC06] These dependencies are represented in the form If item A , then item B or $A \rightarrow B$. An item is the expression of an item. An item is the expression of an attribute value of a data set. An example of a simple rule would be: If a customer buys beer, then 70 per cent of the time he will also buy chips. These relationships are not assumed as hypotheses, but are to be discovered from the data with the association analysis. Only after a conspicuous pattern has been found is it investigated whether it is really a dependency and, if so, association rules are established for it. The parameters of association rules are support, confidence and lift. The greater the support value, the more relevant the rule.

Regression analysis is the analytical procedure for calculating a regression in the form of a regression line or function. The regression indicates which directional linear relationship exists between two or more variables [GC06]. The coefficient of determination (R^2) expresses how well the regression line reflects the relationship between the independent and dependent variable. R^2 lies between 0 and 1, whereby the value $R^2 = 1$ would mean that every observed data point lies directly on the regression line. By determining a regression function, no statement can be made about the significance of a correlation. The significance of the regression is determined by the F-test.

At the beginning of the regression analysis is the preparation of the data. Missing data are omitted or filled in, data are transformed and the interactions (in linear regression) are taken into account. By means of mathematical procedures, a function f is determined so that the residuals e become minimal (residuals: difference between a regression line and the measured values [Kä11]). Model validation, i.e. checking whether the model is a good description of the correlation, includes the

- residual analysis,
- overfitting,
- examining the data for outliers and influential data points, and
- multicollinearity of the independent variables.

The validated model can be used to forecast values of y given values of x . To estimate the uncertainty of the prediction, a confidence interval is often given in addition to the predicted y -value.

Outlier Detection

Outliers are measured values or findings that are inconsistent with the rest of the data, for example, by having unusual attribute values or not meeting expectations. Expectation is most often the range of dispersion around the expected value in which most measured values are found. „Robust bounds for detecting outliers for many distribution types can also be derived based on quartiles and quartile distance.“[SH06] Values of exploratory studies that lie further than 1.5 times the quartile distance outside this interval are called outliers. Particularly high outliers are shown separately in the boxplot.

For example, the Local Outlier Factor procedure searches for objects that have a density that deviates significantly from their neighbours, then this is referred to as density-based outlier detection. Identified outliers are then usually verified manually and removed from the data set, as they can worsen the results of other procedures. Before deciding in favour of removing values, it is therefore still necessary to check in each case what data loss occurs when deleting or labelling the missing values. If the number of available data sets falls below the level necessary to proceed, the removal of the outliers should be avoided.

Correlation Analysis

An important task of data analysis is the analysis of the correlation between individual characteristics. The strength of the connection between two quantitative characteristics is called correlation in descriptive statistics and inferential statistics and can be differentiated into linear and non-linear correlation. In multivariate data sets, the correlation coefficient is additionally calculated for each pair of variables.[Gö] „For correlation analysis, mainly methods of classical, multivariate, robust and exploratory statistics are used, but also various non-linear regression methods whose approximation errors can be used as correlation measures.“[Run15] A prerequisite for performing correlation analysis is the normal distribution of the data under investigation.

Summary as a Method of Data Mining

By transforming a data set into a more compact description of its information, summarisation ensures a lossless representation of essential information. The summary can be textual, visual or combined.

3. Neural Networks

While computers are superior to humans in many areas, there are nevertheless tasks that a human can intuitively due to their intelligence and ability to learn, while transferring these abilities to the computer is a great challenge. A popular approach to the development of artificial intelligence lies in artificial neural networks. These are information-processing systems whose structure and mode of operation are modelled on the nervous system, especially the brains of animals and humans.

Neuronal networks consist - according to their name - of neurons, which are interconnected and influence each other. Like their biological counterparts, the modelled neurons can, if sufficiently stimulated by one or more input signals. The biological neuron can be compared to a capacitor that is charged by small electrical voltage pulses from other neurons. If the voltage thus reached exceeds a threshold value, the neuron itself transmits a voltage pulse to connected neurons. This principle is used in artificial neural networks by mathematical models. Figure 3.1 shows the first stage of abstraction from biological neural networks to mathematical models. [Ert16]

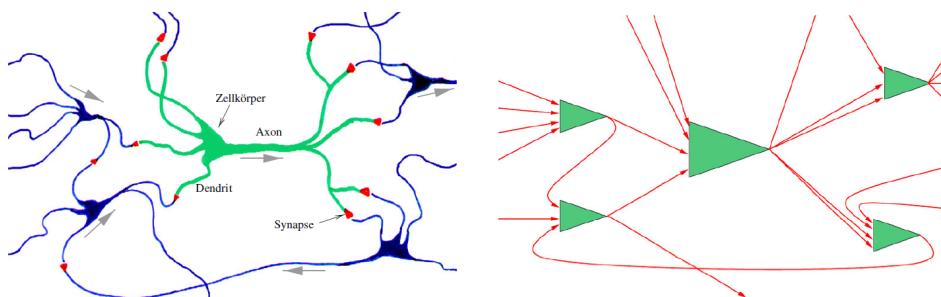


Figure 3.1.: Biological (left) and abstracted (right) model of neural networks.
Source:[Ert16]

In 1958, Frank Rosenblatt presented the simplest form of a neural network. This model, presented as a single-layer perceptron, contains a layer of input neurons, each connected to an output neuron. [Dör18] If several neurons are connected in layers in series, we are talking about the multilayer perceptron. The output neuron acts as an input signal for the following neuron, until a final output takes place at the output layer.

3.1. Mathematical Modelling

The birth of neural network AI can be seen in the 1943 publication „A logical calculus of the ideas immanent in nervous activity“[MP43] by McCulloch and Pitts. In their description, neurons can have either the output value 0 for inactivity or 1 for activity. This is referred to as binary neural networks. (Sometimes the activity level -1 for inactivity is also used, cf. [Ert16]). With McCulloch and Pitts, the input values for a neuron are added together and this is set to activity level 1 if this sum exceeds a

predefined threshold θ . One therefore also speaks of „threshold elements“ [Kru+15], a representation of how this works, taking into account different weights w_1 and w_2 , can be seen in Figure ???. Several neurons are connected in layers to form networks in order to be able to map complex relationships. As will be described in the following, different weights (section ??), as well as activation and output functions (section 3.1.3) with continuous (instead of binary) output values and different architectures play a role in more complex neural networks. With the mathematical modelling of the neuron, however, McCulloch and Pitts laid the foundation for the further development of neural networks, although the computer technology of the time was not yet sufficiently powerful for the simulation of simple brain structures. [Ert16]

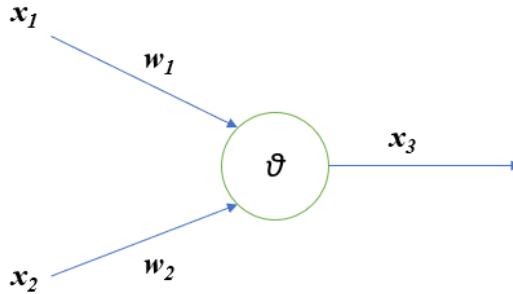


Figure 3.2.: Representation of a neuron as a simple threshold element

3.1.1. Weights

In order to be able to map different relationships between the activations of the neurons, the information flows between the neurons are weighted differently. In addition, constant input values can provide a neuron-specific bias, the weighting of which may also need to be determined. [Moe18] In the representation of the network as a directed graph, the weights are usually specified at the edges. Mathematically, the representation as a matrix is common. An example of the graph representation and the equivalent weight matrix can be seen in Figure ???. The weights are often named by convention with first the index of the neuron receiving the signal as input and then the index of the sending neuron.

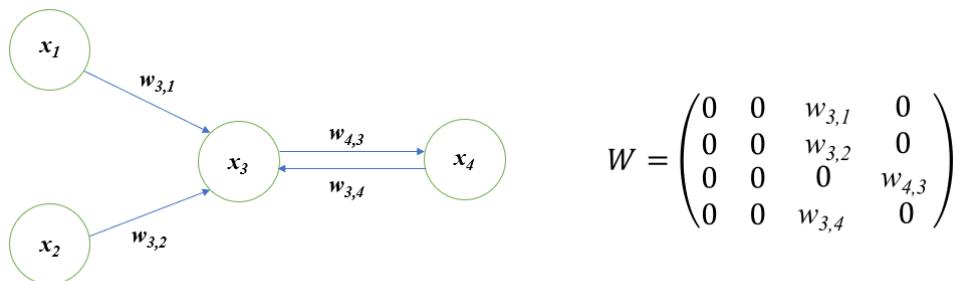


Figure 3.3.: Graph with weighted edges and equivalent weight matrix

At the beginning, the weights are set to random values. Usually values between -1 and 1 are chosen, although the values resulting from the training can also lie outside

this interval. [Moe18] For certain(!) training methods, initialisation with zeros is also conceivable. [KK07]

3.1.2. Bias

Bias is the neuron-specific bias that can be interpreted as an additional input with a constant value. The bias value can be modelled both explicitly or as a weighting of an additional constant input value and can strengthen or weaken the result of a neuron. [Zie15]

3.1.3. Activation and Output Function

Which value a neuron passes on depends on the incoming signals, their weighting, the activation function and the output function. From the incoming output signals of the connected neurons (and possibly the feedback of the own output signal of the considered neuron itself), the activation function f calculates the activity level of the neuron, taking into account the weighting of the signals. [Kru+15] The output of the neuron x_i is calculated from the w_{ij} weighted incoming signals of the n neurons x_j with $j = 1 \dots n$.

$$x_i = f\left(\sum_{j=1}^n w_{ij} x_j\right)$$

In the simplest case, the activation function is linear, then the signals are added weighted and possibly scaled with a constant factor. More complex, non-linear relationships can only be modelled using linear activation functions in multi-layer networks. [KK07] Non-linear functions facilitate generalisation and adaptation to diverse data and are therefore widely used. [Gup20]

The output value of the neuron is then determined from the activity level of the neuron determined in this way, with the help of the output function. In most cases, the identity is used as the output function [Bec18a], so that often only the activation function is addressed and the output function is neglected.

Function ReLu

The Rectified Linear Unit function is considered the most commonly used activation function. It is defined as $R(z) = \max(0, z)$, i.e. all negative values become zero and all values above zero remain unchanged. However, this can also lead to problems, as negative activations are ignored and neurons are deactivated. This is what the Leaky ReLu function is designed to correct, using a low slope < 1 in the negative range (see figure 3.4). Both functions and their derivatives are monotonic. This is important so that the activation cannot decrease with increasing input values. [Gup20; Sha18]

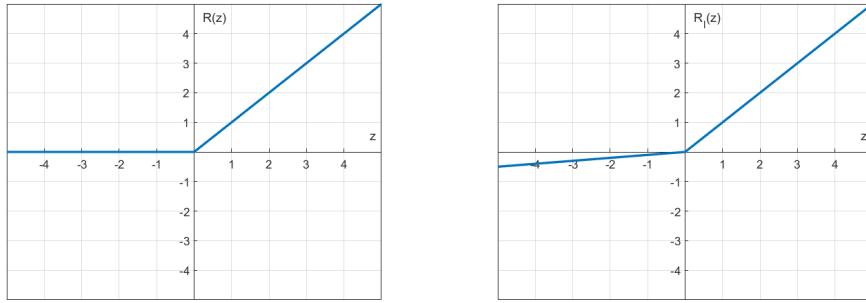


Figure 3.4.: ReLu function (left) and Leaky ReLu

Function Sigmoid

The sigmoid function transforms the input values into the range [0,1], see figure 3.5. The function for this is $S(z) = \frac{1}{1+e^{-z}}$. Due to the range of values, the function is well suited for the representation of probabilities. It is always positive, so that the output cannot have a weakening influence on the subsequent neurons. The function is differentiable and monotonic, but its derivative is not monotonic. [Gup20; Sha18]

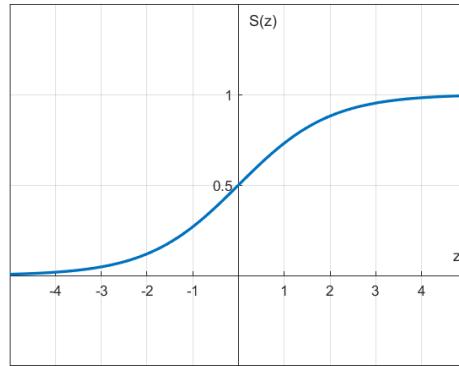


Figure 3.5.: Sigmoid function

Function Softmax

The softmax function is often described as a superposition of several sigmoid functions. It is mainly used in the output layer to reflect the probabilities for the different categories. The mathematical expression for the softmax function is [Gup20]:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$

Function tanh

The tangent hyperbolic function is a non-linear function in the value range from -1 to 1 . It is similar to the sigmoid function with the major difference of point symmetry at the origin. The function is defined as:

$$\tanh(z) = \frac{2}{1+e^{-2z}} - 1$$

The negative inputs are strongly negative and the zero inputs are close to the zero point of the diagram. Negative outputs can occur, which can also negatively affect the weighted sum of a neuron of the following layer. This function is particularly well suited to the differentiation of two classes. The function is differentiable, monotonic and the derivative is non-monotonic. [Gup20] The figure 3.6 shows the function tangent hyperbolicus in the definition range from -6 to 6 .

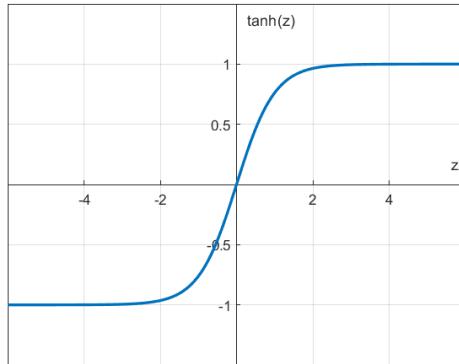


Figure 3.6.: Funktion Tangens Hyperbolicus

3.2. Training

The great strength of neural networks lies in their ability to learn. In neural networks, the learning process consists of adapting the weights with the aim of outputting the desired output value(s) in the output layer in the case of corresponding input signals in the input layer. For this purpose, the weights are first initialised randomly and then adjusted according to the learning rules.

3.2.1. Propagation

Propagation refers to the process in which information is sent via the input layer into the network and passes through it to the output layer. If the information only moves from the input to the output layer, it is referred to as a forward network or forward propagation. At the beginning, the input is copied into the input layer and then forwarded to the first hidden layer. What information this layer passes on to the next depends on the weights and the activation functions. Finally, the output can be read from the output layer. [Kri08]

3.2.2. Error

Error is the deviation between the actual network output and the desired network output after the input data has been propagated through the entire neural network. [Kri08]

3.2.3. Delta Rule

The basis of the learning process is the delta rule. For each input data point in the learning phase, the difference between the actual and expected (correct) output is determined. Then the weights are changed so that the error becomes smaller. To determine the direction in which the weights must be changed, the derivative of the

error function is used (gradient descent). The strength of the adjustment is determined by the learning rate. This process is repeated iteratively until the termination criterion, e.g. a sufficiently small error, is met.[KK07]

$$\Delta w_{ij} = \varepsilon(x_i - l_i)f'(\sum_j w_{ij}x_j)$$

This rule can only be applied directly to the weights between the output layer and the layer before it. For the layers before it, the error must be propagated backwards through the network, since the desired outputs for the hidden networks are not given. [Kru+15]

3.2.4. Error Backpropagation

In the backpropagation procedure, the change in the values of the hidden neurons is recursively calculated from the change in the neurons in the higher layer. For this, the network output must first be calculated (forward propagation). The error obtained is used in backward propagation to adjust the weights from layer to layer. This process is applied to all training data until the termination criterion is met. [Ert16]

This procedure is well established for training neural networks. Nevertheless, there are some difficulties that need to be taken into account. For example, a local minimum of the error can be found, which is not left anymore, so that the global minimum cannot be found. This problem can be mitigated by running the procedure with different starting values. [Kru+15]

3.2.5. Problem of Overfitting

If a neural network is repeatedly trained with known data sets, the network may learn the data by heart instead of developing an abstract concept. develop. This condition is called overfitting. For this reason, not all data should be used for training. It makes sense to withhold a validation set and a test set. [Bec18c]

3.2.6. Epochs, Batches and Steps

An epoch is a complete run of all input data, with the measurement of the error, and backpropagation to adjust the weights. For large data sets, the input data can be divided into groups of equal size, called batches. In this way, a neural network can be trained more quickly. It is important that the values within each batch are normally distributed. For example, if a data set of 2,000 images is divided into batches of 10 images each, the epoch will run through 200 steps. Alternatively, the number of steps can be specified and the batch size adjusted accordingly.[Bec18b]

3.2.7. Dropout

Dropout disables some neurons by multiplying a defined percentage of neurons by zero during forward propagation. These neurons have no longer have any influence on subsequent activations. Deactivation occurs on a random basis per run. If a neuron is imprinted on a feature and fails, the surrounding neurons learn to deal with this problem. This increases the generalisation of the network, it learns more abstract concepts. A disadvantage is the increase in training time, because the parameter feedback is noisy. At the end of the training, all neurons are reactivated. [Bec18b]

3.2.8. Correct Classification Rate and Loss Function

The correct classification rate measures the ability of the network to assign data points to the correct class. This contrasts with the loss function. [Cho18]

Both are often documented during the training process to allow conclusions to be drawn about the training process and to identify problems such as overfitting or too few runs.

3.2.9. Training Data, Validation Data and Test Set

The training data is actively used to teach the network the features of the classifications. The validation data is fed to the network at the end of each epoch to test how accurately the network performs. Even if the validation set is not actively used for training, some information flows back into the network. For this reason, a test set is used that is not related to the training of the neural network and only has the purpose of checking the theoretical accuracy of the network. [Bec18c]

3.2.10. Transfer Learning

Transfer learning means the transfer of the results of a finished neural network to a new task. Layers can be kept constant and only the output layer can be retrained, or several layers can be retrained on the basis of the current training status. If several or all layers are retrained, the weights of the trained network are used as the starting point. Which strategy should be used depends on two factors:

- similarity of the data
- size of the new data set

If the data sets are highly similar, transfer learning is a good way to improve a neural network. Similarities exist, for example, in a trained network for the recognition of dog breeds, which would also be suitable for the recognition of cat breeds. However, this network would be unsuitable for the recognition of different car models, as there would be too many deviations in the basic structures. As far as the size of the data set is concerned, transfer learning may be particularly suitable for small data sets, as otherwise overfitting would quickly occur. [Bec18c]

3.2.11. Weight Imprinting

Retraining all the weights in a model for transfer learning can be a time consuming process. In actuality, it's only the last layer of the network that decides the final classification, so only retraining the last set of weights could achieve the desired result. Weight imprinting is one such method developed by engineers at Google. Their proposed imprinting method is to directly set the final layer weights for new classes from the embeddings of training exemplars. Consider a single training sample x_+ from a novel class, their method computes the embedding $\Phi(x_+)$ and uses it to set a new column in the weight matrix for the new class, i.e. $w_x = \Phi(x_+)$. Figure 1 illustrates this idea of extending the final layer weight matrix of a trained classifier by imprinting additional columns for new categories. Intuitively, one can think of the imprinting operation as remembering the semantic embeddings of low-shot examples as the templates for new classes. Figure ?? illustrates the change of the decision boundaries after a new weight column is imprinted. The underlying assumption is that test examples from new classes are closer to the corresponding training examples, even if only one or a few are observed, than to instances of other classes in the embedding space.

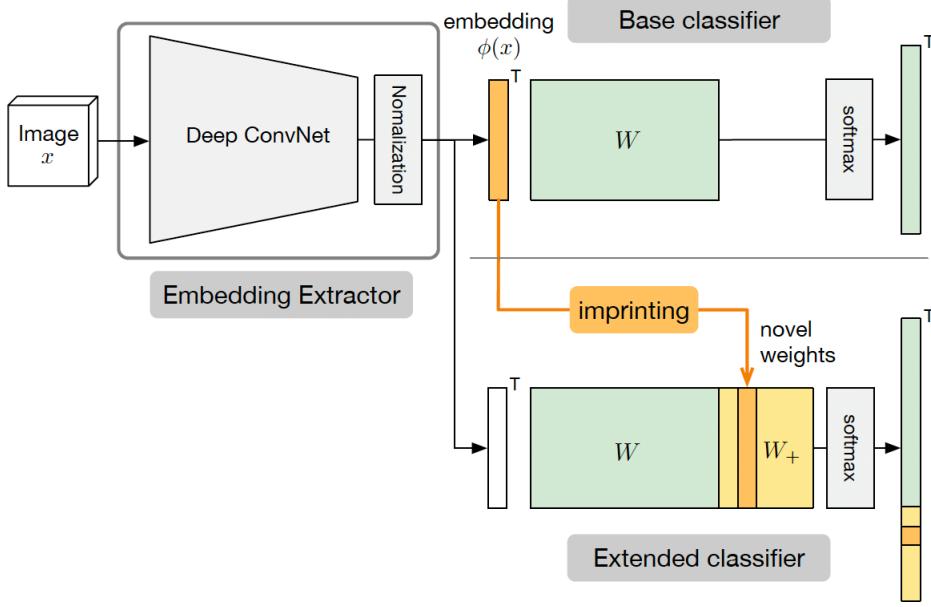


Figure 3.7.: Illustration of the weight imprinting process [QBL18]

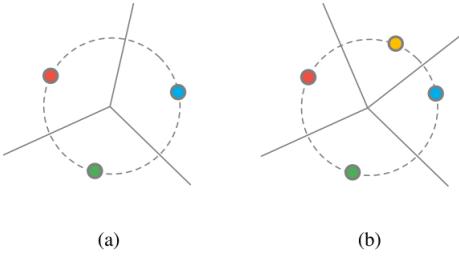


Figure 3.8.: Illustration of imprinting in the normalized embedding space. (a) Before imprinting, the decision boundaries are determined by the trained weights. (b) With imprinting, the embedding of an example (the yellow point) from a novel class defines a new region. [QBL18]

Average Embedding

If $n > 1$ examples $(x_+^{(i)})_{i=1}^n$ are available for a new class, new weights are computed by averaging the normalized embeddings $\tilde{w}_x = \frac{1}{n} \sum_{i=1}^n \Phi(x_+^{(i)})$ and re-normalizing the resulting vector to unit length $w_+ = \frac{\tilde{w}_+}{\|\tilde{w}_+\|}$. In practice, the averaging operation can also be applied to the embeddings computed from the randomly augmented versions of the original low-shot training examples.

Fine-tuning

Since the model architecture has the same differentiable form as ordinary ConvNet classifiers, a finetuning step can be applied after new weights are imprinted. The average embedding strategy assumes that examples from each novel class have a unimodal distribution in the embedding space. This may not hold for every novel class since the learned embedding space could be biased towards features that are salient and discriminative among base classes. However, fine-tuning (using backpropagation to further optimize the network weights) should move the embedding space towards having unimodal distribution for the new class. [QBL18]

3.3. Convolutional Neural Networks

Complex tasks such as pattern and object recognition with high-dimensional data sets lead to convergence problems and high computation times with conventional approaches. Such tasks can be solved with deep learning algorithms. To this class belong CNN. They can solve tasks with less preprocessing than is usual for other classification algorithms. Moreover, the filters used do not have to be designed by the user, but are learned by the network itself during training. [Sah18]

According to [SJM18], one of the first successful implementations of CNNs was the recognition of handwritten digits in 1998, which is described in the paper „Gradient-Based Learning Applied to Document Recognition“[LeC+98]. In the meantime, CNNs have been able to prove themselves in many application areas. The typical application for CNNs is processing data that has a known network topology. For example, they can recognise objects in images or patterns in time series. [Nam17] Besides image classification, object recognition and tracking, text, speech and gesture recognition are also typical applications [Gu+18] describes. But CNNs are also used in industry for defect detection. [LP20]

In general, CNNs are composed of the building blocks Convolutional Layer, Pooling Layer and Fully Connected Layer, where each building block is usually used more than once. These building blocks are described below. The primary assumption is the use of CNNs for image classification.

3.3.1. Convolutional Layer

The main component of CNN are filters that are applied to the input data/images in the Convolutional Layer. As input, the Convolutional Layer receives an n-dimensional tensor, for example an image with three colour channels, and applies several kernels to it. By convolution of tensor and kernel, the image is filtered. The results of the convolution with the different filters are stacked so that the result is again a tensor. [Mic19]

The operation of convolution between two matrices or tensors means the addition of the products of the multiplication of the corresponding elements of both matrices or tensors. Since the convolution in the Convolutional Layer usually takes place between a tensor with many entries and a small kernel, the convolution must be carried out separately for individual parts of the input tensor. Usually, one starts with the first entries at the top left and, metaphorically speaking, shifts the kernel after the first convolution relative to the input tensor by a certain number of columns and calculates the convolution again. When one has reached the right edge, the kernel is shifted in the row and the process is repeated. The value called stride indicates by how many columns and rows the kernel is shifted relative to the input tensor. In addition, an activation function is usually applied for each calculated pixel. [Mic19]

In this process, a tensor with dimensions $h_1 \times w_1 \times d_1$ and kernels each with dimension $h_2 \times w_2 \times d_1$, which are stacked to form dimensions $h_2 \times w_2 \times d_2$, becomes an output tensor with dimensions $h_3 \times w_3 \times d_2$. [Aru18]

Here h_3 and w_3 depend on the height and width of the input tensor as well as the filters and the chosen stride, while d_2 is given by the number of filters.

If the size of the input tensor, the kernel and the stride do not match accordingly, pixels at the edge of the image cannot be reached according to the procedure described above. Therefore, sometimes additional pixels are added. This is called padding, or zero-padding, because the added pixels are usually filled with the value zero. [Mic19] The parameters learned in a CNN are the filters themselves. For k filters of dimension $m \times n$, considering one biasterm per filter, a total of $k \cdot m \cdot n + k$ parameters must be learned. It is worth noting that this number is independent of the size of the input image, whereas in conventional feed-forward networks the number of weights depends on the size of the input.[Mic19]

The convolutional layer does not necessarily have to be the first layer, but can also receive preprocessed data as input. Usually, multiple Convolutional Layers are used.[Mic19]

The filters of the first Convolutional Layer detect low-level features such as colours, edges and curves, while higher layers detect more abstract features. By stacking multiple Convolutional and Pooling Layers, higher level feature representations can be progressively extracted. Moreover, the dimensionality is also reduced continuously, which reduces the computational cost. [Gu+18; Sah18]

3.3.2. Pooling Layer

Usually, each Convolutional Layer is followed by a Pooling Layer. Sometimes both are referred to together as one layer because no weights are learned in the pooling layer. [Mic19]

It reduces the resolution of the previous feature maps by compressing them, thus reducing complexity and hence computational cost. It also makes the network more robust to small variations in the learned features and focuses on the most important patterns. [Nam17]

There are different types of pooling. The most common are maximum and average pooling. In maximum pooling, only the maximum value is stored from each section of the tensor covered by the kernel. With average pooling, on the other hand, the average value is determined for each section. [Sah18]

The pooling layer does not introduce any new learnable parameters, but it does introduce two new hyperparameters, namely the dimension of the kernel and the stride. [Mic19]

3.3.3. Fully Connected Layer

After several convolutional and pooling layers, the image is flattened into a single vector. This output is fed into a neural feed-forward network (one neuron per entry in the vector). This is fully interconnected and thus able to extract the non-linear correlations from the extracted features and thus arrive at a classification.[Sah18]

The ReLu function is usually used as the activation function. In the output layer, the softmax function is used to reflect the probabilities of each class to be assigned. [Aru18]

3.3.4. Hyperparameter

When building and/or training a CNN, several parameters need to be considered. The architecture of the network and the order of the layers need to be determined. There are new approaches to different convolutional methods from which to choose. In addition, the size and number of filters and the stride must be defined. The pooling procedure must also be defined, as well as the activation functions in the fully networked layers. Likewise, it must be defined how many neurons the input layer should have (this influences the possible image size) and how many layers and neurons per layer the Fully Connected Layer should be composed of.

Many considerations must be made regarding the database. The complexity of the network to be trained depends mainly on whether only one input channel (black and white or greyscale) is used or three (RGB). If only one colour channel is to be used to reduce complexity, the images must be prepared accordingly. Likewise, the size (number of pixels, height and width) of the images may need to be adjusted to fit the architecture used. Normalising the images can also be helpful.

Since the images are presented to the neural network as data arrays, the file format does not play a direct role. Indirectly, however, the different image qualities of the various formats can influence the performance of the networks. For example, networks

trained with high-resolution images may produce poor results when they are asked to classify compressed images. [DK16] Differences can also arise from the different compressions, so different file formats should not be mixed.

3.3.5. Fine-tuning

Fine-tuning allows specific layers from the convolution layer to be trained specifically. With a small number of training images, the mesh can be easily transferred to a new task. This is much faster than designing a neural network from scratch. [Cho18]

3.3.6. Feature Extraction

In feature extraction, convolutional layers (convolutional layer and pooling layer) are extracted from a trained CNN and a new classifier, i.e. a fully connected layer with new classifications, is added. Two problems arise in this process:

- The representation only contains information about the probability of occurrence of the class trained in the basic model.
- The fully connected layer contains no information about where the object is located.

Should the position of an object in the image matter, the fully connected layers are largely useless, as the later layers only generate abstract concepts. [Cho18]

3.3.7. AlexNet

The most popular CNNs for object detection and classification from image data are AlexNet, GoogleNet and ResNet50. [SJM18]

Gu et al. [Gu+18] refer to the development of AlexNet in 2012 as a breakthrough in large-scale image classification. Nevertheless, its architecture is only complex enough to keep its operation comprehensible. Therefore, it seems a good choice for first attempts at your own training.

AlexNet consists of five Convolutional Layers and three Fully Connected Layers. The number and size of the filters as well as their stride are shown in table 3.1 for the sake of clarity. The first and second convolutional layers are each followed by a response normalisation layer, also known as batch normalisation. This additional layer is intended to mitigate the effect of unstable gradients through normalisation. After the response normalisation layers as well as after the fifth convolutional layer, a max-pooling layer follows. Overlapping pooling was chosen as the pooling method. This means that the individual sections to which pooling is applied overlap. The pooling area measures 3x3 and the stride 2 pixels. ReLu is applied to the output of each Convolutional Layer and each of the Fully Connected Layers. Each Fully Connected Layer consists of 4096 neurons. The output layer uses the softmax function to map the probabilities for the different categories via the output neurons. [KSH12a; Ala08]

Convolutional Layer	Anzahl Kernel	Dimension	Stride
1	96	$11 \times 11 \times 3$	4
2	256	$5 \times 5 \times 48$	1
3	384	$3 \times 3 \times 256$	1
4	384	$3 \times 3 \times 192$	1
5	256	$3 \times 3 \times 192$	1

Table 3.1.: Convolutional Layer Specifications

In total, AlexNet consists of 650000 neurons. About 60 million parameters need to be trained. [KSH12a]

This puts it about in the middle compared to the number of learnable parameters in other successful CNNs.

The input images for AlexNet need to be cropped to size 227×227 . Three colour channels are used (see table 3.1), so no conversion to greyscale is necessary, but one usually normalises the data to mean 0 and standard deviation 1.[Ala08]

3.3.8. YOLO

YOLO (You Only Look Once) is a state-of-the-art real-time object detection system that performs classification and localisation (with bounding box) much faster than conventional systems.[Red21]

Detection (in the sense of classification AND localisation) of objects is more complex than mere classification. In conventional systems, the image is divided into many regions, each of which is classified to determine the regions that contain the detected object. [Cha17]

In YOLO, on the other hand, the entire image is given once to a single neural network, which decomposes the image itself into individual regions, determining bounding boxes and classification probabilities for each of these regions themselves, and weighting the bounding boxes with the probabilities.

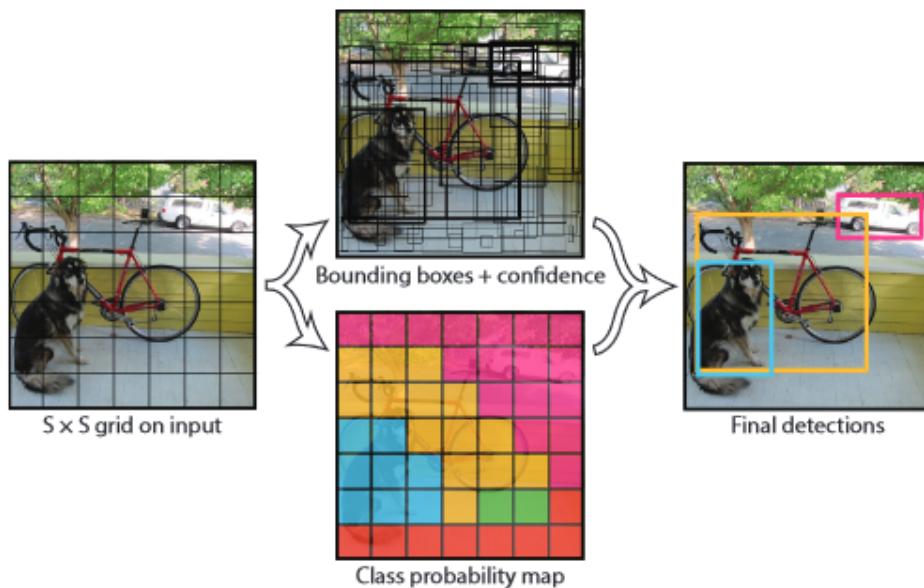


Figure 3.9.: The YOLO system

The network architecture of YOLO is based on GoogLeNet. It consists of 24 convolutional layers followed by two fully networked layers. The first 20 convolutional layers are pre-trained with the ImageNet dataset over a period of one week. The images in ImageNet have a resolution of 224×224 , the final YOLO network takes images with a resolution of 448×448 as input. Also, the training always takes place with the entire images, without prior subdivision. [Red+15]

The advantage of YOLO lies primarily in its speed, which allows a video stream to be processed with a latency of less than 25 ms. This makes YOLO $1000 \times$ faster than R-CNN and $100 \times$ faster than Fast R-CNN. Since the entire image is presented to the

network and not just individual parts of it, contextual information can be taken into account. For example, fewer errors due to interpretation of parts of the background as objects occur less frequently than with other systems. In addition, what is learned by the system can be better transferred to other areas, so that objects learned from photographs, for example, can also be recognised better in artistic representations than with other systems. [Red+15; Red21]

The YOLO code as well as the code used for the training are open source.

3.4. Images and Image Classification with a CNN

The classification of images is divided into two areas: image classification and object classification. Object classification attempts to identify one or more objects. The identification is done by assigning them to predefined classes. As a rule, the results are given with probabilities. Image classification attempts to identify a suitable description for the image; in this case, the image is recorded and interpreted in its entirety. [ZJ17] Several challenges exist when classifying images. If one uses images, one is processing larger amount of data. An image with 200×200 pixels has 40,000 pixels. If it is a colour image with three colour channels, one processes 120,000 attributes. Furthermore, it is desirable that the result of the classification is independent of affine mappings. That is, the result is independent of scaling or rotation of the image. If only a section of the image is examined, the corresponding result should still be delivered. The mapping ?? represents an image with corresponding operations. The classic „Lena“ of image processing is used as the basis of the demonstration [Mun96]. This is an image of Lena Soderberg from 1972 that is available in the USC-SIPI Image Database [Web97].



Figure 3.10.: Original image, scaling, rotation and cropping of the image

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	0	0	0	0	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	0	0	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1

10 × 10 Pixel

Figure 3.11.: Structure of a black and white image with 10×10 pixels

Resolution	=	width × height
VGA	=	640×480
HD	=	1280×720
Full HD	=	1920×1080
4K	=	3840×2160

Table 3.2.: Standard resolutions of camera images

3.4.1. Representation of Images

For a computer programme, an image is a matrix of numbers. The type of matrix is determined by the resolution, the number of colours and the colour depth. The resolution is the pair of values of how many elements, called pixels, the matrix has in height and in width. A simple example is shown in the image ???. The image has a resolution of 10 pixels in width and in height. The values can be either 0 or 1; in this illustration the value 0 represents the colour black, 0 the colour white.

Typical resolutions for camera images are recorded in the table???. A pixel is defined by the number of colours and the colour depth. The colour depth indicates the range of values taken from the individual elements. These are usually integer values, specified in bits. Thus, an 8-bit value corresponds to a number between 0 and 255. A pixel can now contain such a numerical value for each colour. Often, however, a separate matrix is used for each colour. If one works with the RGB representation, one has the colours red, green and blue. The colour impression is obtained by superimposing the three images. The situation is shown in the figure ???.

Often greyscales are used instead of colour images. For this purpose, 256 greyscales are often used. When converting a colour image with three colours, the three matrices are added weighted. It has been shown that equal weighting of the colour channels is

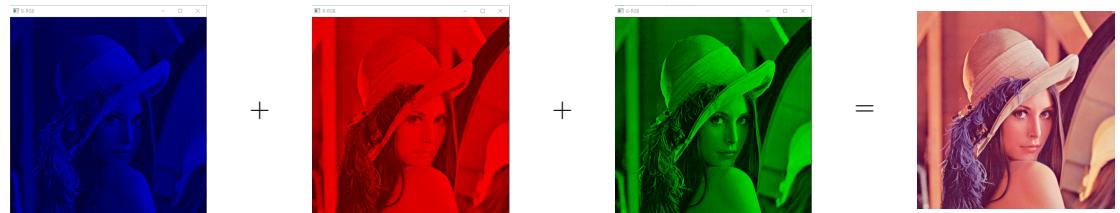


Figure 3.12.: Image with 3 colour channels

inappropriate. The OpenCV library for processing images uses the following weighting [Tea20a]:

$$0.299 \cdot \text{Red} + 0.587 \cdot \text{Green} + 0.114 \cdot \text{Blue}$$

The conversion can be applied to the image „Lena“; this is in the figure ???. Other weights are used depending on the application. The above weights are empirically determined and are proportional to the sensitivity of the human eye to each of the trichromatic colours. The image ?? has a colour depth of 8-bit. Other colour depths can also be used. A black and white image has the lowest colour depth.



Figure 3.13.: Greyscale with a value range from 0 to 255



Figure 3.14.: Black and white image, image with 4 grey levels and with 256 grey levels.

However, the quality of the image does not only depend on the parameters mentioned. The sharpness of the image also contributes to it. However, there is unfortunately no simple definition here, but it is often shaped by subjective perception. On the other hand, it is possible to influence image sharpness by mathematical operations. Blurring of an image can be achieved by blurring. Each pixel value is thereby represented by the average of the pixel values in a small window centred on the pixel.

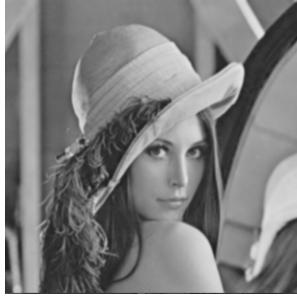


Figure 3.15.: Softened image with a 5×5 filter



Figure 3.16.: Edge detection on an image

The resolution of the image affects the amount of computation required. For an image with a resolution of 674×1200 and a window size of 101×101 , the new pixel value must be calculated separately for all three colour channels. In total this results in

$$674 \cdot 1200 \cdot 101 \approx 8.2 \cdot 10^9$$

operations. An example is shown in the figure ?? with a filter size 5×5 .

Another possibility of image processing is edge detection. With edge detection, the goal is to identify regions of the image where intensities of colour differ greatly. This is used to delineate objects and for object recognition. An example is given in Figure ??.

3.5. Convolution Neural Network

One focus for machine learning is the interpretation of images. Two cases must be distinguished here

1. Object detection
2. Image classification.

When detecting objects, an image is examined to see if one or more objects can be identified. If an object is detected, the coordinates of the surrounding rectangle and the class or classes of the object are returned with a respective probability. With the segmentation, instead of the coordinates of the surrounding rectangle, a polygon course is returned that surrounds the object. In contrast, when classifying an image, a description of the image is searched for. In both cases CNN are commonly used and are described in the following section.

A CNN expects an image as input information. If images do not contain colour information, they are represented as matrices whose number of columns and rows correspond to the number of pixels in width and height. In the case that an image

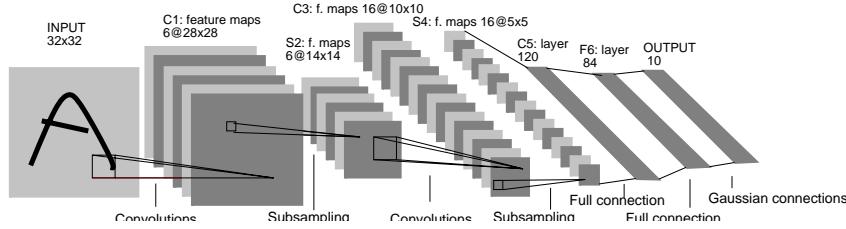


Figure 3.17.: Struktur der CNN-Architektur gemäß LeCun et.al. [LeC+98]

Layer	Feature Map	Size	Kernel Size	Stride	Activation function
Input	Image	1	32×32	-	-
1	Convolution	6	28×28	5×5	1
2	Average Pooling	6	14×14	2×2	2
3	Convolution	16	10×10	5×5	1
4	Average Pooling	16	5×5	2×2	2
5	Convolution	120	1×1	5×5	1
6	Neural Network	-	84	-	tanh
Ausgabe	Neural Network	-	10	-	softmax

Table 3.3.: Structural elements of LeNet

contains colour information, such a matrix is created for each colour. In this case, one speaks of colour channel or simply of channelindexchannel.

CNN is a commonly used shift-invariant method for extracting adaptive features. Its success has contributed to the popularity of machine learning. In its history, its architectures have undergone a great evolution. Some well-known architectures are now described.

3.5.1. CNN Architectures

3.5.2. LeNet

In 1998, the CNN architecture LeNet was the first CNN architecture to use back-propagation for practical applications. It thus transported Deep Learning from theory to practice. LeNet was used for handwritten digit recognition and was able to outperform all other existing methods. The architecture was very simple with only 5 layers consisting of 5×5 convolutions and 2×2 max-pooling, but paved the way for better and more complex models. [LeC+98] The listing of elements in Table 3.17 contains several points that will be described later.

3.5.3. AlexNet

The AlexNet architecture was the first to use CNN models implemented on Graphics Processing Unit (GPU)s. In 2012, this really connected the growing computing power with Deep Learning. AlexNet is significantly deeper and more complex than LeNet. They also started using ReLU activations instead of sigmoid or tanh, which helped train better models. AlexNet not only won the 2012 ImageNet classification competition,

but beat the runner-up by a margin that suddenly made non-neural models almost obsolete. [KSH12a]

3.5.4. InceptionNet

The architecture InceptionNet used the increased possibilities of the hardware. The architecture was again much deeper and richer in parameters than the existing models. To deal with the problem of training deeper models, they used the idea of multiple auxiliary classifiers present between the model to prevent the gradient from breaking. One of their ideas was to use kernels of different sizes in parallel, thus increasing the width of the model instead of the depth. This allows such an architecture to extract both larger and smaller features simultaneously. [Sze+14]

3.5.5. VGG

Many architectures tried to achieve better results by using larger convolution kernels. For example, the architecture AlexNet uses cores of size 11×11 , among others. The architecture VGG took the path of using several cores of size 3×3 in succession and more non-linearities in the form of activation functions. This again improved the results. [Zha+15; SZ15]

3.5.6. Residual Neural Network (ResNet)

The deeper the architectures became, the more often the problem occurred during training that the amount of the gradient became too small. The backpropagation then aborted and the parameters could not be determined. In the Residual Neural Network (ResNet) architecture, to prevent the problem, residual or shortcut links were introduced to create alternative paths for the gradient to skip the middle layers and directly reach the initial layers. In this way, the authors were able to train extremely deep models that previously did not perform well. It is now common practice to have residual connections in modern CNN architectures. [He+16]

3.5.7. MobileNet & MobileNetV2

The MobileNet architecture is designed for edge computers and smart phones that are limited in memory and computing power. This has opened up a large field of new applications. This is in contrast to the previous strategy of increasing the size of architectures by introducing separable convolutions. In simple terms, a two-dimensional convolutional kernel is split into two separate convolutions, namely the depth convolution, which is responsible for collecting the spatial information for each individual channel, and the point convolution, which is responsible for the interaction between the different channels. Later, MobileNetV2 was also introduced with residual connections and other optimisations in the architecture to further reduce the model size. [How+; San+18]

3.5.8. EfficientNet

The architecture is the result of considering that the various architectures to date, focus on either performance or computational efficiency. The authors of [TL20] claimed that these two problems can be solved by similar architectures. They proposed a common CNN skeleton architecture and three parameters, namely the width, the depth and the resolution. The width of the model refers to the number of channels present in the different layers, the depth refers to the number of layers in the model and the resolution refers to the size of the input image to the model. They claimed that by keeping all these parameters small, one can create a competitive yet computationally

Model	Size	Accuracy	Parameter	Depth
VGG16	528 MB	71,3%	138.357.544	23
VGG19	549 MB	71,3%	143.667.240	26
ResNet-50	98 MB	74,9%	25.636.712	50
ResNet-101	171 MB	76,4%	44.707.176	101
ResNet-152	232 MB	76,6%	60.419.944	152
InceptionV3	92 MB	77,9%	23.851.784	159
InceptionResNetV2	215 MB	80,3%	55.873.736	572
MobileNet	16 MB	70,4%	4.253.864	88
MobileNetV2	14 MB	71,3%	3.538.984	88

Table 3.4.: Characteristics of different CNN architectures

efficient CNN model. On the other hand, by increasing the value of these parameters, one can create a model that is designed for accuracy.

3.5.9. General Structure

In the description of the architectures, some elements necessary for the construction were mentioned. All building blocks can be listed as follows:

- Convolutional layer (convolution)
- Activation function
- Pooling
- Flattening
- Neural network

Each component is characterised by further variables. The combination and configuration of the building blocks are summarised in the so-called hyperparameters. By specifying the hyperparameter, the architecture and the number of parameters is determined. For example, in a neural network, the hyperparameters are the number of layers, the number of neurons per layer and the determination of the activation functions. If the number of all neurons is now determined, this results in the number of parameters that are determined during the training.

In the table 3.4 some architectures are compared. The table indicates the number of parameters to be trained. If an image is classified, the models give different possibilities and assign a probability to them. The accuracy now indicates the percentage with which the image matches the prediction with the highest probability. For this purpose, the models were trained and validated with the ImageNet.

The individual building blocks are now described below.

3.5.10. Input and Output

When a model receives an image as input, fields with pixel values are transferred. The height, the width and the number of colour channels determine the size of the fields. The elements of the fields then contain values between zero and 255. This value is the intensity of the pixel. The model then determines probabilities of how to classify an image based on these fields. For example, the result could be that 81% of the image is a turtle, 11% is a gun and 8% is a chocolate bar.

$$A * C = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} * \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix} = \sum_{i=1}^3 \sum_{j=1}^3 a_{ij} \cdot c_{ij}$$

Figure 3.18.: Definition of the convolution of two matrices

3.5.11. Convolution Layer

In a convolution layer, the input data is modified by one or more filters. In image processing, this is the application of a defined set of filters to two-dimensional data. The size of this filter is defined by the so-called kernel: a 3×3 kernel is represented by a 3×3 matrix. In order to be able to apply the filter, a range of pixel values is defined that has the same size; the pixel values are then also stored in a matrix. By convolution of the matrix, as shown in the equation ??, a value is assigned to the matrix and the filter.

To apply a filter, matrices must therefore be systematically selected from the pixel values. Since the dimension of a kernel is defined by an odd number $2n + 1$, a surrounding matrix can be assigned to each pixel value that has a distance greater than n from the edge. In the figure 3.19 such a situation is shown. The filter, labelled C and coloured purple, has a dimension of 3×3 . In the matrix A , the field of pixel values, a region is coloured red. In the middle is the pixel from the fifth column and the second row. The filter is now applied to this area of the matrix A . The result is then entered into the matrix $A * C$. In this case the value is 4 and marked green.

A filter is now moved step by step over the data. The filter can be applied to any pixel value that is far enough from the edge, or individual pixels can be skipped. This is defined by the step size, also called stride. With a step size of one, every pixel is used. To reduce the output size, stride sizes larger than one can be used; it is also possible to set different stride sizes for width and height. If a step size of 2 is specified in both directions, the amount of data has been reduced to 25%. [DV16]

In the figure 3.19, the input matrix is a 7×7 matrix. Since a step size of one is applied here, the result is a 5×5 matrix.

Often an activation function is applied to each result.

0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	1	0	0	0	0
1	1	0	0	0	0	0

*

1	0	1
0	1	0
1	0	1

=

1	4	3	4	1
1	2	4	3	3
1	2	3	4	1
1	3	3	1	1
3	3	1	1	0

A * **C** = **$A * C$**

Figure 3.19.: Convolution layer with 3×3 kernel and stride $(1, 1)$

When defining the filters, make sure that the filter is defined for each channel. To capture the features in an image, the values in the filter must match the structure of the original pixel values of the image. Basically, if there is a shape in the input image that generally resembles the curve that this filter represents, then the convolution will result in a large value.

As a rule, several filters are used in parallel. The result is so-called feature maps. Further convolution layers are then applied to this result, which in turn generate feature maps.

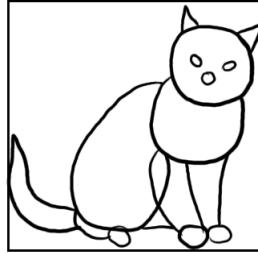


Figure 3.20.: Beispielanwendung für einen Filter und ein Ausschnitt

Feature Identification

A convolution is defined by filters. Each filter can be interpreted as an identifier of features. Features are, for example, edges, colours or even curves. For example, the 7×7 filter is considered:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

This filter can also be visualised as images:

To demonstrate the effect of the filter, a sample image 3.20 is considered:

In the example image 3.20 a section is marked to which the filter is applied.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 30 \\ 0 & 0 & 0 & 0 & 80 & 80 & 80 \\ 0 & 0 & 0 & 30 & 80 & 0 & 0 \\ 0 & 0 & 0 & 80 & 80 & 0 & 0 \\ 0 & 0 & 0 & 80 & 80 & 0 & 0 \\ 0 & 0 & 0 & 80 & 80 & 0 & 0 \\ 0 & 0 & 0 & 80 & 80 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 30 \\ 0 & 0 & 0 & 0 & 80 & 80 & 80 \\ 0 & 0 & 0 & 30 & 80 & 0 & 0 \\ 0 & 0 & 0 & 80 & 80 & 0 & 0 \\ 0 & 0 & 0 & 80 & 80 & 0 & 0 \\ 0 & 0 & 0 & 80 & 80 & 0 & 0 \\ 0 & 0 & 0 & 80 & 80 & 0 & 0 \end{pmatrix}$$

The basic property of filters is clarified. If there is a shape in the input image that generally resembles the filter, then large values result. A probability is then assigned via the activation function. Various filters are then defined and applied in the convolution layer. The result is then the feature map. The figure 3.21 shows some examples of concrete visualisations of filters of the first convolutional layer of a trained network.

Edge Detection

Suitable filters can also be defined for edge detection. The filter

$$\begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

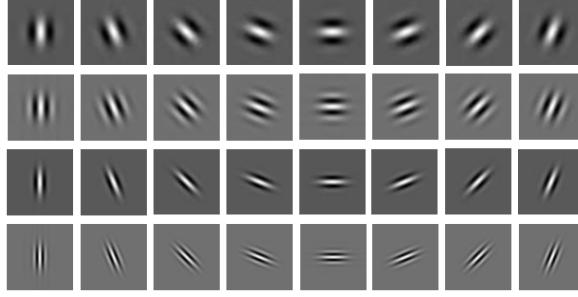


Figure 3.21.: Visualisation of filters of a trained network [LXW19; Sha+16]

$$\begin{pmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 3.22.: Detection of vertical and horizontal edges

can be used for a vertical edge detection, whereas the filter

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

can be used for a horizontal one. Both filters are now applied to a vertical edge.

Edges can be present in any arrangement in images, edge detection must take this into account. That is, if a CNN is trained, the filters evolve based on the given images.

3.5.12. Padding

In the previous representation, filters could not be applied to boundary values. On the one hand, this means that information is not used, on the other hand, the size of the images is reduced each time the folding is applied. To prevent this, padding is applied. For a filter of dimension $(2n + 1) \times (2m + 1)$, the input image is padded with an additional border of size n and m . The additional values are each zero occupied.

3.5.13. Activation function

The activation function is applied to the respective output of a layer and determines how the output value should be interpreted by applying a defined function. Here, the respective input values, as well as a bias, are included in the calculation and determine whether the neuron is triggered [Nwa+18]. In this example, the Rectified Linear Unit (ReLU) function is used for this: for input values less than or equal to

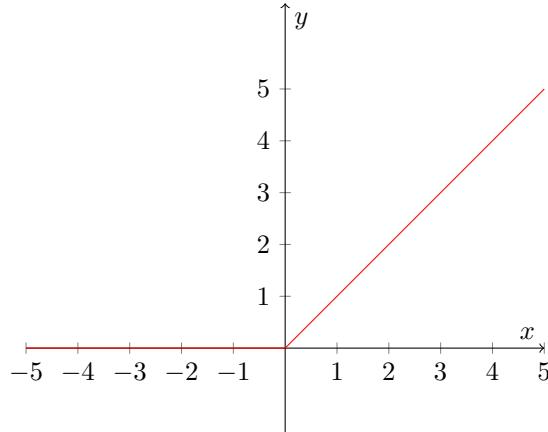


Figure 3.23.: Representation of the ReLU activation function

$$\text{ReLU}(A) = \begin{pmatrix} \text{ReLU}(a_{11} + b) & \text{ReLU}(a_{12} + b) & \dots & \text{ReLU}(a_{1m} + b) \\ \text{ReLU}(a_{21} + b) & \text{ReLU}(a_{22} + b) & \dots & \text{ReLU}(a_{2m} + b) \\ \vdots & \ddots & & \vdots \\ \text{ReLU}(a_{n1} + b) & \text{ReLU}(a_{n2} + b) & \dots & \text{ReLU}(a_{nm} + b) \end{pmatrix}$$

Figure 3.24.: Application of an activation function with a bias $b \in \mathbb{R}$ to a matrix

zero, the function returns zero; for positive values, it behaves linearly [Aga18b]. Thus, only positive values are returned (Abb. 3.23).

For example, an activation function can be applied after a convolution. This is shown in the equation ??.

3.5.14. Pooling

After the use of convolutional layers, pooling layers are often used in CNN to reduce the size of the representation, to also speed up the calculations as well as to make some of the detection functions a bit more robust. A distinction is usually made between two pooling methods.

- Max Pooling
- Average Pooling

In pooling, areas of a matrix are combined. The ranges are defined by two values n and m . The ranges are submatrices of the matrix of size $n \times m$. A function is then applied to each submatrix.

In max-pooling, the maximum value of the submatrix is determined and entered as the result. In the case of average pooling, the average value of the matrix elements is determined instead of the maximum value and used further.

In the case of pooling, a window with a defined size is moved over the input matrix and the corresponding value within the window is taken over as a pooled value. The step size with which the window moves over the matrix is defined via a parameter [DV16]. The default value corresponds to the size of the sub-matrix. An example is shown in the representation 3.25: a 4×4 -matrix is defined with size `pool_size` 2×2 and a step size `stride_size` 2×2 processed. Since two values are combined into each dimension, the dimensions of the output matrix are halved in contrast to the input matrix. [Bri15]

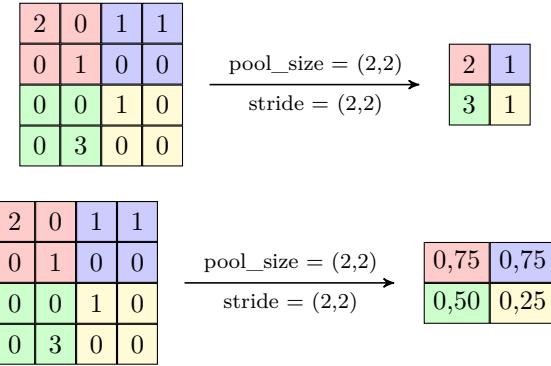


Figure 3.25.: Application of max-pooling and average-pooling

3.5.15. Flattening

With a flatten layer, the multidimensional input object is converted into a vector. For example, if an object with dimensions $5 \times 7 \times 7$ is passed in, it results in a one-dimensional vector with 245 elements. In the figure 3.26, the function Flatten is applied to a 3×3 matrix. The function is used when a neural network is subsequently used.

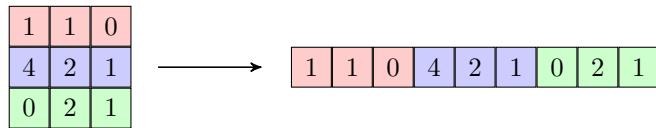


Figure 3.26.: Application of flattening

3.5.16. Dense Layer

In a layer, or fully-connected layer, is a layer that uses a neural network. This means that each of the previous outputs are fed into each of the inputs of the layer. Similarly, each output is linked to each of the subsequent inputs. This type of layer is usually used to be able to classify using the feature maps generated by the convolution. [SIG20a] It is determined by the number of neurons and the associated activation function.

3.5.17. Model used for a CIFAR 10 classification

The model used is an object `keras.models.Sequential`. With the help of the method `add`, the individual layers can be added to this model. The structure of the model is modelled on the Tensorflow tutorial for CNNs [Goo20a]. The structure of the model is shown in Abb. 3.27.

The layers described previously are now used to create a model for classifying the dataset CIFAR-10 with Keras. Here we look at how these layers are applied in the Python environment and how the size and dimension of the data changes between each layer. The dimensions of the input as well as output data of the individual layers are mapped in Abb. 3.27.

At the beginning, as described in ??, a Keras object `Sequential` is created. The first layer added to this model is a convolution layer `Conv2D`, see line 2. This is defined with 32 filters, each with a size of 3×3 . If, when creating the object, the parameter `stride` is not explicitly specified when the object is created, the default step size (1, 1) is used. Since the input data into the model are RGB images with 32×32 pixels, see Abschnitt 4.1.2, the parameter `input_shape` is taken to be $32 \times 32 \times 3$. This

```

1 MODEL = models.Sequential()
2 MODEL.add(layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
3   ↪ input_shape=(32, 32, 3)))
4 MODEL.add(layers.MaxPooling2D(pool_size=(2, 2)))
5 MODEL.add(layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
6 MODEL.add(layers.MaxPooling2D(pool_size=(2, 2)))
7 MODEL.add(layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
8 MODEL.add(layers.Flatten())
9 MODEL.add(layers.Dense(units=64, activation='relu'))
10 MODEL.add(layers.Dense(units=10))

```

Listing 3.1.: Building a CNN with Keras
src:cifarlayers

configuration transforms the input data in the first convolutional layer from $32 \times 32 \times 3$ to an output of $30 \times 30 \times 32$. A separate feature map is stored for each of the applied filters.

The next layer, a max-pooling layer with a `pool_size` of 2×2 , reduces the data size from $30 \times 30 \times 32$ to $15 \times 15 \times 32$ (line 3).

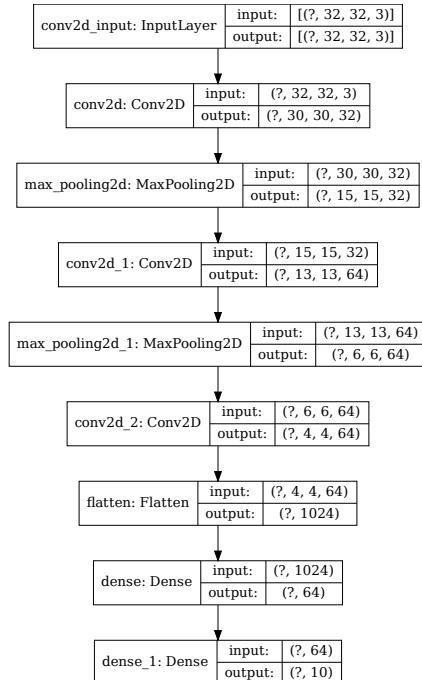


Figure 3.27.: Structure of the neural network used

This data is now fed again into a convolutional layer, which here is also configured with a `kernel_size` of 2×2 , ??, line 4, but with 64 filters. This transforms the data from $15 \times 32 \times$ to $13 \times 64 \times$.

To reduce the amount of data again, another max-pooling layer is applied, which is also configured with a `pool_size` of 2×2 in line 5, reducing the amount of data from $13 \times 13 \times 64$ to $6 \times 6 \times 64$.

The last convolutional layer is then applied to this network. This, like the previous convolutional layer, works with `kernel_size` 2×2 and 64 filters as per line 6. This transforms the data from $6 \times 64 \times$ to $4 \times 64 \times$.

To enable classification of this information with a neural network, the so-called dense layer, the results of the convolutional layers are transformed into a vector with a flatten layer, see line 7. Due to the input size of $4 \times 64 \times$ this results in a vector of

length 1024.

The vector is fed into a Dense layer in line 8. This layer is parameterised with 64 **units** and the activation function ReLu, which combines the 1024 input data into 64 nodes and calculates the corresponding outputs.

In the last layer, the 64 output values are passed to another Dense layer, which has been configured with 10 **units**. The input data are thus now reduced to 10 output data, which correspond to the classes to be identified of the data set CIFAR-10.

4. Databases and Models for Machine Learning

Data is the basis for machine learning. The success of the models depends on their quality. This is because machine learning relies on accurate and reliable information in the training of its algorithms. This self-evident fact is well known, but unfortunately not sufficiently taken into account. Poor data leads to insufficient or incorrect results. This goes without saying, but is often overlooked. The training data is realistic if it reflects the data that the AI system records in real use. Unrealistic data sets hinder machine learning and lead to expensive misinterpretations. If one wants to develop software for drone cameras, realistic images must also be used. In such a case, if one uses corresponding images from the web, they usually have the following characteristics:

- The perspective is more like head height.
- The targeted property is located in the centre.

If one wants to use data sets for one's own needs, care must be taken that only data that are also realistic are used. The data sets must also not contain any outliers or redundancies. When checking the quality of the data, the following questions can be helpful:

- By what means and what technique was the data generated?
- Is the data source credible?
- With what intention was the data collected?
- Where does the data come from? Is it suitable for the intended application?
- How old is the data?
- In what environment/conditions was the data created?

If necessary, collect your own data or have it collected.

Anyone who does data science can measure their developed algorithms against the results of others by using standardised data sets. Many databases and pre-trained models are available on the internet for this purpose. This chapter describes some of them. It should be noted that many data sets are available in different versions. Depending on the provider, the data may already have been processed and prepared for training. Here it is important to make sure that a suitable variant is used. Access to different data sets is available on the following internet sites:

Kaggle.com: Hier werden über 20.000 Datensätze angeboten. Dazu ist nur ein kostenloses Benutzerkonto notwendig.

lionbridge.ai: The website offers a good overview of data sets from the public and commercial sectors.

govdata.de: The Data Portal for Germany offers freely available data from all areas of public administration in Germany.

```
# Construct a tf.data.Dataset
ds = tfds.load('mnist', split='train', shuffle_files=True)

# Build your input pipeline
ds = ds.shuffle(1024).batch(32).prefetch(tf.data.experimental.AUTOTUNE)
for example in ds.take(1):
    image, label = example["image"], example["label"]
```

Listing 4.1.: Loading a dataset with TensorFlow

American government database: The American government also operates a portal where records of the administration are available.

scikit data sets: Data sets are also installed with the Python library. There are only a few datasets, but they are already pre-worked so that they are easy to load and use.

UCI - Center for Machine Learning and Intelligent System: The University of Irvine in California offers around 600 data sets for its own study.

TensorFlow: TensorFlow data sets: A collection provides ready-to-use datasets. All datasets are made available via the structure [alstf.data.Datasets](#) or [wastf.data.Datasets](#). The datasets can also be retrieved individually via GitHub.

Open Science Data Cloud: The platform aims to create a way for everyone to access high-quality data. Researchers can house and share their own scientific data, access complementary public datasets, create and share customised virtual machines with the tools they need to analyse their data.

Amazon: Amazon also provides data sets. You have to register for this free of charge.

KDnuggets.com: Similar to Kaggle, but links to other websites.

paperswithcode.com: The platform provides a possibility to exchange data sets. Here you will also find many well-known datasets with their links.

Google Dataset Search: The website does not directly offer datasets, but rather search support. Google restricts its search engine here to data sets.

faces: Labeled Faces in the Wild is a public benchmark for face verification, also known as pair matching. No matter what the performance of an algorithm, it should not be used to conclude that an algorithm is suitable for any commercial purpose.

4.1. Datenbanken

4.1.1. Data Set MNIST

The dataset MNIST is available for free use and contains 70,000 images of handwritten digits with the corresponding correct classification. [Den+09; Den12] The name of the dataset is justified by its origin, as it is a modified dataset from two datasets from the US National Institute of Standards and Technology. These contain handwritten digits from 250 different people, consisting of US Census Bureau employees and high school students. The NIST Special Database 3 (SD-3) and NIST Special Database 1 (SD-1) datasets collected in this way were merged because the former dataset of clerical workers contains cleaner and more easily recognisable data. Initially, SD-3 was used

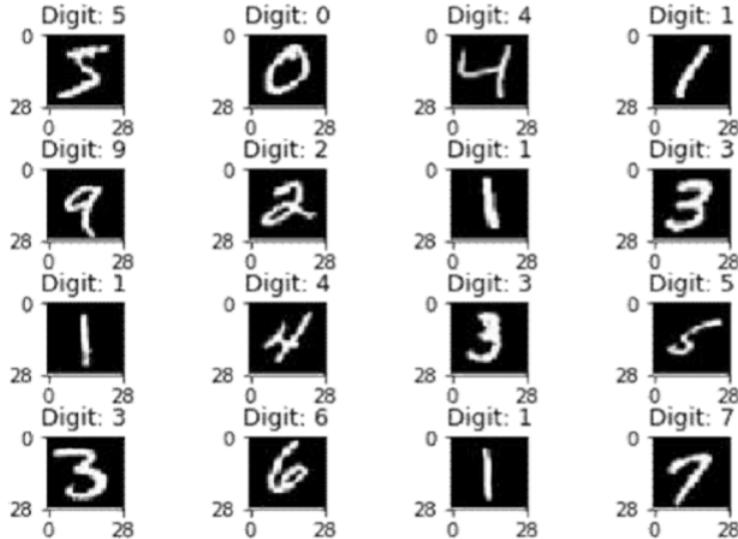


Figure 4.1.: Examples from the training set of the dataset MNIST [SSS19]

as the training set and SD-1 as the test set, but because of the differences, it makes more sense to merge the two. The dataset is already split into 60,000 training images and 10,000 test images. [LCB13; Nie15]

The data is provided in four files, two for the training dataset and two for the test dataset, one of which contains the image data and the other the associated labels:

- [train-images-idx3-ubyte.gz](#): Training pictures (9912422 bytes)
- [train-labels-idx1-ubyte.gz](#): Classification of the training data (28881 bytes)
- [t10k-images-idx3-ubyte.gz](#): Test patterns (1648877 bytes)
- [t10k-labels-idx1-ubyte.gz](#): Classification of the test patterns (4542 bytes)

The images in the dataset MNIST are greyscale images of size 28×28 pixels.[LCB13] The data is in IDX file format and so cannot be opened and visualised by default. However, you can write a programme to convert the data into CSV format or load a variant in CSV format directly from other websites. [Red20]

The library TensorFlow provides the dataset MNIST under [tensorflow_datasets](#). This is not the only dataset provided by TensorFlow. A list of all datasets that can be loaded via so can be found in catalogue of datasets.

4.1.2. CIFAR-10 and CIFAR-100

Data Set CIFAR-10

The data sets CIFAR-10 and CIFAR-100 were developed by Alex Krizhevsky and his team and therefore named after the Canadian Institute for Advanced Research.

The data set CIFAR-10 is a partial data set of the data set *80 Million Tiny Images*. This part consists of 60,000 images, divided into 50,000 training images and 10,000 test images, which are divided into 10 classes and labelled accordingly. The existing classes represent aeroplanes, cars, birds, cats, deer, dogs, frogs, horses, ships and trucks. Thus, 6,000 images exist per class, with each image having a size of 32×32 pixels with three colour channels [KH09; KSH17]. An extract of the dataset can be seen in Abb. 4.2.



Figure 4.2.: Extract from ten illustrations of the data set CIFAR-10

```

1   (TRAIN_IMAGES, TRAIN_LABELS), (TEST_IMAGES, TEST_LABELS) = datasets.cifar10.
2     ↪ load_data()
3   TRAIN_IMAGES = TRAIN_IMAGES / 255.0
3   TEST_IMAGES = TEST_IMAGES / 255.0

```

Listing 4.2.: Load and normalise the data set CIFAR-10

To import the dataset into the Python environment, the module `keras.datasets` is used here. This contains a direct interface to the dataset CIFAR-10, which is downloaded to the local system by calling the function `load_data()`. In the background, the dataset is downloaded as a packed file with the extension `.tar.gz` and stored in the folder `C:\Users\<username>\.keras\datasets`. After unpacking, the dataset is contained in several serialised objects, which are loaded into the Python environment via `pickle`. The training data, training labels, test data and test labels can then be accessed via corresponding lists (Abschnitt 4.1.2). In order to reduce the vanishing gradient problem during the later training, the RGB colour values stored in the data set with the value range from 9 to 255 are converted into values from 0 to 1. The listing ?? shows the loading and normalisation of the data.

Data Set CIFAR-100

The data set CIFAR-100, on the other hand, also contains 60,000 images, but divided into 100 categories with 600 images each. In addition, there are 20 upper categories, to each of which five of the 100 categories are assigned; see table 4.3.

The dataset can also be downloaded and imported directly into TensorFlow via Keras [Raj20]:

```

1   import tensorflow as tf
2   from tensorflow.keras import datasets
3
4   (TRAIN_IMAGES100, TRAIN_LABELS100), (TEST_IMAGES100, TEST_LABELS100) = tf.keras.
5     ↪ datasets.cifar100.load_data()
5   TRAIN_IMAGES100 = TRAIN_IMAGES100 / 255.0
6   TEST_IMAGES100 = TEST_IMAGES100 / 255.0

```

Listing 4.3.: Load and normalise the data set CIFAR-100

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Figure 4.3.: Supercategories and categories in CIFAR-100 with the original designations

4.1.3. Fisher's Iris Data Set

The recognition of the iris is another classic for image classification. There are also several tutorials on this classic. The Fisher's Iris Data Set was used in R.A. Fisher's classic 1936 work and can be found in the UCI Machine Learning Repository. [Fis; Fis36a; SW16; UK21] Four features of the flowers Iris Setosa, Iris Versicolour and Iris Virginica were measured. For each of the three classes 50 data sets with four attributes are available. The width and length of the sepal and petal were measured in centimetres. [And35; SP06]. One species of flower is linearly separable from the other two, but the other two are not linearly separable from each other.

The data set can be accessed in several places:

- Archive of machine learning datasets from the University of California at Irvine. [Fis36b]
- Python package [skikitlearn](#) [Ped+11]
- Kaggle - Machine Learning Website [Kag]

The data set is available on various websites, but attention must be paid to the structure of the data set. Since it is a file in CSV format, the structure is column-oriented. As a rule, the title of the individual column is given in the first line: `sepal_length`, `sepal_width`, `petal_length`, `petal_width` and `species`. The values are in centimetres, the species is given as `setosa` for Iris setosa, `versicolor` for Iris versicolor and `virginica` for the species Iris virginica. The columns in this record are:

- Sequence number
- Length of sepal in centimetres
- Sepal width in centimetres
- Petal length in centimetres

```
1 from sklearn.datasets import load_iris
2 iris = load_iris()
```

Listing 4.4.: Loading the Fisher's Iris Data Set

- Petal width in centimetres
- Class

The aim is to classify the three different iris species based on the length and width of the sepal and petal. Since the dataset is delivered with the library `sklearn`, this approach is chosen.

The record is a `dictionary`. Its keys can be easily displayed:

```
1 >>> iris.keys()
```

The related issue is as follows:

```
dict\_keys(['data', 'target', 'frame', 'target\_names', 'DESCR', 'feature\_na
```

The individual elements can now be viewed. After entering the command

```
1 iris['DESCR']
```

a detailed description is output:

With the input

```
1     iris['feature_names']
```

you get the names of the attributes:

```
[ 'sepal_length_(cm)',  
  'sepal_width_(cm)',  
  'petal_length_(cm)',  
  'petal_width_(cm)' ]
```

The names of the flowers that appear after entering

```
1     iris['target_names']
```

are displayed, are

```
array(['setosa', 'versicolor', 'virginica'], dtype='|<U10')
```

For further investigation, the data with the headings are included in a data frame.

```
1 X = pd.DataFrame(data = iris.data, columns = iris.feature_names)
2 print(X.head())
```

The command (`X.head()`) shows – as in Figure 4.4 – the head of the data frame. It is obvious that each data set consists of four values.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Figure 4.4.: Header lines of Fisher's Iris Data Set

Each record also already contains its classification in the key `target`. In the figure 4.5 this is listed for the first data sets.

```

1  '.. _iris_dataset:
2
3 Iris plants dataset
4 -----
5
6 **Data Set Characteristics:***
7
8 :Number of Instances: 150 (50 in each of three classes)
9 :Number of Attributes: 4 numeric, predictive attributes and the class
10 :Attribute Information:
11   - sepal length in cm
12   - sepal width in cm
13   - petal length in cm
14   - petal width in cm
15   - class:
16     - Iris-Setosa
17     - Iris-Versicolour
18     - Iris-Virginica
19
20 :Summary Statistics:
21
22 ===== Min Max Mean SD Class Correlation
23 Min Max Mean SD Class Correlation
24 ===== Min Max Mean SD Class Correlation
25 sepal length:  4.3  7.9  5.84  0.83  0.7826
26 sepal width:  2.0  4.4  3.05  0.43  -0.4194
27 petal length: 1.0  6.9  3.76  1.76  0.9490 (high!)
28 petal width:  0.1  2.5  1.20  0.76  0.9565 (high!)
29 ===== Min Max Mean SD Class Correlation
30
31 :Missing Attribute Values: None
32 :Class Distribution: 33.3% for each of 3 classes.
33 :Creator: R.A. Fisher
34 :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
35 :Date: July, 1988
36
37 The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
38 from Fisher's paper. Note that it's the same as in R, but not as in the UCI
39   ↪ nMachine Learning Repository, which has two wrong data points.
40
41 This is perhaps the best known database to be found in the
42 pattern recognition literature. Fisher's paper is a classic in the field and
43 is referenced frequently to this day. (See Duda & Hart, for example.) The
44 data set contains 3 classes of 50 instances each, where each class refers to a
45   ↪ ntype of iris plant. One class is linearly separable from the other 2;
46   ↪ the latter are NOT linearly separable from each other.
47
48 .. topic:: References
49
50   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
51     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
52     Mathematical Statistics" (John Wiley, NY, 1950).
53   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
54     (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.
55   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
56     Structure and Classification Rule for Recognition in Partially Exposed
57     Environments". IEEE Transactions on Pattern Analysis and Machine
58     Intelligence, Vol. PAMI-2, No. 1, 67-71.
59   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions
60     on Information Theory, May 1972, 431-433.
61   - See also: 1988 MLC Proceedings, 54-64. Cheeseman et al's AUTOCLASS II
62     conceptual clustering system finds 3 classes in the data.
63   - Many, many more ...'
```

Listing 4.5.: Description of Fisher's Iris Data Set

```

1   y = pd.DataFrame(data=iris.target, columns = ['irisType'])
2   y.head()

```

irisType	
0	0
1	0
2	0
3	0
4	0

Figure 4.5.: Output of the categories of Fisher's Iris Data Set

By means of the command

```
1   y.irisType.value_counts()
```

can be used to determine how many classes are present. The output of the command is shown in the figure 4.6; it results in 3 classes with the numbers 0, 1 and 2. There are 50 records assigned to each of them.

```

2      50
1      50
0      50
Name: irisType, dtype: int64

```

Figure 4.6.: Names of the categories in Fisher's Iris Data Set

4.1.4. Common Objects in Context - COCO

The dataset COCO is labelled and provides data for training supervised computer vision models that are able to identify common objects in the dataset. Of course, these models are still far from perfect. Therefore, the dataset COCO provides a benchmark for evaluating the periodic improvement of these models through computer vision research.[Lin+14; Lin+21]

- Object recognition
 - The dataset COCO contains \approx 330,000 images.
 - The dataset COCO contains \approx 1,500,000 annotations for objects.
 - The data set COCO contains 80 categories.
 - The pictures each have five headings.
 - The images have a medium resolution of 640×480 pixels.
- Semantic segmentation

```
{
  "info": {...},
  "licenses": {...},
  "images": {...},
  "categories": {...},
  "annotations": {...}
}
```

Listing 4.6.: Information of the data set COCO

```
{
  "description": "COCO_2017_Dataset",
  "url": "http://cocodataset.org",
  "version": "1.0",
  "year": 2017,
  "contributor": "COCO Consortium",
  "date_created": "2017/09/01"
}
```

Listing 4.7.: Metainformation of the dataset COCO

- Panoptic segmentation requires models for drawing boundaries between objects in semantic segmentation.
- Key recognition
 - The images contain 250,000 people labelled with the corresponding keys.

Due to the size and frequent use of the dataset, there are many tools, for example COCO-annotator and COCOapi, to access the data.

Actually, the dataset COCO consists of several, each made for a specific machine learning task. The first task is to determine surrounding rectangles for objects. That is, objects are identified and the coordinates of the surrounding rectangle are determined. The extended task is object segmentation. Here, objects are also identified, but in addition, instead of the surrounding rectangles, polygons are drawn to delimit the objects. The third classical task is cloth segmentation. The model should perform object segmentation, but not on individual objects, but on continuous background patterns such as grass or sky.

The annotations are stored in JSON format. The JSON format is a dictionary with key-value pairs in curly brackets. It can also contain lists, ordered collections of elements within curly braces, or dictionaries nested within them.

Section “Info”

The dictionary for the `info` section contains metadata about the record. For the official record COCO it is the following information:

As can be seen, only basic information is included, with the `url` value pointing to the official website of the dataset. This is common for machine learning datasets to point to their websites for additional information. For example, there you can find information on how and when the data was collected.

In the section “licenses” you will find links to licenses for images in the dataset with the following structure:

```
[
{
  "url": "http://creativecommons.org/licenses/by-nc-sa/2.0/",
  "id": 1,
  "name": "Attribution-NonCommercial-ShareAlike License"
},
{
  "url": "http://creativecommons.org/licenses/by-nc/2.0/",
  "id": 2,
  "name": "Attribution-NonCommercial License"
},
...
]
```

Listing 4.8.: Licence information of the data set COCO

```
{
  "license": 3,
  "file_name": "000000391895.jpg",
  "coco_url": "http://images.cocodataset.org/train2017/000000391895.jpg",
  "height": 360,
  "width": 640,
  "date_captured": "2013-11-14 11:18:45",
  "flickr_url": "http://farm9.staticflickr.com/8186/8119368305_4e622c8349_z.jpg",
  "id": 391895
}
```

Listing 4.9.: Image information of the dataset COCO

This dictionary “images” is probably the second most important and contains metadata about the images.

The images dictionary contains the field id. In this field the licence of the image is given. The full text is given in the URL. When using the images, it must be ensured that no licence infringement occurs. If in doubt, do not use them. This also means, however, that when creating one’s own data set, one assigns an appropriate licence to each image.

The most important field is the id field. This is the number used in the annotations section to identify the image. So, for example, if you want to identify the annotations for the given image file, you have to check the value of the id field for the corresponding image document in the images section and then cross-reference it in the annotations section.

In the official record COCO, the value of the id field is the same as the name `file-name` after removing the leading zeros. If one uses a custom record COCO, this may not necessarily be the case.

Section “Categories”

The categories section is a little different from the other sections. It is designed for the task of object recognition and segmentation and for the task of substance segmentation.

For object recognition and object segmentation, the information is obtained according to the listing 4.10

```
[
  {"supercategory": "person", "id": 1, "name": "person"},  

  {"supercategory": "vehicle", "id": 2, "name": "bicycle"},  

  {"supercategory": "vehicle", "id": 3, "name": "car"},  

  ...  

  {"supercategory": "indoor", "id": 90, "name": "toothbrush"}  

]
```

Listing 4.10.: Class information of the dataset COCO

```
[
  {"supercategory": "textile", "id": 92, "name": "banner"},  

  {"supercategory": "textile", "id": 93, "name": "blanket"},  

  ...  

  {"supercategory": "other", "id": 183, "name": "other"}  

]
```

Listing 4.11.: Substance information of the dataset COCO

In the section, the lists contain the categories of objects that can be recognised on images. Each category has a unique number id and it should be in the range [1, number of categories]. Categories are also grouped into supercategories that can be used in programs to recognise, for example, vehicles in general, when you don't care if it is a bicycle, a car or a truck.

There are separate lists for substance segmentation, see Listing 4.11

The number of categories in this section start high to avoid conflicts with object segmentation, as these tasks can be performed together in the panoptic segmentation task. The values from 92 to 182 represent the well-defined background material, while the value 183 represents all other background textures that do not have their own classes.

The annotations section is the most important section of the dataset, containing important information for the specific dataset for each task.

The fields according to the listing 4.12 haben folgende Bedeutung.

“segmentation”: This is a list of segmentation masks for pixels; this is a flattened list of pairs, so you should take the first and second values (x and y in the image), then the third and fourth, and so on, to get coordinates; note that these are not image indices, as they are floating point numbers — they are created and compressed from the pixel coordinates by tools like COCO-annotator.

“area”: This corresponds to the number of pixels within a segmentation mask.

“iscrowd”: This is a flag indicating whether the caption applies to a single object (value 0) or to several objects close to each other (value 1); for fill segmentation, this field is always 0 and is ignored.

“image_id”: The id field corresponds to the number of the id field from the images dictionary; warning: this value should be used to cross-reference the image with other dictionaries, i.e. not the field id.

“bbox”: The heading contains the surrounding rectangles or bounding box, i.e. the coordinates in the form of the x and y coordinates of the upper left corner, as

```
{  
    "segmentation":  
        [[  
            239.97,  
            260.24,  
            222.04,  
            ...  
        ]],  
    "area": 2765.1486500000005,  
    "iscrowd": 0,  
    "image_id": 558840,  
    "bbox":  
        [  
            199.84,  
            200.46,  
            77.71,  
            70.88  
        ],  
    "category_id": 58,  
    "id": 156  
}
```

Listing 4.12.: Annotations of the dataset COCO

well as the width and height of the rectangle around the object; it is very useful for extracting individual objects from images, as in many languages such as Python this can be done by accessing the image array as;

```
cropped_object = image[bbox[0]:bbox[0] + bbox[2], bbox[1]:bbox[1] + bbox[3]]
```

“category_id”: The field contains the class of the object, corresponding to the field id from the section “categories”

“id”: The number is the unique identifier for the annotation; note that this is only the ID of the annotation, so it does not refer to the respective image in other dictionaries.

When working with crowd images (“iscrowd”: 1), the “segmentation” part may be a little different:

```
"segmentation":  
{  
    "counts": [179, 27, 392, 41, ..., 55, 20],  
    "size": [426, 640]  
}
```

This is because with a large number of pixels, explicitly listing all the pixels that create a segmentation mask would take a lot of space. Instead, the dataset COCO uses a custom compression Run-Length Encoding (RLE), which is very efficient because segmentation masks are binary and RLE for only zeros and ones can reduce the size many times.

4.1.5. ImageNet

ImageNet is an image database with more than 14 million images that is constantly growing. Each image is assigned to a noun. For each category there are on average more than 500 images. ImageNet contains more than 20,000 categories in English with a typical category such as „balloon“ or „strawberry“. The database of third-party image URL annotations is freely accessible directly through ImageNet, although the actual images are not owned by ImageNet. [Den+09; Sha+20; KSH12b]

4.1.6. Visual Wake Words

The „Visual Wake Words“ dataset consists of 115,000 images, each with an indication of whether or not it contains a person. [Cho+19] Images are extracted from the COCO dataset. Since the dataset COCO provides information about the images, it can be queried during extraction whether there is a person inside a surrounding rectangle or not. The data is marked with the information.

4.2. Datensatz UTKFace

Um ein maschinelles Lernmodell für die Gesichtserkennung zu trainieren, müssen wir zunächst einen Datensatz von Bildern mit Gesichtern erstellen. Dies können wir tun, indem wir die Bilder mit der Vision Shield-Kamera aufnehmen. Wir können auch einen Datensatz online herunterladen und ihn zum Trainieren des Modells verwenden. UTKFace ist ein Datensatz mit einer langen Altersspanne (0 bis 116 Jahre alt), der aus über 20.000 Einzelbildern von Gesichtern besteht. Wir können diesen Datensatz über den Link <https://susandqq.github.io/UTKFace/> herunterladen.

Sicherlich können auch manuell aufgenommene Bilder von der Kamera verwendet werden.

In der folgenden Abbildung sehen wir die Beispielbilder aus dem UTKFace-Datensatz. Die Größe der Bilder in diesem Datensatz beträgt 200×200 Pixel.

WS:Besser beschreiben

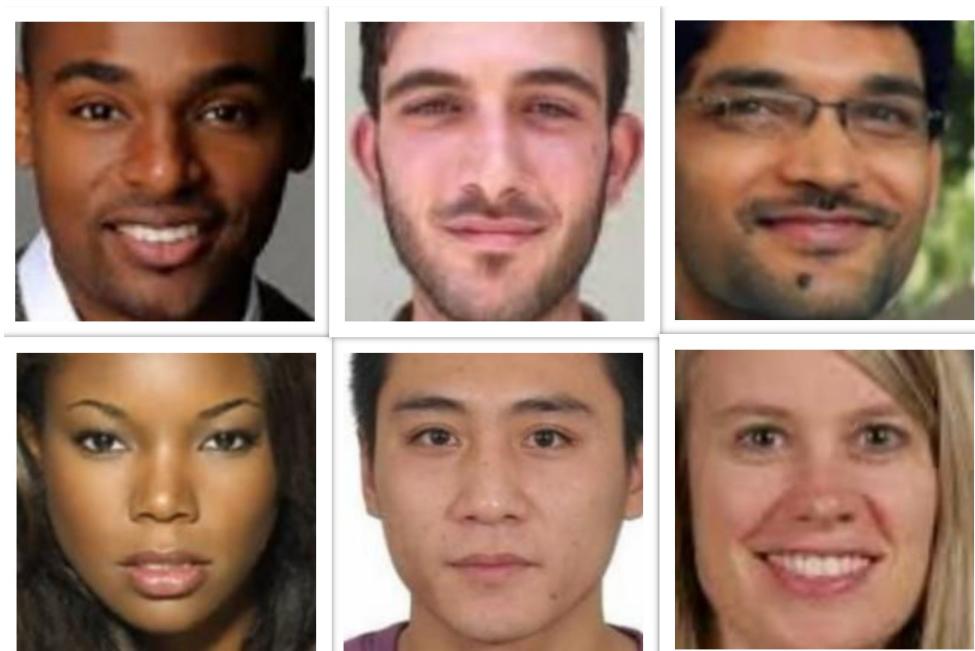


Figure 4.7.: Beispielbilder aus dem Datensatz UTKFace

Network	CLI argument	NetworkType enum
AlexNet	alexnet	ALEXNET
GoogleNet	googlenet	GOOGLENET
GoogleNet-12	googlenet-12	GOOGLENET_12
ResNet-18	resnet-18	RESNET_18
ResNet-50	resnet-50	RESNET_50
ResNet-101	resnet-101	RESNET_101
ResNet-152	resnet-152	RESNET_152
VGG-16	vgg-16	VGG-16
VGG-19	vgg-19	VGG-19
Inception-v4	inception-v4	INCEPTION_V4

Table 4.1.: Pre-trained models of the project jetson-inference

4.3. Models

When installing NVidia's jetson-inference project, the utility offers a large selection of pre-trained deep learning models for download. Among them are well-known ones like the AlexNet from 2012 as well as various so-called ResNet. Also included are SSD-MobileNet-V2, which recognises 90 objects from apples to toothbrushes, and DeepScene from the University of Freiburg for recognising objects in road traffic. To download more models, the Model Downloader can be called up:

```
$ cd jetson-inference/tools
$ ./download-models.sh
```

The models GoogleNet and ResNet-18, which are based on the ImageNet image database, are automatically downloaded in the build step. The table ?? lists all models that are available in the jetson-inference project. The first column of the table is the name of the structure of the model. [Alo+18]. The second column contains the passing parameter for the `-network` argument of the `imagenet-camera.py` program.

4.4. FaceNet

[SKP15]

FaceNet lernt eine Abbildung von Gesichtsbildern auf einem kompakten euklidischen Raum, in dem Abstände direkt einem Maß für die Ähnlichkeit von Gesichtern entsprechen. Sobald dies geschehen ist, sind Aufgaben wie Gesichtserkennung, -verifizierung und -clusterung mit Standardtechniken (unter Verwendung der FaceNet-Einbettungen als Merkmale) einfach zu erledigen. Verwendet ein Deep CNN, das trainiert wird, um die Einbettung selbst zu optimieren, anstatt die Ausgabe einer dazwischenliegenden Engpassschicht zu verwenden. Das Training erfolgt mit Triplets: ein Bild eines Gesichts ("Anker"), ein weiteres Bild desselben Gesichts ("positives Exemplar") und ein Bild eines anderen Gesichts ("negatives Exemplar"). Der Hauptvorteil liegt in der Repräsentationseffizienz: Mit nur 128 Byte pro Gesicht kann eine Spitzenleistung erzielt werden (Rekordgenauigkeit von 99,63 % bei LFW, 95,12 % bei Youtube Faces DB).

siehe [.../MLbib/CNN/class10_FaceNet.pdf](#)

FaceNet ist ein tiefes Faltungsneuronales Netzwerk, das von Google-Forschern entwickelt und um 2015 eingeführt wurde, um die Hürden bei der Gesichtserkennung und -verifizierung effektiv zu lösen. Der FaceNet-Algorithmus transformiert das Gesichtsbild in einen 128-dimensionalen euklidischen Raum, ähnlich wie beim Word Embedding[9]. Das so erstellte FaceNet-Modell wird auf Triplet-Verlust trainiert, um die Ähnlichkeiten und Unterschiede auf dem bereitgestellten Bilddatensatz zu erfassen. Die

vom Modell erzeugten Einbettungen mit 128 Dimensionen können verwendet werden, um Gesichter auf sehr effektive und präzise Weise zu clustern. Durch die Verwendung von FaceNet-Embeddings als Merkmalsvektoren könnten nach der Erstellung des Vektorraums Funktionalitäten wie Gesichtserkennung und -verifikation implementiert werden[10]. Kurz gesagt, die Abstände für die ähnlichen Bilder würden viel näher sein als die zufälligen nicht ähnlichen Bilder. Die allgemeine Blockdarstellung des FaceNet-Ansatzes der Gesichtserkennung ist in Abb.1 dargestellt.

siehe [.../.../MLbib/CNN/10.1109ICACCS.2019.8728466.pdf](#)

Eingabeformat? Farben? Pixelauflösung?

5. Frameworks and Libraries

As a programming language for data science, Python represents a compromise between the R language, which focuses on data analysis and visualisation, and Java, which forms the backbone of many large-scale applications. This flexibility means that Python can act as a single tool that brings your entire workflow together.

Python is often the first choice for developers who need to apply statistical techniques or data analysis to their work, or for data scientists whose tasks need to be integrated into web applications or production environments. Python particularly shines in the area of machine learning. The combination of machine learning libraries and flexibility makes Python unique. Python is best suited for developing sophisticated models and predictive modules that can be directly integrated into production systems.

One of Python's greatest strengths is its extensive library. Libraries are sets of routines and functions written in a specific language. A robust set of libraries can make the job of developers immensely easier to perform complex tasks without having to rewrite many lines of code.

5.1. Development Environment

5.1.1. Jupyter-Notebook

The Jupyter Notebook application can be used to create a file in a web browser. The advantage here is that programmes can be divided into smaller programme sections and executed section by section. This makes working very interactive. The notebook also offers the possibility of annotating Python programme codes with additional text. [BS19]

5.2. Frameworks

5.2.1. TensorFlow

The TensorFlow framework, which is supported by Google, is the undisputed top dog. It has the most GitHub activity, Google searches, Medium articles, books on Amazon and ArXiv articles. It also has the most developers using it, and is listed in the most online job descriptions. [Goo19a]

Variant also exist for TensorFlow that is specific to a hardware. For example, NVIDIA graphics cards are addressed by a variant with Compute Unified Device Architecture (CUDA) and Intel also has an optimised variant. However, you may have to do without the latest version.

TensorFlow Lite, the small device version, brings model execution to a variety of devices, including mobile devices and IoT, and provides more than a 3x increase in inference speed compared to the original TensorFlow.

TensorFlow uses data flow graphs to represent computation, shared state, and the operations that mutate that state. By fully representing the dependencies of each step of the computation, parts of the computation can be reliably parallelised [Aba+16]. Originally, TensorFlow was to be used for Google's internal use, but was released in November 2015 under the Apache 2.0 open source licence. Since then, TensorFlow has brought together a variety of tools, libraries and communities. TensorFlow provides low-level interfaces for programming languages such as Python, Java, C++ and Go.

The mid-level and high-level APIs provide functions for creating, training, saving and loading models, training, saving and loading models. [Cho18]

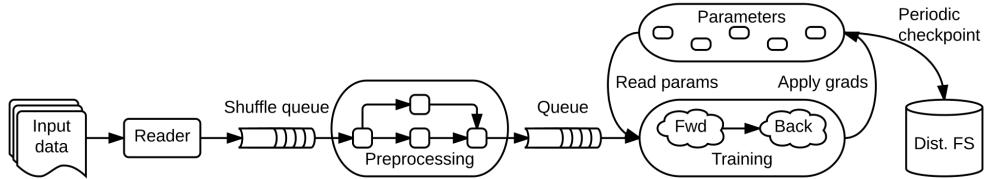


Figure 5.1.: Tensorflow Datenflussdiagramm [Aba+16]

5.2.2. TensorFlow Lite

Tensorflow Lite is an optimised environment for TensorFlow models for mobile devices and the Internet of Things (IoT). Essentially, it consists of two components: the converter, which converts pre-trained TensorFlow models into optimised TensorFlow Lite models, and the interpreter, which enables optimised execution on the various end devices. The goal here is to reduce the size of the existing models and to reduce latency on less powerful devices [Goo20b].

5.2.3. TensorRT

TensorRT is a neural network runtime environment based on CUDA, which specialises in optimising the performance of neural networks [NVI15]. This is achieved by reducing the computational accuracy from Floating Point 32-bit (FP32) to Floating Point 16-bit (FP16) resp. Integer 8-bit (INT8). Since the remaining interference accuracy is sufficient for most use cases, the speed can be significantly increased with this method [GMG16]. When optimising the network, the operators used are replaced by TensorRT operators, which can only be executed within a TensorRT environment.

5.2.4. Keras

Keras is a high-level Application Programming Interface (API) application programming interface (API) for neural networks built in Python and running on TensorFlow, Theano or CNTK. It enables the definition and training of different deep learning models using optimised Tensor libraries, which serve as a backend engine. The TensorFlow backend, Theano backend and CNTK backend implementations are currently supported. Any code written in Keras can be run on the backends without customisation. Keras offers a choice of predefined layers, optimisation functions or other important neural network components. The functions can also be extended with custom modules. The two most important model types provided by Keras are sequential and functional api. With sequential, straight-line models can be created whose layers are lined up one after the other. For more complex network structures with feedback, there is the functional api. [Cho18; SIG20b]

5.2.5. PyTorch

PyTorch, which is supported by Facebook, is the third most popular framework. It is younger than TensorFlow and has rapidly gained popularity. It allows customisation that TensorFlow does not. [Fac20]

5.2.6. Caffe

Caffe has been around for almost five years. It is in relatively high demand by employers and frequently mentioned in academic articles, but has had little recent coverage of its use. [JES20]

5.2.7. Caffe2

Caffe2 is another open source product from Facebook. It builds on Caffe and is now in the PyTorch GitHub repository. [Sou20]

5.2.8. Theano

Theano was developed at the University of Montreal in 2007 and is the oldest major Python framework. It has lost much of its popularity and its leader stated that major versions are no longer on the roadmap. However, updates continue to be made. [AR+16]

Theano uses a NumPy-like syntax to optimise and evaluate mathematical expressions. What makes Theano different is that it uses the GPU of the computer's graphics card. The speed thus makes Theano interesting.

5.2.9. Apache MXNet

MXNET is supported by Apache and used by Amazon. [MXN20]

5.2.10. CNTK

CNTK is the Microsoft Cognitive Toolkit. It reminds me of many other Microsoft products in the sense that it tries to compete with Google and Facebook offerings and fails to gain significant acceptance. [Mic20]

5.2.11. DeepLearning4J

DeepLearning4J, also called DL4J, is used with the Java language. It is the only semi-popular framework that is not available in Python. However, you can import models written with Keras into DL4J. The framework has a connection to Apache Spark and Hadoop. [Ecl20]

5.2.12. Chainer

Chainer is a framework developed by the Japanese company Preferred Networks. It has a small following. [PN20]

5.2.13. FastAI

FastAI is built on PyTorch. Its API was inspired by Keras and requires even less code for strong results. Jeremy Howard, the driving force behind Fast.AI, was a top Kaggle and president of Kaggle. [fas20]

Fast.AI is not yet in demand for careers, nor is it widely used. However, it has a large built-in pipeline of users through its popular free online courses. It is also both powerful and easy to use. Its uptake could grow significantly.

5.3. General libraries

These are the basic libraries that transform Python from a general-purpose programming language into a powerful and robust tool for data analysis and visualisation. They are sometimes referred to as the SciPy stack and form the basis for the special tools.

5.3.1. NumPy

NumPy is the fundamental library for scientific computing in Python, and many of the libraries in this list use NumPy arrays as their basic inputs and outputs. In short, NumPy introduces objects for multidimensional arrays and matrices, and routines that allow developers to perform advanced mathematical and statistical functions on these arrays with as little code as possible. [Fou20b]

5.3.2. SciPy

SciPy builds on NumPy by adding a collection of algorithms and high-level commands for manipulating and visualising data. The package also includes functions for numerically calculating integrals, solving differential equations, optimisation and more.

5.3.3. pandas

Pandas adds data structures and tools designed for practical data analysis in finance, statistics, social sciences and engineering. Pandas is well suited for incomplete, messy and unlabelled data (i.e. For the kind of data you are likely to face in the real world) and provides tools for shaping, merging, reshaping and splitting datasets.

5.3.4. IPython

IPython extends the functionality of Python's interactive interpreter with a souped-up interactive shell that adds introspection, rich media, shell syntax, tab completion and command archive retrieval. It also acts as an integrated interpreter for your programs, which can be particularly useful for debugging. If you have ever used Mathematica or MATLAB, you will feel at home with IPython.

5.3.5. Matplotlib

Matplotlib is the standard Python library for creating 2D diagrams and plots. The API is quite low-level, i.e. it requires several commands to produce good-looking graphs and figures compared to some more advanced libraries. However, the advantage is greater flexibility. With enough commands, you can create almost any graph with matplotlib.

5.3.6. scikit-learn

scikit-learn builds on NumPy and SciPy by adding a set of algorithms for general machine learning and data mining tasks, including clustering, regression and classification. As a library, scikit-learn has much to offer. Its tools are well documented and among the contributing developers are many machine learning experts. Moreover, it is a very helpful library for developers who do not want to choose between different versions of the same algorithm. Its power and ease of use make the library very popular.

5.3.7. Scrapy

Scrapy is a library for creating spiderbots to systematically crawl the web and extract structured data such as prices, contact information and URLs. Originally developed for web scraping, Scrapy can also extract data from APIs.

5.3.8. NLTK

NLTK stands for Natural Language Toolkit and provides an effective introduction to Natural Language Processing (NLP) or text mining with Python. The basic functions of NLTK allow you to mark up text, identify named entities and display parse trees that reveal parts of speech and dependencies like sentence diagrams. This gives you the ability to do more complicated things like sentiment analysis or generate automatic text summaries.

5.3.9. Pattern

Pattern combines the functionality of Scrapy and NLTK into a comprehensive library that aims to serve as an out-of-the-box solution for web mining, NLP, machine learning and network analysis. Its tools include a web crawler; APIs for Google, Twitter and Wikipedia; and text analytics algorithms such as parse trees and sentiment analysis that can be run with just a few lines of code.

5.3.10. Seaborn

Seaborn is a popular visualisation library built on top of matplotlib. With Seaborn, graphically very high quality plots such as heat maps, time series and violin plots can be generated.

5.3.11. Bokeh

Bokeh allows the creation of interactive, zoomable plots in modern web browsers using JavaScript widgets. Another nice feature of Bokeh is that it comes with three levels of user interface, from high-level abstractions that let you quickly create complex plots to a low-level view that provides maximum flexibility for app developers.

5.3.12. basemap

Basemap supports adding simple maps to matplotlib by taking coordinates from matplotlib and applying them to more than 25 different projections. The library Folium builds on Basemap and allows the creation of interactive web maps, similar to the JavaScript widgets of Bokeh.

5.3.13. NetworkX

This library allows you to create and analyse graphs and networks. It is designed to work with both standard and non-standard data formats, making it particularly efficient and scalable. With these features, NetworkX is particularly well suited for analysing complex social networks.

5.3.14. LightGBM

Gradient Boosting is one of the best and most popular libraries for Machine Learning and helps developers to create new algorithms by using newly defined elementary models and especially decision trees.

Accordingly, there are special libraries designed for a fast and efficient implementation of this method. These are LightGBM, XGBoost and CatBoost. All these libraries are competitors but help solve a common problem and can be used in almost similar ways.

5.3.15. Eli5

Most of the time, model prediction results in machine learning are not really accurate. Eli5, a library built into Python, helps overcome this very challenge. It is a combination of visualising and debugging all machine learning models and tracking all steps of the algorithm.

5.3.16. Mlpy - Machine Learning

As an alternative to scikit-learn, Mlpy also offers a powerful library of functions for machine learning. Mlpy is also based on NumPy and SciPy, but extends the functionality with supervised and unsupervised machine learning methods.

5.3.17. Statsmodels - Statistical Data Analysis

Statsmodels is a Python module that allows users to explore data, estimate statistical models and run statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions and result statistics is available for various data types and each estimator. The module makes predictive analytics possible. Statsmodels is often combined with NumPy, matplotlib and Pandas.

5.4. Specialised Libraries

5.4.1. OpenCV

OpenCV, derived from Open Computer Vision, is a free program library with algorithms for image processing and computer vision. The development of the library was initiated by Intel and was maintained by Willow Garage until 2013. After their dissolution, it was continued by Itseez, which has since been acquired by Intel. [Tea20a]

5.4.2. OpenVINO

The OpenVINO toolkit helps accelerate the development of powerful computer vision and deep learning in vision applications. It enables Deep Learning via hardware accelerators as well as simple, heterogeneous execution on Intel platforms, including accpu, GPU, Field-Programmable Gate Array (FPGA) and Video Processing Unit (VPU). Key components include optimised features for OpenCV. [Int19; Tea20b]

5.4.3. Compute Unified Device Architecture (CUDA)

The abbreviation CUDA stands for Compute Unified Device Architecture. It is an interface technology and computing platform that allows graphics processors to be addressed and used for non-graphics-specific computations. CUDA was developed by NVIDIA and accelerates programmes by parallelising certain parts of the programme with one or more GPUs in addition to the Central Processing Unit (CPU). [TS19; Cor20; NVI20]

5.4.4. OpenNN

OpenNN is a software library written in C++ for advanced analysis. It implements neural networks. This library is characterised by its execution speed and memory allocation. It is constantly optimised and parallelised to maximise its efficiency. [AIT20]

5.5. Special Libraries

When programming the neural network, some predefined libraries are accessed. are used. In the following, some interesting libraries are presented.

5.5.1. glob

The glob module finds all pathnames in a given directory which match a given pattern and returns them in an arbitrary way. [Fou20a]

5.5.2. os

The os module provides general operating system functionalities. The functions of this library allow programmes to be designed in a platform-independent way. platform-independent. The abbreviation os stands for Operating System. It enables access to and reference to certain paths and the files stored there. [Bal19]

5.5.3. PIL

The abbreviation PIL stands for Python Image Library and provides many functions for image processing. image processing. It provides fast data access to the basic image formats. image formats is guaranteed. In addition to image archiving and batch function applications file formats, create thumbnails, print images and perform many other functions. many other functions can be performed.

5.5.4. LCC15

5.5.5. math

As can be easily inferred from the name, this library provides access to all mathematical functions defined in the C standard. These include among others trigonometric functions, power and logarithm functions and angle functions. tions. Complex numbers are not included in this library. [Fou20b]

5.5.6. cv2

With the cv2 module, input images can be rewritten into three-dimensional arrays. OpenCV contains algorithms for image processing and machine vision. The algorithms are based on the latest research results and are continuously being further developed. The modules from image processing include, for example, face recognition, as well as many fast filters and functions for camera calibration. The machine vision modules include boosting (automatic classification), decision tree learning and artificial neural networks. [Wik20]

5.5.7. random

The Random module implements pseudo-random number generators for different distributions. distributions. For example, random numbers can be generated or a mixture of elements can be created. of elements can be generated. [Fou20c]

5.5.8. pickle

The pickle module implements binary protocols for serialising and deserialising a Python object structure. of a Python object structure. Object hierarchies can be converted in a binary stream (pickle) and then (pickle) and then be returned from a binary architecture back to the object hierarchical structure (unpickle). structure (unpickle). [Fou20d]

5.5.9. PyPI

The Python Software Foundation organisation organises and manages various packages for Python programming. [Fou21a] Here you can find packages for various tasks. An example is the official API for accessing the dataset COCO. [Fou21b]

6. Libraries, Modules, and Frameworks

Libraries, Module, and Framework have an inevitable role in Machine learning and computer vision field. They are written on a specific topic, which have build-in functions and classes who help the developer and data scientist to run usefull application. Some usefull libraries, modules, and framework use in Machine learning (ML) and Computer Vision(CS) are:

- Tensor Flow
- MediaPipe
- OpenCV
- Pyfirmata

6.1. TensorFlow Framework

TensorFlow is a machine learning system that operates at large scale and in heterogeneous environments. It uses dataflow graphs to represent computation, shared state, and the operations that mutate that state. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations. It maps the nodes of a dataflow graph across many machines in a cluster, and within a machine across multiple computational devices, including multicore CPUs, general-purpose GPUs, and custom-designed ASICs known as Tensor Processing Units (TPUs). TensorFlow enables developers to experiment with novel optimizations and training algorithms. TensorFlow supports a variety of applications, with a focus on training and inference on deep neural networks. Several Google services use TensorFlow in production, we have released it as an open-source project, and it has become widely used for machine learning research. In this paper, we describe the TensorFlow dataflow model and demonstrate the compelling performance that TensorFlow achieves for several real-world applications.

6.1.1. TensorFlow Use Case

Tensor flow is the framework made by google for making the Machine learning application and training the model. Google search engine also based on Tensor flow AI application, e.g; when a user type "a" in the google search bar, the search engine predict the most suitable complete word to select, or it may be predicts depends upon the previous search on the same system by the user too. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. Tensor flow has a various collection of workflows to develop and train the model using Python and Javascripts programming languages, after training the model we can deploy the model on the Cloud or also on any device for making edge computing application no matter which language use in the device. It is obvious that, the most important part in every Machine learning application is to train the ML model, so the model will apply in real time condition and take the decision without any human intervention as the human normally make.

6.2. OpenCV

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing. It plays a major role in real-time operation which is very important in today's systems. Computer Vision is the science of programming a computer to process and ultimately understand images and video, or simply saying making a computer see. Solving even small parts of certain Computer Vision challenges, creates exciting new possibilities in technology, engineering and even entertainment. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When we create applications for computer vision that we don't want to build from scratch we can use this library to start focusing on real world problems. There are many companies using this library today such as Google, Amazon, Microsoft and Toyota. OpenCV.

6.2.1. Application of OpenCV

There are lots of applications which are solved using OpenCV, some of them are listed below.

- Face recognition
- Automated inspection and surveillance
- Number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- Object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

6.2.2. Image Processing

Image Processing is one of the basic functionality of OpenCV, it is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it.

6.2.3. How does a computer read an image

We are humans we can easily make it out that is the image of a person who is me. But if we ask computer "is it my photo?". The computer can't say anything because the computer is not figuring out it all on its own. [Image Processing] The computer reads any image as a range of values between 0 and 255. For any color image, there are 3 primary channels -red, green and blue.

6.3. MediaPipe

MediaPipe is one of the most widely shared and re-usable libraries for media processing within Google. Google open-source MediaPipe was first introduced in June, 2019. It aims to provide some integrated computer vision and machine learning features. It has some build in module who can detect the Human motion by tracking the different part of the body. This Github link [MediaPipe Github] shows all the available application the MediaPipe Module able to perform. It is best suited for gesture detection for edge computing application.

6.4. MediaPipe Applications

MediaPipe has numerous application in Machine learning and computer vision. It is a python supportive library and perform well for edge devices too. The module is train on 30,000 images, so its accuracy and precision is very high. The reason for using MediaPipe is that, it work well for independent of background, because it detects just the Landmarks of different part of body. The remaining things in the frame is invisible for the MediaPipe Module. The Following most important use cases of MediaPipe are:

- Hand Landmarks
- Face Detection
- Object Detection
- Holistic

6.4.1. Hand Landmarks Detection

The ability to perceive the shape and motion of hands can be a vital component in improving the user experience across a variety of technological domains and platforms. Every hand there are 21 landmarks, each finger has 4 and there is landmark in the middle of hand. For gesture detection we can use this technique to make the different types of gesture. We are able to make different types of gestures by changing the position of landmarks. MediaPipe Hands is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame. Whereas current state-of-the-art approaches rely primarily on powerful desktop environments for inference, our method achieves real-time performance on a mobile phone, and even scales to multiple hands. Fig6.1 shows all the Hand Landmarks of a hand, which can help us to make different gestures. [Hand Landmarks]

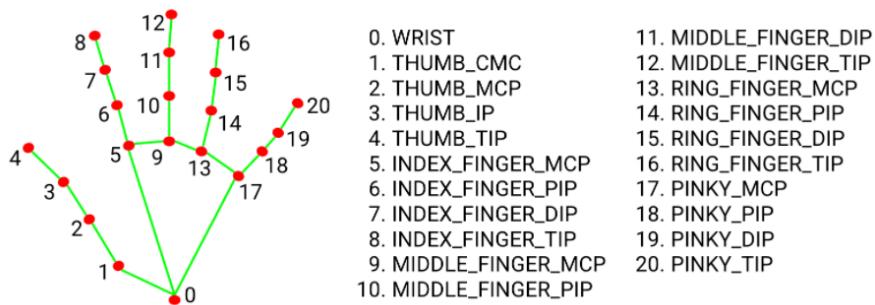


Figure 6.1.: Hand Landmarks

6.4.2. Face Detection

MediaPipe Face Detection is an ultrafast face detection solution that can detect 6 landmarks on the face and also support multiple faces. Fig6.2 shows the detection of multiple faces using landmarks. The detector's super-realtime performance enables it to be applied to any live viewfinder experience that requires an accurate facial region of interest as an input for other task-specific models, such as 3D facial keypoint or geometry estimation (e.g., MediaPipe Face Mesh), facial features or expression classification, and face region segmentation. Face Detection

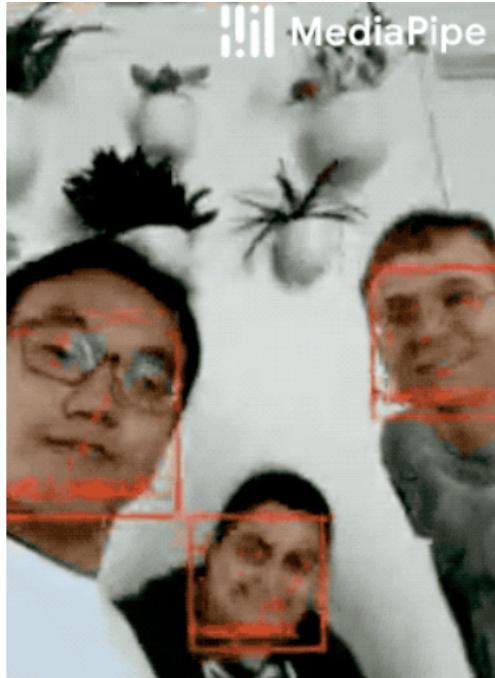


Figure 6.2.: Face Detection Using Landmarks

7. Gesichtserkennung

Gesichtserkennung ist eine auf künstlicher Intelligenz basierende Computertechnologie, die zur Erkennung menschlicher Gesichter in Bildern verwendet wird. Sie bestimmt die Position und Größe menschlicher Gesichter in einem Bild oder Videostream und sollte, nachdem das Gesicht erkannt wurde, eine Bounding Box der erkannten Gesichter zurückgeben. Die Gesichtserkennung wird in vielen Anwendungen wie biometrischen Sicherheitssystemen verwendet, in denen es möglich ist, Personen in Echtzeit zu verfolgen und zu überwachen.

Es gibt zwei Arten von Ansätzen für die Erkennung eines Gesichts in einem Bild:

1. Merkmalsbasierter Ansatz
2. Bildbasierter Ansatz.

7.0.1. Merkmalsbasierter Ansatz

Bei dieser Art von Methode werden die verschiedenen Merkmale des Gesichts im Bild gefunden, z. B. die Erkennung von Augen, Nase, Augenbrauen und Mund. Da diese Merkmale in einem Bild, das ein Gesicht enthält, trotz anderer Variabilitäten wie Beleuchtung und Posen konsistent sind. Die Hauptidee dieses Ansatzes besteht also darin, diese Merkmale mit Hilfe von Kantenerkennungstechniken zu extrahieren und auf der Grundlage dieser extrahierten Merkmale statistische Modelle zu erstellen, um die Beziehung zu beschreiben und das Vorhandensein eines Gesichts in einem Bild zu erkennen.

7.0.2. Bildbasierter Ansatz

Die bildbasierte Methode versucht, Vorlagen aus Beispielen in Bildern zu lernen. Das heißt, diese Methode stützt sich auf maschinelles Lernen und statistische Analysetechniken, um die Gesichtsmerkmale zu finden und zu erklären, ob das Bild ein Gesicht enthält oder nicht. Bei dieser Methode werden neuronale Netze für die Erkennung verwendet.

7.0.3. Herausforderungen

Die Herausforderungen bei der Erkennung von Gesichtern sind die Gründe für eine geringere Genauigkeit und Erkennungsrate in Bildern. Einige der Herausforderungen sind:

- Beleuchtungsbedingungen:

Die Beleuchtung in der Umgebung eines Bildes kann in Teilen des Bildes schwach oder stark sein, was die Erkennung von Gesichtern erschwert.

- Abstand:

Wenn der Abstand der Kamera zu einem Gesicht sehr gering oder groß ist, wird die Erkennung ebenfalls schwierig.

- Orientierung:

Die Ausrichtung des Gesichts und der Winkel zur Kamera können ebenfalls zu Erkennungsfehlern führen.

- Komplexer Hintergrund:

Wenn der Hintergrund des Bildes eine größere Anzahl von Objekten enthält, wird die Erkennung zu einer schwierigen Aufgabe.

- Niedrige Auflösung:

Wenn die Bildauflösung sehr niedrig ist oder das Bild Rauschen enthält, ist die Genauigkeit der Gesichtserkennung geringer.

7.0.4. Lösungen

- Optimale Beleuchtungsbedingungen sollten berücksichtigt werden, wenn man versucht, ein Gesicht in einem Bild zu erkennen. Ein gut und gleichmäßig ausgeleuchtetes Bild erleichtert die Erkennung von Gesichtern in einem Bild.
- Der Abstand zwischen der Kamera und dem Gesicht sollte optimal sein. In unserem Fall ist ein Abstand zwischen 7cm-10cm am besten für die Gesichtserkennung geeignet.
- Die Anzahl der Trainingsbilder mit verschiedenen Winkeln und Ausrichtungen sollte genommen werden, um die Erkennungsrate in Bildern mit verschiedenen Ausrichtungen von Gesichtern zu verbessern.
- Der Hintergrund des Bildes sollte auch so berücksichtigt werden, dass das zu erkennende Gesicht nicht durch eine größere Anzahl von Objekten im Bild verdeckt wird.
- Die Verwendung modernster Methoden wie Faltungsneuronale Netze (CNN) zur Gesichtserkennung trägt erheblich zur Erkennungsrate bei.

7.0.5. Anwendungen

- Qualitätsprüfung:

Prüfung der Qualität von Fertigprodukten in einer Produktionsanlage

- Überwachung:

Es wird zur Erkennung von Gesichtern in einer Menschenmenge, z. B. an einem öffentlichen oder privaten Ort, verwendet.

- Erwartung:

Es kann verwendet werden, um die Anwesenheit von Menschen in einer Klasse zu erkennen, und wenn es mit einem biometrischen Sicherheitssystem kombiniert wird, kann es auch für die Zugangsverwaltung in einem Gebäude verwendet werden.

- Fotografie:

Handykameras erkennen Gesichter für den Autofokus in einem Bild.

Part II.

Tools

8. Arduino IDE 1.8.x

8.1. Introduction

In this chapter we will be going through the description of the Arduino IDE and describe the features of it and what are all the options present in the IDE and how can we use it. Further we will be describing about the installation procedure of the Arduino IDE and how to get started with it.

8.2. Arduino IDE Description

It is an open source official Arduino software which used for editing, uploading and compiling codes in to the Arduino module. It is a cross-platform software which is available for Operating Systems like Windows, Linux, macOS. It runs on Java platform and supports a range of Arduino modules. It supports C and C++ languages. The microcontrollers present on the Arduino boards are programmed which accepts the information in the form of code. The program written in the IDE is called a sketch which will generate a Hex file which is then transferred and uploaded in the controller. The IDE environment is made up of two parts: an editor and a compiler. The editor is used to write the required code, while the compiler is used to compile and upload the code to the Arduino Module.[FAD18]

The Menu bar has options such as File in which there are many options including Opening a new file or existing, Examples-in which we can find sketches for different applications like Blink, Fade etc. There is an error console at the bottom of the screen for displaying errors.

The 6 buttons are present on top of the screen are presented in figure 8.2.

- The check mark is used to verify your code. Click this once you have written your code.
- The arrow uploads your code to the Arduino to run.
- The dotted paper will create a new file.
- The upward arrow is used to open an existing Arduino project.
- The downward arrow is used to save the current file.
- The far right button is a serial monitor, which is useful for sending data from the Arduino to the PC for debugging purposes.

8.3. Installation

To install the Arduino IDE, we need to download the latest version from the Arduino webpage <https://www.arduino.cc/en/software>. We can select the version based on the operating system we are using. Here we are installing Arduino 1.8..15 for a Windows 10 operating system. The set up file name is arduino-1.8.15-windows.exe and the size of it is 1,17,470 KB. we can specify the path according to our needs. Here the path is set as **C:/Program Files (x86)/Arduino**. The most recent offline arduino IDE 1.8.15 can be seen in Figure. 8.3 it is also supportive for all operating systems.

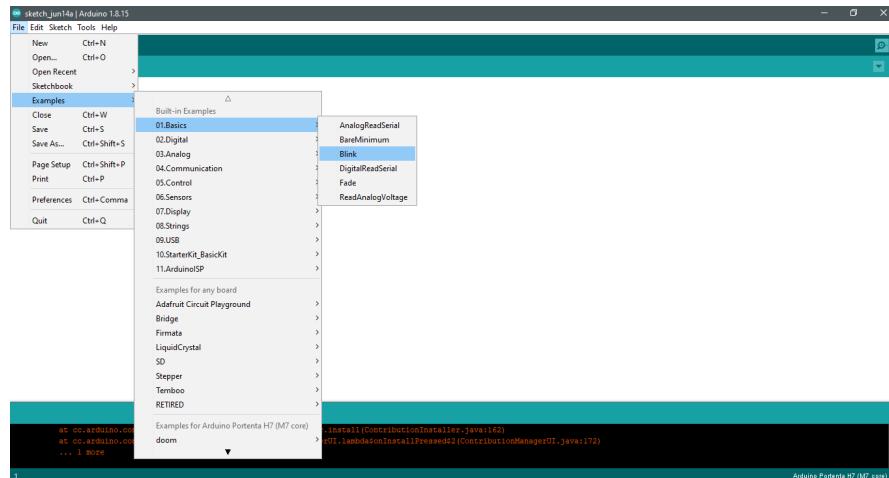


Figure 8.1.: Menu bar options



Figure 8.2.: Menu buttons

After the download is done, open the setup file and proceed to install. Select all the components in the dialog box and click Next.

Select the destination folder and click Install

Once the installation is done, open the Arduino IDE and a default sketch appears on the screen as shown.

It can be seen from the figure 8.6 that the basic arduino sketch has two parts. The first part is the function **void setup()** which returns void and we do the intiliaztion such as the output LED color, specifying the core etc. The second part is the function **void loop()** where we define functions which are to be performed through out the loop. These codes are placed between paranthesis {} and each function has a return type, here it has void return type.



Figure 8.3.: Arduino Creat Agent Installation

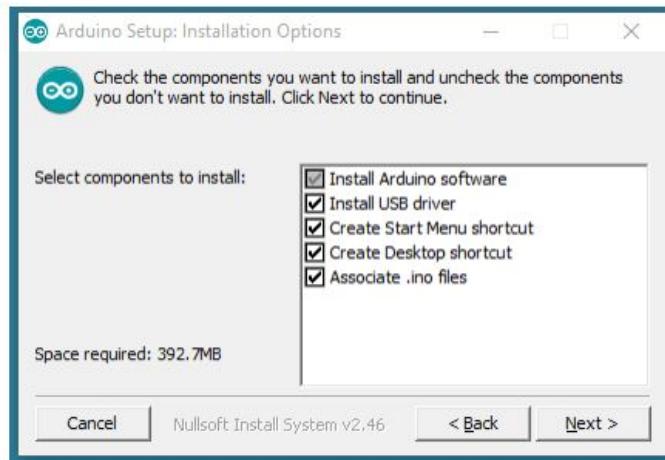


Figure 8.4.: Arduino Setup Installation options

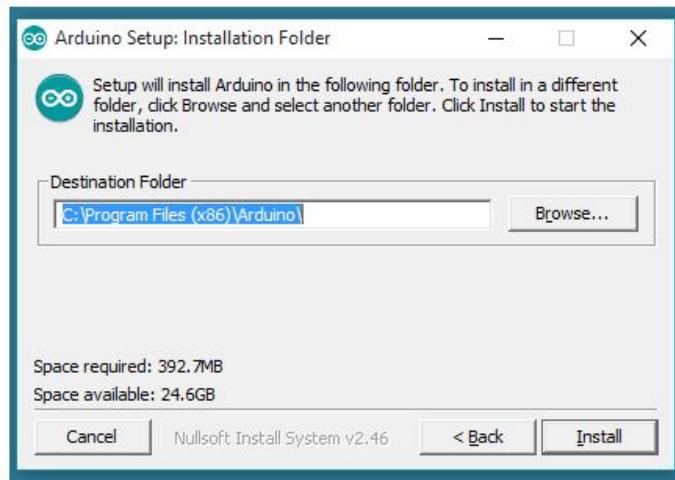


Figure 8.5.: Arduino Setup: Installation Folder

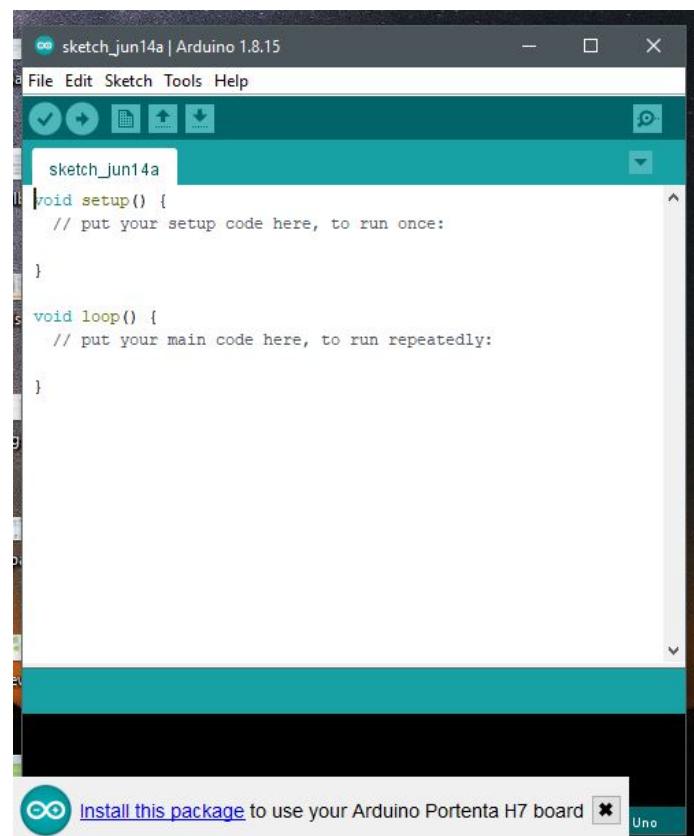


Figure 8.6.: Arduino Sketch

9. First Steps with the Portenta H7

9.1. Introduction

In this chapter we will be looking at how to connect the Arduino Portenta H7 with a PC/laptop in order to use the board. Then, we will be going through the first steps of installing the relevant packages to use the Arduino Portenta H7 board and then we will be implementing a basic example sketch from the Arduino IDE.

9.2. Configuration

Connect the Arduino Portenta H7 board to the computer via USB Type-C cable. Press the reset button twice on the board and the LED on the board starts blinking green as shown in the figure 20.1 indicating that it is ready.

First we need to install the packages in order to use the Arduino Portenta board. To do this search for Portenta in the boards and select Arduino Mbed OS Portenta Boards and click install as shown in figure 20.2. After the installation is done, we can start writing sketches in the IDE.

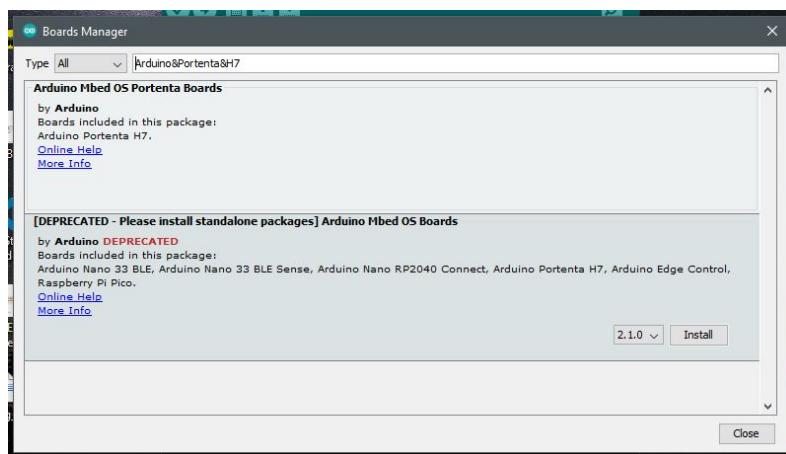


Figure 9.2.: Install packages for Arduino Portenta H7

9.2.1. Example Blink Sketch

To upload the example sketch, go to File then click on examples and then click on 01.Basics and then select blink as shown in figure 20.3.

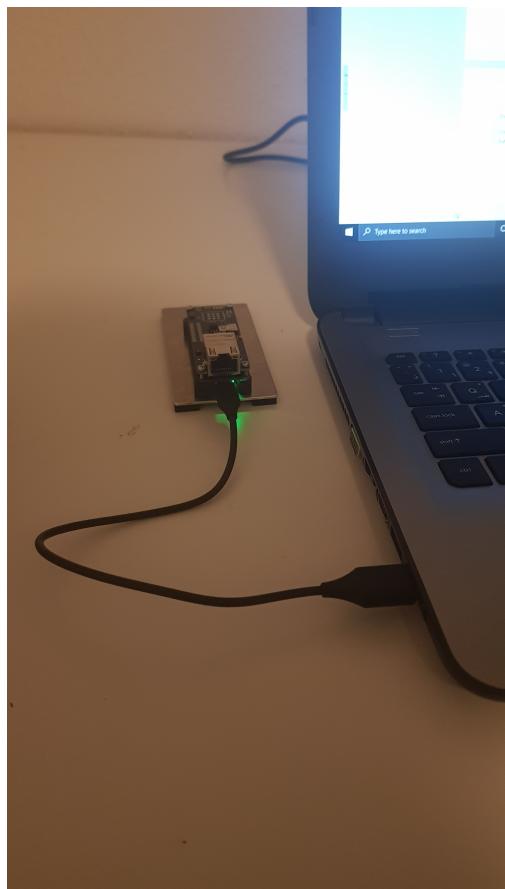


Figure 9.1.: Arduino Portenta H7 Connected to a laptop

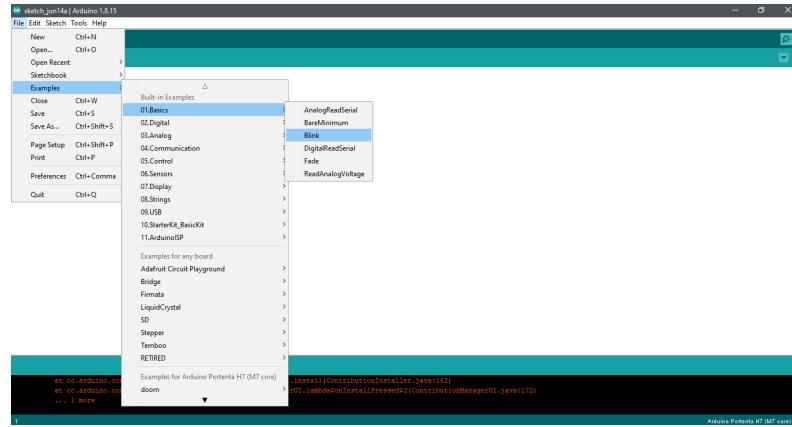


Figure 9.3.: Upload Basic Sketch

In the program, we initialize LED_BUILTIN as the output in the setup function. In the loop function we turn on the LED by using digitalWrite() function. Click on upload and wait for it to complete. After the upload is done, the green LED on the board starts blinking with a delay of 1000ms. The blink sketch appears on the screen as shown in figure 20.4.

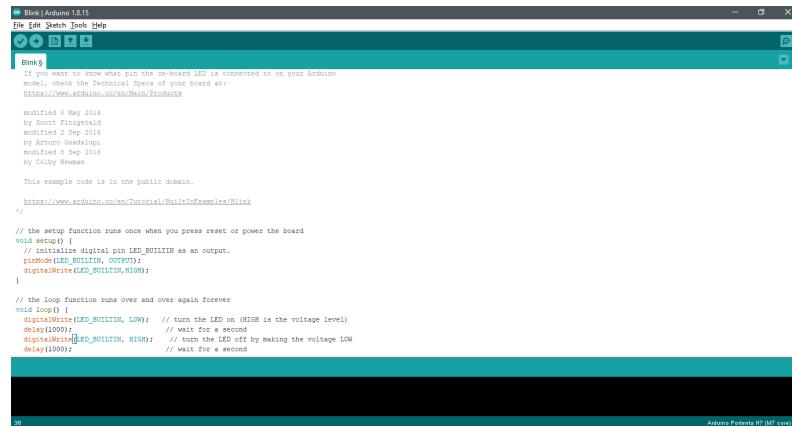


Figure 9.4.: Compile Blink Sketch

9.2.2. Serial Blink Example for Dual Core Processing

The following figure 20.5 represents the serial blink sketch. This sketch helps in demonstrating the serial data transmission by showing the LED blinking after a certain time period.

Here, Serial.begin(115200) command sets the data transmission rate to 115200 bits per second. The LEDB is the output which blinks in blue color and we have specified core M7. If we would like to use core M4 then we need to boot it first using bootM4() command. The ouput LED blinks blue in color after a delay of 3 seconds as defined in the sketch.

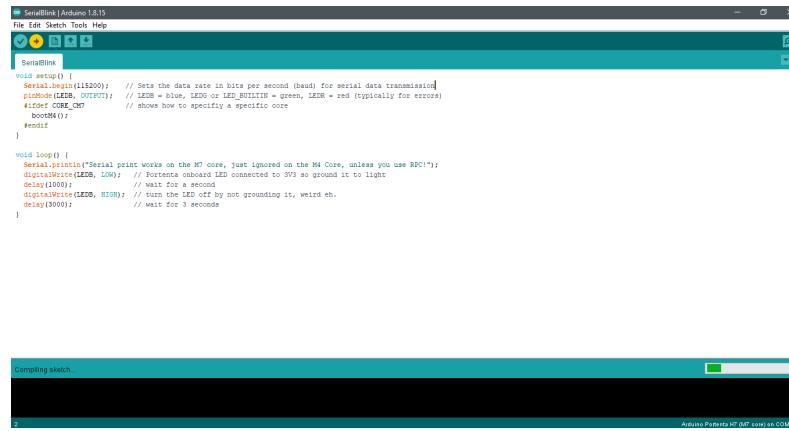


Figure 9.5.: Serial Blink Sketch

In order to use core M4, we need to de-select the core M7 from the board and select core M4. After selecting, we can see at the bottom the selected core for confirmation. Also, We need to write another sketch and now change the LED color to red in the sketch and add a delay of 4 seconds.

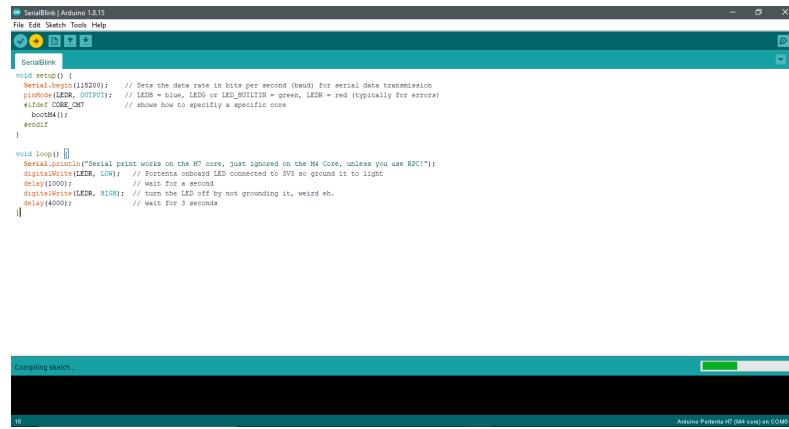


Figure 9.6.: Serial Blink Sketch

Notice that in the output both LED blue and green blinks and at a point in time they almost blink together resulting in a Lila color. This shows the dual core processing concept by using an LED.

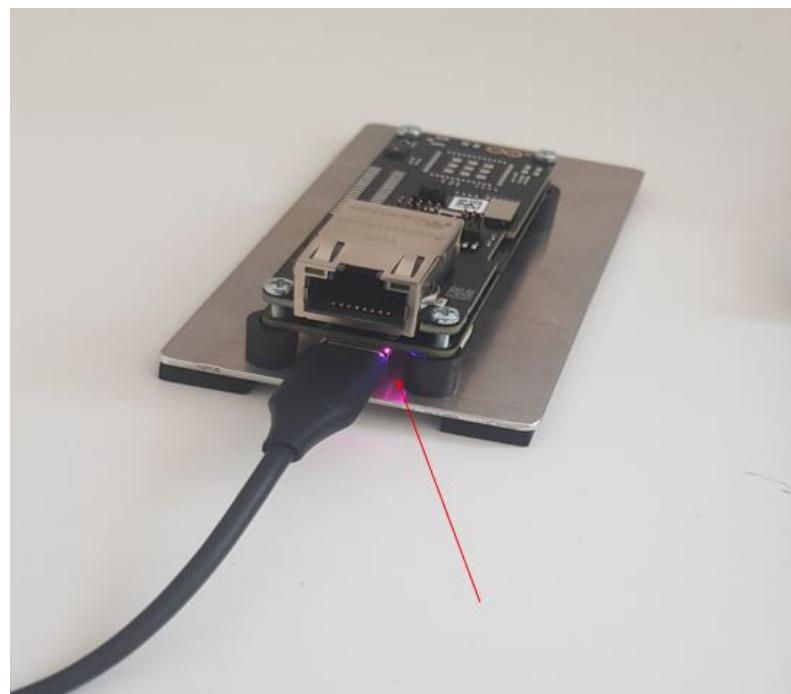


Figure 9.7.: Output LED color

It can be seen from the figure 20.7 that the output LED blinks in lila color due to the overlapping of the green and blue LED blinking at the same time.

10. Kommandozeileninterpreter (CLI)

Quellen für die Bilder fehlen!

Seit 2018 bietet die Firma Arduino auch einen Kommandozeileninterpreter an. [Ard] Die Web-Version der Entwicklungsumgebung verwendet diese Schnittstelle. Das heißt, alle Funktionalitäten, die die Entwicklungsumgebung zur Verfügung stellt, stehen hier auch zur Verfügung. So kann unter Windows, MacOS und Linux [Ard22c]. Die Entwicklung mit Hilfe des Kommandozeileninterpreters ermöglicht es, wenn konsequent mit Hilfe von Batch-Dateien gearbeitet wird, qualitätssichernde Maßnahmen gezielt durchzuführen. So kann die Konfiguration der Entwicklungsumgebung gesichert und nachvollziehbar wiederholt werden.

WS:Erklärung von
Kommandos in den
Anhang

Der Quellcode kann frei genutzt werden, allerdings muss man bei einer gewerblichen Nutzung eine Lizenz bei der Firma anfragen [Ard22a]. Eine ausführliche Dokumentation wird zur Verfügung gestellt. [Ard20; Ard22b]

Bei der Entwicklung der Command Line Interface (CLI) stehen drei Aspekte im Fokus.

1. Einerseits ermöglicht die Schnittstelle die Einbettung der Entwicklung in die gewohnte Entwicklungsumgebung. So kann die Schnittstelle auch mit Edge Impulse genutzt werden.
2. Zweitens ermöglicht sie die Integration der Entwicklung in die Prozesse Continuous Development and Continuous Integration. Sie ermöglicht damit die Automatisierung der typischen Aktivitäten im Bereich der Softwareentwicklung.
3. Des Weiteren ermöglicht es nun die vereinfachte Kommunikation mit Edge Computern, zum Beispiel mit einem Raspberry PI.

10.1. Installation und Verwendung des CLI

Da CLI ständig weiterentwickelt wird, muss zunächst die aktuelle Version aus dem GitHub-Projekt <https://github.com/arduino/arduino-cli> geladen werden. Die hier verwendete Version ist 0.33. [Ard]

Nach dem erfolgreichen Download der Datei `arduino-cli_0.33.0_Windows_64bit.zip` kann der Ordnerinhalt an einen selbst definierten Speicherpfad extrahiert werden. Die gepackte Datei enthält die Lizenzbestimmungen und die ausführbare Datei `arduino-cli.exe`. Um die Funktion des Kommandozeileninterpreters in verschiedenen Pfaden nutzen zu können, sollte der Pfad zum Speicherort der Datei `arduino-cli.exe` der Systemvariablen `PATH` hinzugefügt werden. Anschließend lässt sich das Interface durch die Eingabe `arduino-cli` in die Kommandozeile nutzen.

Nachdem die Installation abgeschlossen ist, kann `arduino-cli board list` in der Eingabeaufforderung eingegeben werden. Das angeschlossene Gerät wird als Arduino Portenta H7 mit Portnummer und Typ angezeigt, wie in Abbildung 10.1 dargestellt. Es ist zu beachten, dass die Versionsnummer, hier 0.33.1, gegebenenfalls anzupassen ist. Wenn ein Arduino Nicla Vision oder ein Arduino Nano 33 BLE sense angeschlossen ist, werden entsprechende Meldungen ausgegeben.

Das Programm `arduino-cli-0.33.1.windows.exe` wird nun in `arduino-cli.exe` umbenannt. Daher kann im Folgenden im der Programmname `arduino-cli.exe` verwendet werden.

```
C:\Users\Shoeb>arduino-cli board list
Port Type      Board Name          FQBN                         Core
COM6 Serial Port (USB) Arduino Portenta H7 (M7 core)    arduino:mbed:envie_m7      arduino:mbed
                                         Arduino Portenta H7 (M7 core)    arduino:mbed_portenta:envie_m7  arduino:mbed_portenta
                                         Arduino Portenta H7 (ThreadDebug) arduino:mbed:envie_m7_thread_debug  arduino:mbed

C:\Users\Shoeb>
```

Figure 10.1.: Installation von Arduino-CLI

10.2. Konfiguration des Arduino CLI

<https://learn.sparkfun.com/tutorials/efficient-arduino-programming-with-arduino-cli-and-yaml-introduction-to-the-arduino-cli>

Damit CLI dir Arduino-Installation finden kann, hilft es, eine Konfigurationsdatei für CLI zu erstellen. Diese Konfigurationsdatei ist in einem Format YAML definiert. Zur Erstellung einer Basis-Konfigurationsdatei kann CLI verwendet werden, in dem in einer Kommandozeile

`arduino-cli.exe config init`

eingegeben wird. Dieser Befehl erstellt eine neue Datei `.cli-config.yml`. Dort sind alle notwendigen Parameter deklariert. Die Standardeinstellungen der Konfigurationsdatei sind in Abbildung 10.2 dargestellt. Folgende Parameter müssen in der Regel geändert werden:

- `sketchbook_path`: Verzeichnis des Arduino-Sketche. Hier werden alle Bibliotheken und Hardware-Definitionen installiert.
- `arduino_data`: Installationsort des Arduino-Boards und des Bibliotheksmanagers. In den meisten Fällen sollte dies nicht geändert werden müssen.

Die anderen Optionen können in der Regel auf ihren Standardwerten bleiben.

S: Welche Möglichkeiten gibt es in der Konfigurationsdatei?

10.3. Funktionsübersicht

Die Datei `README.md` im GitHub-Repository “Arduino CLI” enthält eine hervorragende Übersicht über die Funktionen und Möglichkeiten.[Ard22a] Folgende Funktionen stehen zur Verfügung [Ard]:

- `arduino-cli`
- `board`
 - `board attach`
 - `board details`
 - `board list`
 - `board listall`
 - `board search`

```
board_manager:
  additional_urls: []
build_cache:
  compilations_before_purge: 10
  ttl: 720h0m0s
daemon:
  port: "50051"
directories:
  data: C:\Users\denni\AppData\Local\Arduino15
  downloads: C:\Users\denni\AppData\Local\Arduino15\staging
  user: C:\Users\denni\Documents\Arduino
library:
  enable_unsafe_install: false
logging:
  file: ""
  format: text
  level: info
metrics:
  addr: :9090
  enabled: true
output:
  no_color: false
sketch:
  always_export_binaries: false
updater:
  enable_notification: true
```

Figure 10.2.: Standardeinstellungen der Konfigurationsdatei

- **burn-bootloader**
- **cache**
- **cache clean**
- **compile**
- **completion**
- **config**
 - **config dump**
 - **config init**
 - **config add**
 - **config delete**
 - **config remove**
 - **config set**
- **core**
 - **core download**
 - **core install**
 - **core list**
 - **core search**
 - **core uninstall**
 - **core update-index**
 - **core upgrade**
- **daemon**
- **debug**

- **lib**
 - **lib deps**
 - **lib download**
 - **lib examples**
 - **lib install**
 - **lib list**
 - **lib search**
 - **lib uninstall**
 - **lib update-index**
 - **lib upgrade**
- **monitor**
- **outdated**
- **sketch**
 - **sketch archive**
 - **sketch new**
- **update**
- **upgrade**
- **upload**
- **version**

10.3.1. Grundfunktionen

Als Grundfunktionen werden die Funktionen definiert, die notwendig sind, um einen fertigen Sketch auf einen Arduino zu laden. Die Kommunikation mit einem angeschlossenen Arduino wird später mit Hilfe des seriellen Monitors umgesetzt.

Die erste Grundfunktion ist der Befehl **board list**. Mit diesem lassen sich alle am PC angeschlossenen Arduino Boards mit zusätzlichen Informationen – wie beispielsweise dem verwendeten Port oder dem Kern des Boards – anzeigen. Diese Informationen sind für die Verwendung des korrekten Boards notwendig.

Die Befehle **core list** und **core install** sind ebenfalls notwendig. Mit dem Befehl **core list** können die bereits installierten Kerne aufgelistet werden. Falls der unter **board list** angegebene Kern nicht installiert sein sollte, lässt sich dieser mit Befehl **core install** hinzufügen.

Die dritte Grundfunktion ist der Befehl **lib** mit den Erweiterungen **lib list** und **lib install**. Mit dem Befehl **lib list** lässt sich prüfen, welche Bibliotheken auf dem System bereits installiert sind. Würde für einen Sketch eine zusätzliche Bibliothek benötigt, könnte diese mit dem Befehl **lib install** installiert werden.

Mit diesen Grundfunktionen können die Vorbereitungen für das Kompilieren und Hochladen eines Sketches getroffen werden. Mit der Ausführung des Befehls **compile** wird ein Sketch in für ein ausgewähltes Board lesbaren Code übersetzt. Das anschließende Hochladen eines kompilierten Sketches auf ein Board erfolgt mit dem Befehl **upload**. Dabei wird der aus dem Befehl **board list** bekannte Port des Boards als Ziel für den Upload genannt. Die Grundfunktionen sind in der Tabelle 10.1 zusammengefasst.

Table 10.1.: Liste der Grundfunktionen

Nr.	Grundfunktion	Beschreibung
1	<code>board list</code>	Auflisten der angeschlossenen Arduinos mit Zusatzinfo
2	<code>core list/install</code>	Auflisten und Installieren von Kernen
3	<code>lib list/install</code>	Auflisten und Installieren von Bibliotheken
4	<code>compile</code>	Kompilieren eines Sketches für ein spezielles Board
5	<code>upload</code>	Hochladen eines Sketches auf einen Arduino

10.4. Erste Schritte mit dem Arduino Nano 33 BLE Sense Lite mit Hilfe des CLI

10.4.1. Erkennung der angeschlossenen Boards

Mit Hilfe von CLI kann abgefragt werden, welche Boards installiert sind:

```
arduino-cli board listall
```

Falls das angeschlossenes Board vermisst wird, muss es installiert werden:

```
arduino-cli core install arduino:avr
```

Mit Hilfe von CLI kann abgefragt werden, welche Boards angeschlossen sind:

```
arduino-cli board list
```

So kann nach dem Anschließen des Arduinos an einen PC mit dem Befehl „arduino-cli board list“ geprüft werden, ob das Board von dem Computer erkannt wird. Die Funktion „board list“ besitzt mehrere Rückgabewerte. So wird der Port, über den das Board mit dem PC verbunden ist, das Protokoll, der Typ, der Platinenname und der FQBN sowie Kern, wie in Abb. 10.3 dargestellt, zurückgegeben.

```
C:\Users\denni>arduino-cli board list
Port Protokoll Typ Platinenname FQBN Kern
COM4 serial Serial Port (USB) Arduino Nano 33 BLE arduino:mbed_nano:nano33ble arduino:mbed_nano
```

Figure 10.3.: Rückgabewerte der „board list“-Funktion

```
C:\Users\denni>arduino-cli core install arduino:mbed_nano|
```

Figure 10.4.: Installieren des Kerns für den Arduino Nano 33 BLE Sense Lite

Zu Beginn sollte der richtige Kern für den Arduino Nano 33 BLE Sense Lite, wie in Abb. 10.4 gezeigt, installiert werden. Dies erfolgt mit dem „core install“-Befehl und dem richtigen Kernnamen, der aus Abbildung 10.3 entnommen werden kann.

10.4.2. Erstellung eines neuen Sketches

Der erste Schritt ist die Erstellung eines neuen Sketches:

```
arduino-cli sketch new cli_test
```

Dieser Befehl erstellt ein Verzeichnis mit dem Namen `cli_test`, die eine Datei mit dem gleichen Namen enthält.

```

Select C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\[REDACTED] >arduino-cli-0.2.2-alpha.preview-windows.exe compile --fqbn arduino:avr:uno C:\Users\[REDACTED]\Downloads\Arduino\cli_test
Sketch uses 444 bytes (1%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.

C:\Users\[REDACTED]\Downloads>

```

Figure 10.5.: Kompilieren eines Sketches mit CLI

10.4.3. Kompilieren eines Sketches

Die Funktion „compile“ von CLI kann verwendet werden, um einen Sketch für jedes unterstützte Board zu kompilieren. Die entscheidende Option, die diese Funktion benötigt, ist der Board-Typ, der mit der Option **-fqbn** angegeben werden kann. Die Abkürzung **fqbn** bedeutet „fully-qualified board name“, was „vollqualifizierter Board-Name“ bedeutet. Beispielsweise stehen folgende Boards zur Verfügung:

- Arduino Uno: **arduino:avr:uno**
- Arduino Mega: **arduino:avr:mega**
- SparkFun RedBoard oder SparkFun BlackBoard: **SparkFun:avr:RedBoard**; dies erfordert die zusätzliche Installation der Definition „SparkFun avr board“.
- SparkFun SAMD21 Mini: **SparkFun:samd:samd21_mini**; dies erfordert die zusätzliche Installation der Definition „SparkFun samd board“.

Die möglichen Boards sind wie folgt aufgebaut: **manufacturer:architecture:board**.

Mit dem folgenden Kommando kann der Beispiel-Sketch, der sich im Ordner **C:/Users/user.name/Documents/Arduino** befindet, für einen Arduino UNO kompiliert werden.:

arduino-cli compile -fqbn arduino:avr:uno C:/Users/user.name/Documents/Arduino/cli-test

In der Abbildung 10.5 sieht man die Meldungen von CLI.

Es können folgende Flags hinzugefügt werden:

- Ausführlich **-v**: Nützlich, wenn alle Optionen und Dateien angezeigt werden sollen, die in Ihrem Sketch kompiliert werden.
- Build Pfad **-build-path [string]**: Nützlich, wenn die kompilierten Objekt- und Hex-Dateien zu speichern sind. Auf meinem Windows-System muss der Wert dieses Parameters ein vollständiger Pfad sein.

10.4.4. Hochladen eines Sketches

Ein kompilierter Sketch kann hochgeladen werden. Wie der Kompilierbefehl erfordert auch der Upload-Befehl eine **-fqbn**. Außerdem wird eine serielle Schnittstelle zum Hochladen benötigt, die mit der Option **-p** festgelegt wird. Der folgende Befehl lädt den Beispiel-Sketch an einen Windows-COM-Anschluss auf COM18 hoch:

arduino-cli upload -p COM18 -fqbn -v arduino:avr:uno C:/Users/user.name/Documents/Arduino/cli-test

Falls es ordnungsgemäß durchgeführt wurde, sollte die RX/TX-LEDs des Arduinos zu blinken beginnen und kurz darauf einen leeren Sketch auszuführen.

WS:Portenta H7 und
Nicla Vision fehlen

10.4.5. Installation von Bibliotheken

Falls eine Bibliothek benötigt wird, so kann zunächst mit dem Kommando

```
arduino-cli lib search ethernet
```

on die Bibliothek, hier `ethernet`, installiert ist. Mit dem Befehl

```
arduino-cli lib install "UIPEthernet"
```

wird sie dann installiert.

10.4.6. Beispiel „Hello World“

Um das Kompilieren eines Sketches sowie das anschließende Hochladen mit Hilfe des Arduino-CLI zu demonstrieren, soll das „Hello World!“-Pendant für Mikrocontroller verwendet werden. Dafür wird ein einfacher Sketch, der die LED des Arduinos zum Blinken bringt, benutzt.

```
void setup(){
    pinMode(LED_BUILTIN, OUTPUT);
}
void loop(){
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
}
```

In der zu Beginn einmalig ausgeführten `setup()`-Funktion wird die eingebaute LED als Output definiert `pinMode(LED_BUILTIN, OUTPUT)`. Dadurch lässt sich die LED später ansteuern. Innerhalb der `loop()`-Funktion wird die LED zunächst eingeschaltet `digitalWrite(LED_BUILTIN, HIGH)` und mit einer Verzögerung von einer Sekunde `delay(1000)` wieder abgeschaltet `digitalWrite(LED_BUILTIN, LOW)`. Vor dem erneuten Einschalten der LED zu Beginn der Schleife wird am Schleifenende nochmal eine Sekunde gewartet um den blinkenden Effekt zu erzielen.
Um den Sketch zu kompilieren, kann der zuvor als Grundfunktion definierte Befehl `compile`, wie in Abb. 10.6 dargestellt, verwendet werden.

```
C:\Users\denni>arduino-cli compile -b arduino:mbed_nano:nano33ble C:\Users\denni\A\Ablage\ArduinoProjekte\blink
```

Figure 10.6.: Für das Kompilieren des Blink-Sketches wird der Dateipfad zu dem Sketch mit angegeben (rot unterstrichen). Außerdem sollte der FQBN des Arduinos, für den der Sketch kompiliert wird, genannt werden.

Anschließend kann der kompilierte Sketch mit Hilfe der „upload“-Funktion auf den Arduino hochgeladen werden. Dafür wird, wie in Abb. 10.7 dargestellt, der Speicherpfad zu dem Sketch angegeben. Dadurch erfolgt das Hochladen der kompilierten Daten für den angegebenen Sketch.

```
C:\Users\denni>arduino-cli upload -p COM5 C:\Users\denni\A\Ablage\ArduinoProjekte\blink
```

Figure 10.7.: Hochladen des kompilierten Sketches aus dem Temp-Ordner

Nach dem erfolgreichen Upload beginnt die in Abb. 10.8 gezeigte LED des Arduino Nano 33 BLE Sense Lite zu blinken.

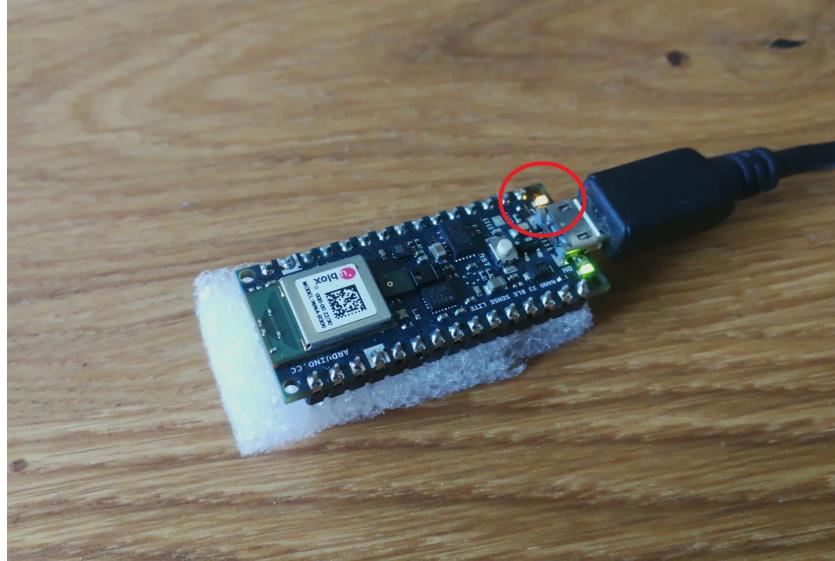


Figure 10.8.: Die orange LED des Arduinos beginnt zu blinken

10.5. Beschreibung der Software auf dem PC

Um einen erhöhten Automatisierungsgrad zu erreichen und die Potentiale des Arduino-CLI zu demonstrieren, wird ein Batch-Skript zum Testen der Sensoren verwendet. Dabei werden die Ergebnisse in einer Log-Datei dokumentiert. In diesem Abschnitt soll die Funktionsweise des Batch-Skriptes beschrieben werden.

```
@echo off
echo Logdatei Arduino-Setup , %time% Uhr , %date% > setup-log.txt
echo. >> setup-log.txt
echo. >> setup-log.txt
```

Mit dem Befehl `echo off` wird die Ausgabe des Skript-Inhaltes bei Ausführung verhindert. Dadurch wird eine ansprechendere Frontend-Programmierung ermöglicht. In der zweiten Zeile wird die Log-Datei erstellt. Beim Erstellen wird mit der Überschrift **Logdatei Arduino-Setup** sowie dem Auslesen der Systemzeit `%time%` und des Datums `%date%` die erste Zeile des Protokolls versehen. Für die spätere Übersichtlichkeit der Log-Datei folgen zwei leere Zeilen `echo. >> setup-log.txt`. Mit dem einmaligen Anwenden des größer als Operators `> setup-log.txt` wird eine neue Datei erstellt oder eine Bestehende mit dem gleichen Namen überschrieben. Bei doppelter Anwendung des größer als Operators `>> setup-log.txt` wird der linksseitige Inhalt in einer neuen Zeile an eine bestehende Datei angehängt. So wird die Log-Datei Zeile für Zeile beschrieben und nicht einmalig komplett am Ende des Batch-Skriptes.

```
:: Gegebenenfalls wird der Kern installiert
echo Kernstatus: >> setup-log.txt
arduino-cli core install arduino:mbed_nano >> setup-log.txt
```

Kommentare in dem Batch-Skript werden durch zwei aufeinanderfolgende Doppelpunkte gekennzeichnet `:: Gegebenenfalls wird der Kern installiert` und sollen die Nachvollziehbarkeit innerhalb des Skriptes erhöhen. Mit dem Befehl `echo Kernstatus: >> setup-log.txt` wird in der Log-Datei gekennzeichnet, dass in der nächsten Zeile Informationen über den zu installierenden Kern für den Arduino Nano 33 BLE Sense Lite folgen. Das Arduino-CLI bringt bereits die Intelligenz mit zu prüfen, ob der angegebene Kern schon installiert ist. Sollte dies der Fall sein, erfolgt die

entsprechende Dokumentation in der Log-Datei. Das Installieren des Kerns wird mit der Zeile `arduino-cli core install arduino:mbed_nano >> setup-log.txt` initialisiert. Dabei ist der für den Arduino Nano 33 BLE Sense Lite notwendige Kern mit dem Namen `arduino:mbed_nano` angegeben. Der Rückgabewert der Funktion zum Kerninstallation wird in der Log-Datei gespeichert.

```
echo Es wird nach angeschlossenen Boards gesucht ...
:: Auflisten der angeschlossenen Arduinos
echo Folgende Boards sind am PC angeschlossen: >> setup-log.txt
echo .
echo . >> setup-log.txt
arduino-cli board list >> setup-log.txt
arduino-cli board list
echo .
```

Zu Beginn dieses Skript-Abschnittes wird zunächst der Nutzer der Software über die Suche nach den angeschlossenen Arduino Platinen informiert **echo Es wird nach angeschlossenen Boards gesucht...**. Folgend wird mit dem Befehl **echo Folgende Boards sind am PC angeschlossen: » setup-log.txt** über den Inhalt der nächsten Zeile in der Log-Datei informiert. Die Information darüber welche Arduinos an dem PC angeschlossen sind, kann mit dem Befehl **arduino-cli board list**, der hier doppelt ausgeführt wird, gewonnen werden. Bei der ersten Ausführung wird die Rückgabe des Arduino-CLI in der Log-Datei gespeichert **» setup-log.txt**, die zweite Ausführung dient zur Anzeige in dem aktuell ausgeführten Skript. Die Information über die angeschlossenen Arduinos benötigt der Nutzer im nächsten Schritt für eine Eingabe.

```
:: Abfragen des Portnamens fuer spaeteren Upload
:: der Sensorentestdatei
set /p port="Bitte den Portnamen des Sense-Lite eingeben und bestaetigen: "
echo Der Port %port% wurde gewahlt. >> setup-log.txt
```

In der Zeile **set /p port="Bitte den Portnamen des Sense-Lite eingeben und bestaetigen: "** erfolgt eine Nutzerabfrage. Der Befehl **set** ermöglicht das Setzen der Variable **port** auf einen bestimmten Wert. Mit dem Zusatz **/p** wird dieser bestimmte Wert auf die folgende Nutzereingabe gesetzt. Die Ausführung des Skriptes wird bis zur erfolgreichen Eingabe des Nutzers gestoppt. Die Nutzereingabe beinhaltet den Port des angeschlossenen Arduino Nano 33 BLE Sense Lite. Für die spätere Nachvollziehbarkeit wird der gewählte Port in der Log-Datei mit der Zeile **echo Der Port %port% wurde gewahlt. » setup-log.txt** dokumentiert.

```
:: Anlegen des Ordners fuer den kompilierten Sensorentest
set ordner=SensorentestCompiledData
mkdir %ordner%
```

Die Variable **ordner** bekommt den Wert **SensorentestCompiledData** zugewiesen. In der nächsten Zeile **mkdir %ordner%** wird der Ordner mit dem Namen **Sensorentest-CompiledData** für das spätere Speichern des kompilierten Sketches angelegt.

```
:: Gegebenenfalls werden die noetigen Bibliotheken fuer
:: den Sensorentest installiert
:: Bib fuer IMU
echo Bib fuer IMU >> setup-log.txt
arduino-cli lib install Arduino_LSM9DS1 >> setup-log.txt
echo . >> setup-log.txt

:: Bib fuer Farbsensor
echo Bib fuer Farbsensor >> setup-log.txt
```

```
arduino-cli lib install Arduino_APDS9960 >> setup-log.txt
echo. >> setup-log.txt
```

```
:: Bib fuer Druck- und Temperatursensor
echo Bib fuer Druck- und Temperatursensor >> setup-log.txt
arduino-cli lib install Arduino_LPS22HB >> setup-log.txt
echo. >> setup-log.txt
```

In der Log-Datei wird jeweils dokumentiert um welche Bibliothek es sich in der folgenden Zeile handelt **SensorentestCompiledData**. Anschließend erfolgt mit dem Befehl **arduino-cli lib install Arduino_LSM9DS1 » setup-log.txt** das Installieren der Bibliothek sowie das Speichern des Rückgabewertes der Installation in der Log-Datei. Eine bereits installierte Bibliothek wird von dem Arduino-CLI erkannt und es folgt eine entsprechende Rückgabe in die Log-Datei. Das Vorgehen ist für alle drei zu installierenden Bibliotheken identisch.

```
:: Kompilieren des Sensorentest-Sketches
echo Kompilieren des Sensorentest-Sketches: >> setup-log.txt
echo. >> setup-log.txt
arduino-cli compile -b arduino:mbed_nano:nano33ble %cd%\SensortestLite
--build-path %cd%\%ordner% >> setup-log.txt
```

Nachdem die notwendigen Bibliotheken installiert sind, kann der zuvor erstellte Sketch zum Testen der Sensoren kompiliert werden. Das Kompilieren des Sketches für den Arduino Nano 33 BLE Sense Lite erfolgt mit dem Befehl **arduino-cli compile -b arduino:mbed_nano:nano33ble %cd%/SensortestLite**. Der hintere Teil des Befehls gibt den Speicherpfad des zu kompilierenden Sketches an. Weiterhin lässt sich der Zielordner für den kompilierten Sketch mit dem Zusatz **-build-path %cd%/%ordner% » setup-log.txt** definieren. Der Rückgabewert des Kompilierens mit dem Arduino-CLI wird in der Log-Datei gespeichert.

```
:: Hochladen des kompilierten Sketches auf den Arduino
echo Hochladen des kompilierten Sketches auf den Arduino >> setup-log.txt
arduino-cli upload -p %port% --input-dir %cd%\%ordner% >> setup-log.txt
```

Der kompilierte Sketch wird in der Zeile **arduino-cli upload -p %port% -input-dir %cd%/%ordner% » setup-log.txt** auf den durch **port** angeschlossenen Arduino Nano 33 BLE Sense Lite hochgeladen. Der Speicherpfad der kompilierten Daten wird mit dem Zusatz **-input-dir %cd%/%ordner%** angegeben.

```
:: Oeffnen des seriellen Monitors
start monitor_log
:: Automatisches Schlieszen des seriellen Monitors nach 4 Sekunden
:: (n=1 Sekunden, mit n=5)
ping 127.0.0.1 -n 5 > nul
taskkill /im serial-monitor.exe /F
```

Mit dem Befehl **start monitor_log** wird ein weiteres Batch-Skript zum Auslesen des seriellen Monitors aufgerufen. Die Zeile **ping 127.0.0.1 -n 5 > nul** stoppt die Ausführung der Software für vier Sekunden. Dabei werden fünf Anfragen an den lokalen Rechner gesendet und jeweils eine Sekunde zwischen zwei Anfragen gewartet. Mit **> nul** wird die Ausgabe der Antworten ins Leere geleitet und nicht angezeigt. Ist die Zeitspanne von vier Sekunden abgelaufen wird der serielle Monitor zum Auslesen der Sensordaten automatisch mit dem Befehl **taskkill /im serial-monitor.exe /F** geschlossen. Über den Zusatz **/im** kann das zu schließende Fenster **serial-monitor.exe** benannt werden. Das erzwungene Schließen des seriellen Monitors erfolgt mit dem Anhang **/F**. Nachdem der serielle Monitor ausgelesen und geschlossen wurde, kann die Software beendet werden. Die Ergebnisse des Sensoren-Tests lassen sich im Anschluss der Log-Datei entnehmen.

Das Öffnen des seriellen Monitors erfolgt in dem Skript `monitor_log.bat`.

```
echo Sensordaten: >> setup-log.txt  
echo. >> setup-log.txt
```

```
echo Der serielle Monitor wird geoeffnet , die Sensordaten werden ausgelesen und i
```

```
arduino-cli monitor -p %port% >> setup-log.txt
```

Der serielle Monitor kann mittels des Arduino-CLI Befehls `arduino-cli monitor -p %port% » setup-log.txt` geöffnet werden. Dabei wird mit `-p %port%` die Verbindung für den angegebenen Port hergestellt. Mit dem Zusatz `» setup-log.txt` werden die empfangenen Daten in die Log-Datei geschrieben.

11. OpenMV IDE

The OpenMV IDE is a premier integrated development environment which is a software tool to use the openMV camera present on the Arduino vision shield. It has a text editor ,frame buffer viewer with a histogram display and a debug terminal. It can be used across platforms, supporting Windows, Mac OS, Linux and Raspian. It was built for machine vision applications and is programmed in python[Tea21c]. Allerdings setzt das Werkzeug die Verwendung von MicroPython voraus.

WS:Prüfen, ob dies stimmt.

11.0.1. OpenMV IDE Installation

To install the OpenMV IDE, first we need to download the IDE from their web-page <https://openmv.io/pages/download>.The set up file name is openmv-ide-windows-2.6.9.exe and the size of it is 1,21,573 KB. we can specify the path according to our needs. Here i have set the path as C:\Users\Shoeb\AppData\Roaming\Microsoft\Windows\StartMenu\Programs\OpenMVIDE

After the download is done,open the setup file. Then click on Next.

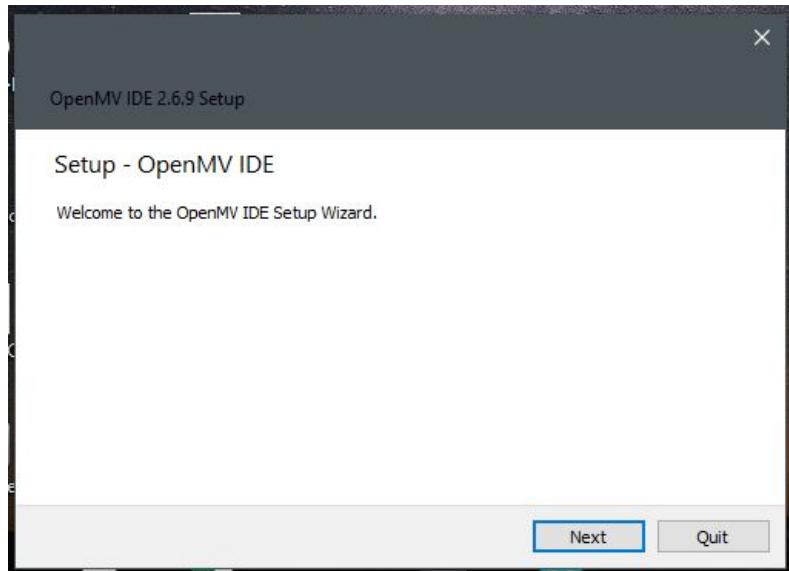


Figure 11.1.: OpenMV IDE Installation

Then select the installation folder and click on Next as shown in figure 11.2.

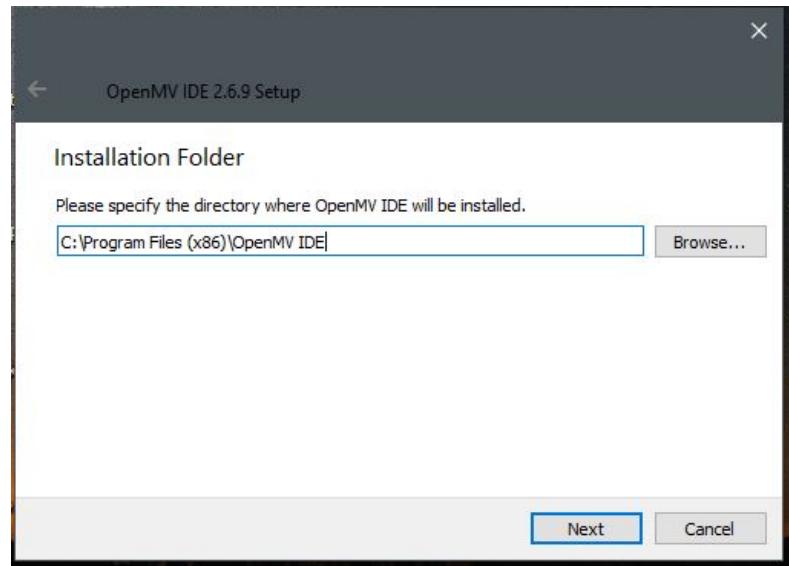


Figure 11.2.: OpenMV IDE Installation folder

Then a dialogue box pops up showing ready to install, click on Install as shown in figure 11.3.

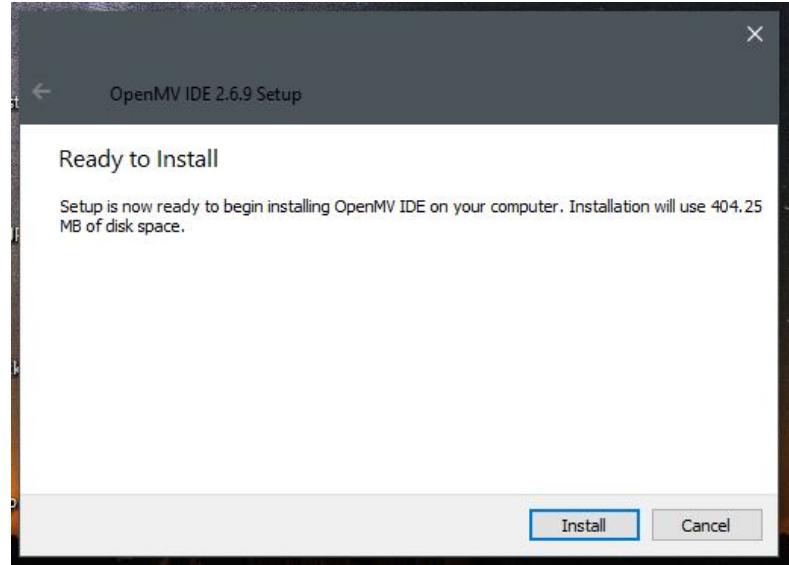


Figure 11.3.: OpenMV IDE Setup Install

After the installation is done, OpenMV IDE opens up with a default program as shown in figure11.4.

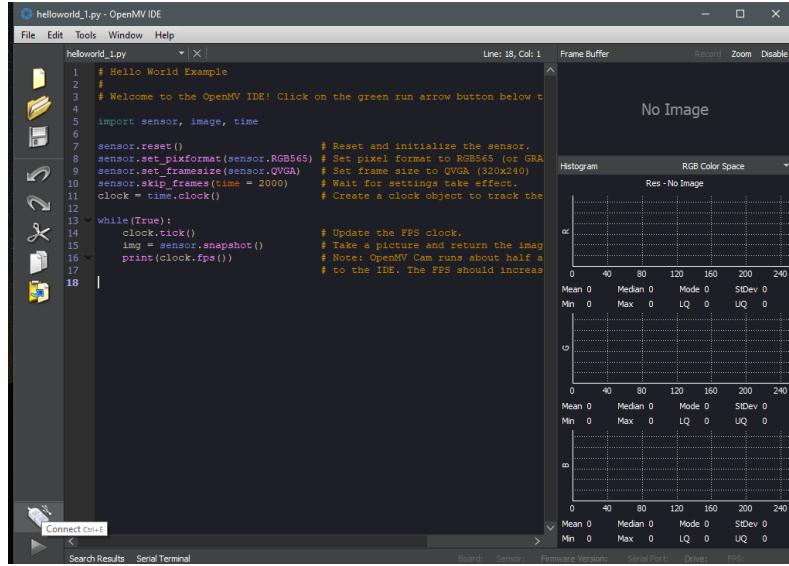


Figure 11.4.: OpenMV IDE

Now open the OpenMV IDE and click on connect. Then select Install the latest release firmware(v4.0.1) and click OK as shown in figure11.5.

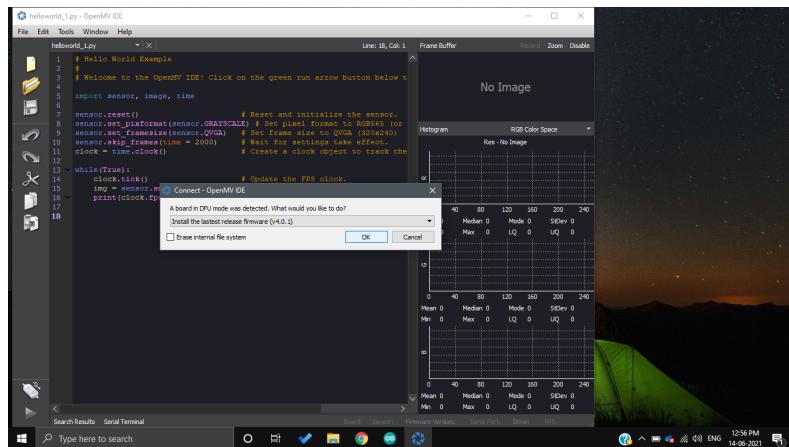


Figure 11.5.: Installing latest firmware

Then a dialog box opens up saying that the update is complete as shown in figure11.6.

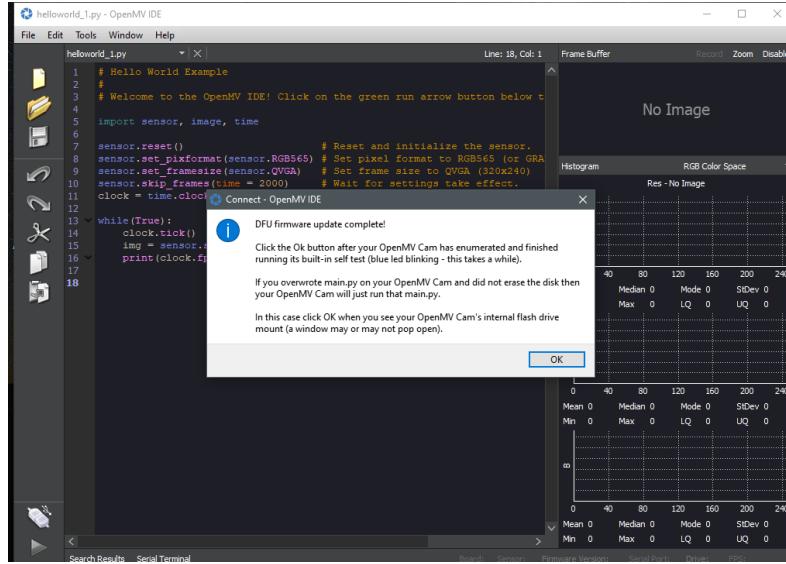


Figure 11.6.: firmware update complete

Now we hit the play button to run the program. If we are not able to run then we need to reset the Arduino Portenta H7 board. After the reset is done, click on the play button at the bottom of the screen. By this, we have successfully connected our Vision shield with Arduino Portenta H7 and now we can start doing machine vision applications using the camera provided by the Vision shield.

11.0.2. OpenMV IDE Overview

In the figure 11.6 we can see that the OpenMV IDE has an integrated frame buffer on the top right side. This lets us see the camera stream from the vision shield camera. There are 3 options on top which are record, zoom and disable.

- **Record** : This lets us record the video stream in the frame buffer at 30 FPS.
- **Zoom** : This option enables to zoom in or out the frame buffer viewer.
- **Disable** : This option lets us control the camera stream. If we disable it, then we won't be able to see the camera stream in the frame buffer. We can also save image from the frame buffer by simply right clicking it and select save as image option.

Histogram Display : The OpenMV IDE has a histogram display which is used for getting feedback about the lighting quality in the room which is basically giving the idea about quality of the image the camera is looking at. We have four different options here to select such as RGB, Grayscale, YUV and LAB. These are useful in programming to determine the correct grayscale and LAB color channel settings which we need to use in the scripts.

Serial Terminal : This is present at the bottom of the OpenMV IDE. All the debug text which we write in the `print()` command will be displayed here.

Status Bar : In the status bar OpenMV IDE will display the OpenMV Cam's Firmware Version, Serial Port, Drive, and FPS. We can also update the firmware version from here if it is not updated. The status bar displays the current FPS, the board which is H7 in our case and the serial port label as well.

Tools : In the tools, there are many options available.

- Save open script to your OpenMV cam : This lets us save the script in the OpenMV cam which is the Arduino vision shield and it will also automatically deletes comments and white spaces to save memory and it will store the script as main.py.
- Configure OpenMV cam settings file : This allow us to modify an .ini file which is stored in the OpenMV cam using OpenMV IDE which the OpenMV cam will read on bootup for particular hardware configurations.
- Reset OpenMV cam : This command resets and disconnects from the OpenMV cam.

12. First Steps with the Vision Shield

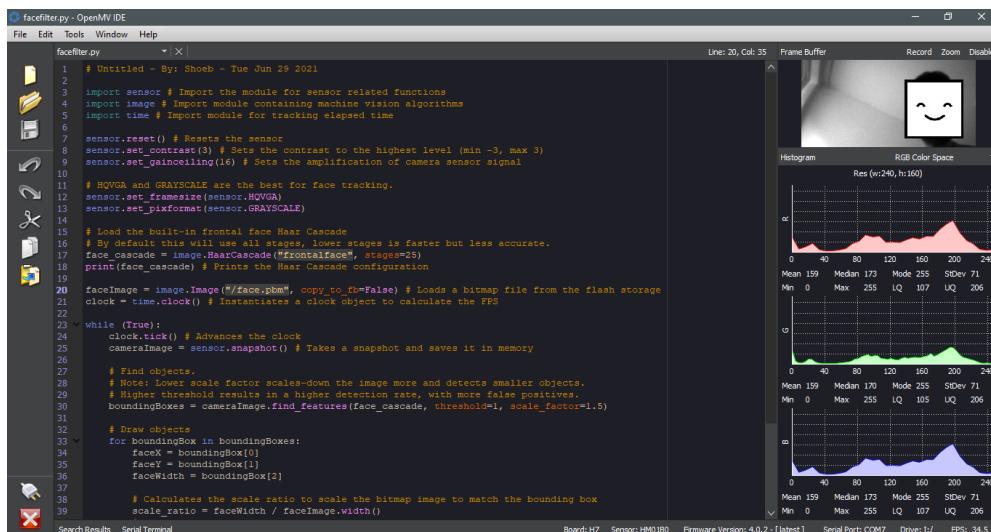
12.1. Introduction

In this chapter we will be going through the installation of OpenMV IDE and the steps required for connecting the Arduino Portenta H7 with the Vision Shield using the OpenMV IDE. Further, we will be looking at the overview of the OpenMV IDE and what are the tools provided in the IDE. Finally, we will be implementing some of the example programs provided in the OpenMV IDE.

12.2. OpenMV Examples

12.2.1. Creating a basic face filter with OpenMV

In this example we use a smiley image in portable bitmap image(.pbm) format because OpenMV supports bmp, pgm or ppm image formats and not the other image formats such as .png [SR21]. This image will be overlayed on the face when it detects a face in the camera stream. Download the image and copy it into the flash drive. First we load the image into a variable named as `faceImage` using the `Image()` function. We use `copy_to_fb` to false because we don't want it to be copied automatically in to the frame buffer. Next we calculate the scale ratio to scale the bitmap image to match the detected face in the camera stream using the function `scale_ratio=faceWidth/faceImage.width()`. Then we draw the bitmap image on the detected face using the function `draw_image()`. Here the face detection is done using the Haar cascade algorithm. In this algorithm, it has multiple stages in which the output of one stage gives additional information to the input of next stage in the cascade. By doing this, it calculates the edges, lines, contrast checks in different stages and ultimately detects the face and draws a bounding box around it.



The screenshot shows the OpenMV IDE interface with a Python script named `facefilter.py` open in the editor. The code implements a face tracking and filtering application. It starts by initializing the sensor, setting contrast, and loading a Haar cascade for frontal face detection. It then loads a smiley face image from the flash storage and sets its width to match the detected face's width. A bounding box is drawn around the detected face, and the smiley face is overlaid on it. The IDE also displays histograms for the RGB color space and three individual channels (R, G, B) to analyze the image's color distribution.

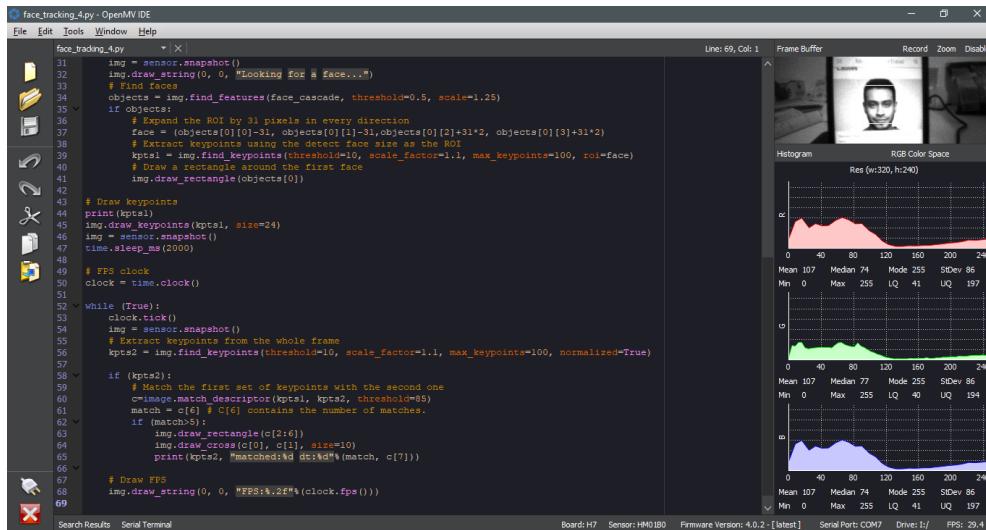
```
# Untitled - By: Shoeb - Tue Jun 29 2021
File Edit Tools Window Help
facefilter.py
1 # Import the module for sensor related functions
2 import sensor
3 # Import module containing machine vision algorithms
4 import image
5 # Import module for tracking elapsed time
6
7 sensor.reset() # Resets the sensor
8 sensor.set_contrast(3) # Sets the contrast to the highest level (min =3, max 3)
9 sensor.set_gaodecanceling(16) # Sets the amplification of cameras sensor signal
10
11 # HQVGA and GRAYSCALE are the best for face tracking.
12 sensor.set_framesize(sensor.HQVGA)
13 sensor.set_pixformat(sensor.GRAYSCALE)
14
15 # Load the built-in frontal face Haar Cascade
16 # By default this will use all stages, lower stages is faster but less accurate.
17 face_cascade = image.HaarCascade("frontalface", stages=25)
18 print(face_cascade) # Prints the Haar Cascade configuration
19
20 faceImage = image.Image("/face.pbm", copy_to_fb=False) # Loads a bitmap file from the flash storage
21 clock = time.Clock() # Instantiates a clock object to calculate the FPS
22
23 while (True):
24     clock.tick() # Advances the clock
25     cameraImage = sensor.snapshot() # Takes a snapshot and saves it in memory
26
27     # Find objects.
28     # Notes: Lower scale factor scales-down the image more and detects smaller objects.
29     # Higher threshold results in a higher detection rate, with more false positives.
30     boundingBoxes = cameraImage.find_features(face_cascade, threshold=1, scale_factor=1.5)
31
32     # Draw objects
33     for boundingBox in boundingBoxes:
34         faceX = boundingBox[0]
35         faceY = boundingBox[1]
36         faceWidth = boundingBox[2]
37
38         # Calculates the scale ratio to scale the bitmap image to match the bounding box
39         scale_ratio = faceWidth / faceImage.width()
```

Figure 12.1.: Face Filter output

As we can see from the above figure 21.4 that as soon as a face is detected in the camera stream, the bitmap image which was loaded earlier is overlapped on to the detected face.

12.2.2. Face tracking

In this example, we are going to detect face and draw a bounding box around it. It uses the Haar cascade algorithm to detect faces. To do this, go to files in the OpenMV IDE and open examples and select face_tracking. In the program, firstly, we reset the sensor using `sensor.reset()` function and we set the frame size to QVGA and set the pixel format to grayscale. Next we load the Haar cascade using the function `image.HaarCascade()`. Next we use the keypoints feature of OpenMV camera to track a face after it is detected by Haar cascade algorithm. For this, first we set the keypoints to none and when the face is detected we will use the function `image.find_keypoints()` function. Then by using `draw_rectangle()` function we draw a rectangle around the detected face and draw keypoints using `draw_keypoints()` function. After this, we extract keypoints in the whole frame and then match this keypoints with first keypoints which was used only for face using `image.match_descriptor()` function. If the keypoints match is greater than 5 then we draw a rectangle around the matched feature.



```

face_tracking_4.py - OpenMV IDE
File Edit Tools Window Help
face_tracking_4.py - x | Line: 69, Col: 1
31     img = sensor.snapshot()
32     img.draw_string(0, 0, "Looking for a face...")
33     # Find faces
34     objects = img.find_features(face_cascade, threshold=0.5, scale=1.25)
35     if objects:
36         # Expand the ROI by 31 pixels in every direction
37         face = (objects[0][0]-31, objects[0][1]-31, objects[0][2]+31*2, objects[0][3]+31*2)
38         # Extract keypoints using the detect face size as the ROI
39         kpts1 = img.find_keypoints(threshold=10, scale_factor=1.1, max_keypoints=100, roi=face)
40         # Draw a rectangle around the first face
41         img.draw_rectangle(objects[0])
42
43         # Draw keypoints
44         print(kpts1)
45         img.draw_keypoints(kpts1, size=24)
46         img = sensor.snapshot()
47         time.sleep_ms(2000)
48
49         # FFS clock
50         clock = time.clock()
51
52     while (True):
53         clock.tick()
54         img = sensor.snapshot()
55         # Extract keypoints from the whole frame
56         kpts2 = img.find_keypoints(threshold=10, scale_factor=1.1, max_keypoints=100, normalized=True)
57
58         if (kpts2):
59             # Match the first set of keypoints with the second one
60             c = image.match_descriptor(kpts1, kpts2, threshold=85)
61             match = c[6] # C[6] contains the number of matches.
62             if (match > 5):
63                 img.draw_rectangle(c[2:6])
64                 img.draw_cross(c[0], c[1], size=10)
65                 print(kpts2, "Matched:%d dt:%d% match, c[%d]" % (match, c[7]))
66
67         # Draw FPS
68         img.draw_string(0, 0, "FPS:%.2f" % (clock.fps()))
69
Search Results Serial Terminal
Board: H7 Sensor: HM01B0 Firmware Version: 4.0.2 [latest] Serial Port: COM7 Drive: /f/ FPS: 29.4

```

Figure 12.2.: Face tracking output

After we run this sketch, we need to point the vision shield camera towards a face and focus it properly. As soon as the face is detected in the camera stream , a rectangular box is formed around it indicating the detected face in the image. In this example, we can see that it clearly identifies the face and the bounding box is also generated around it.

12.2.3. Finding circles around the objects

In this example, we find the circles in an image using Hough transform algorithm. This algorithm is used in image processing for extracting the features such as circles or ellipses in an image. For programming this, we first reset the sensor as described in earlier examples. Then we take snapshot from the camera stream using `sensor.snapshot()` function. Then we find the circle in the image using `find_circles()` function. Lastly we draw circles using `draw_circles()` function.

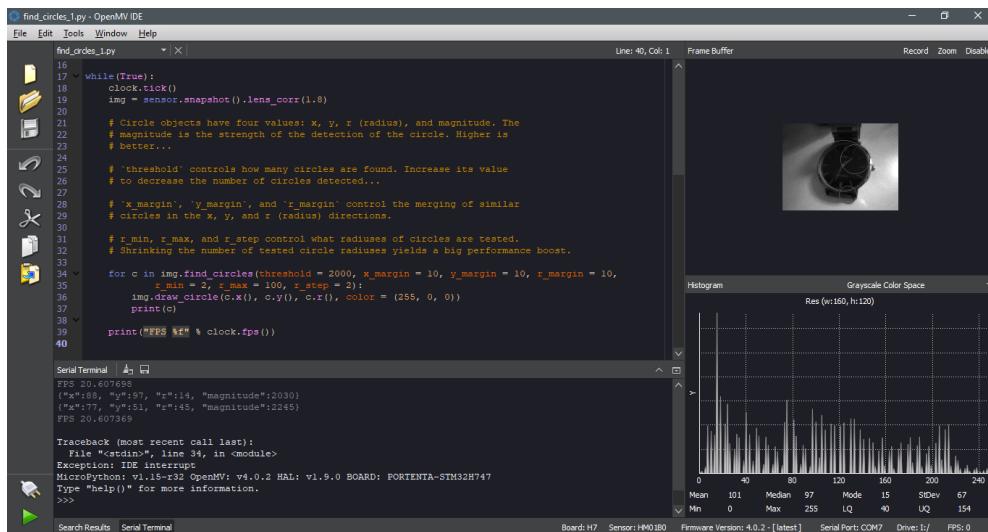


Figure 12.3.: Find circles output

As can be seen from the figure 21.6, the output shows that the circles are drawn on the border of the watch. we need to bring the objects closer to the camera so that the focus is set properly in order to detect the circles around the objects.

12.2.4. QR Code information extraction

In this example we extract the information from QR code by using the vision shield. This is an interesting feature of the openMV camera present in the vision shield. In order to program this, we first follow the same initialization steps as mentioned in the earlier examples. We then take a snapshot from the camera stream using the `sensor.snapshot()` function and then we do the lens correction to 1.8 in order to get a better quality image using `lens_corr()` function. To extract the QR code we use `find_qrcodes()` function and then we draw a rectangle around the QR code using `draw_rectangle()` function.

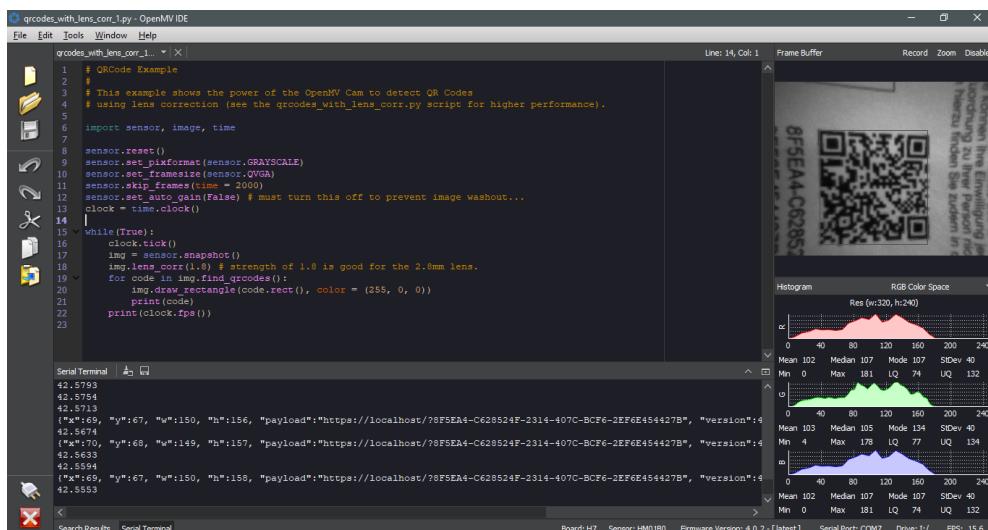


Figure 12.4.: QR Code output

In the output figure 21.7 we can see that a rectangular box is created around the QR

Code and the information is extracted from it and it can be seen in the output serial terminal.

12.2.5. Blob-Erkennung mit OpenMV

Quelle: <https://docs.arduino.cc/tutorials/nicla-vision/blob-detection>

In diesem Beispiel wird der Arduino Nicla Vision verwendet, um das Vorhandensein und die Position von Objekten in einem Kamerabild zu erkennen. Dazu wird eine Technik verwendet, die als Blob-Erkennung bezeichnet wird. Für diese Aufgabe wird ein MicroPython-Skript erstellt und mit Hilfe der OpenMV IDE auf der Nicla Vision ausgeführt. OpenMV IDE ermöglicht einerseits die Kommunikation mit dem Arduino-Board und andererseits wird auf einfache Weise eine Vorschau des Videostreams angezeigt, so dass Farbbereiche visuell inspiziert werden können, um Schwellenwerte zu definieren.

12.2.6. Blob-Erkennung

Dieser Blob-Algorithmus ermöglicht die Erkennung von Bereichen in einem digitalen Bild, die sich in Eigenschaften wie Helligkeit oder Farbe von den umliegenden Bereichen unterscheiden. Diese Bereiche werden als Kleckse bezeichnet. Stellen Sie sich einen Blob als einen Klumpen ähnlicher Pixel vor.

Anwendungsbeispiele:

- Erkennen bestimmter Fahrzeuge, die vor der Kamera vorbeifahren
- Erkennen von fehlenden Teilen in einer Montagelinie
- Insektenbefall auf Gemüse erkennen

12.2.7. Erstellen des MicroPython-Skripts

Um Kleckse zu finden, muss dem Algorithmus ein Bild von der Kamera vorgelegt werden. Dieser analysiert es dann und gibt die Koordinaten der gefundenen Kleckse aus. Diese Koordinaten werden direkt auf dem Bild angezeigt und mit Hilfe der roten und grünen LED wird angezeigt, ob ein Fleck gefunden wurde.

Um dies zu ermöglichen, wird zunächst eine neue Datei angelegt, indem Sie auf die Schaltfläche „New File“ in der Symbolleiste auf der linken Seite geklickt wird. In der leeren Datei werden dann zuerst die erforderlichen Module importiert:

- `pyb`
- `sensor`
- `image`
- `time`

Dies ist in Listing 12.1 dargestellt.

Anschließend wird der Sensor konfiguriert. Die Konfiguration ist nun so, dass ein Schnappschuss mit der Kamera gemacht wird. Das zugehörige Listing 12.2 ist ebenfalls wiedergegeben. Die wichtigsten Funktionen in diesem Ausschnitt sind `set_pixformat` und `set_framesize`. Die Kamera, die mit der Nicla Vision geliefert wird, unterstützt RGB 565 Bilder. Daher müssen wir sie über den Parameter `sensor.RGB565` einstellen. Die Auflösung der Kamera muss auf ein Format eingestellt werden, das sowohl vom Sensor als auch vom Algorithmus unterstützt wird. `QVGA` ist ein guter Kompromiss zwischen Leistung und Auflösung und wird daher verwendet.

```
import pyb # Import module for board related functions
import sensor # Import the module for sensor related functions
import image # Import module containing machine vision algorithms
import time # Import module for tracking elapsed time
```

Listing 12.1.: Skript zur Erstellung der Datei `train.txt` mit den Pfaden zu den Trainingsbildern

```
sensor.reset() # Resets the sensor
sensor.set_pixformat(sensor.RGB565) # Sets the sensor to RGB
sensor.set_framesize(sensor.QVGA) # Sets the resolution to 320x240 px
sensor.set_vflip(True) # Flips the image vertically
sensor.set_hmirror(True) # Mirrors the image horizontally
sensor.skip_frames(time = 2000) # Skip some frames to let the image stabilize
```

Listing 12.2.: Einen Schnappschuss mit der Kamera machen

Je nachdem, wie die Kamera positioniert wird, sollten die Funktionen `set_vflip` und `set_hmirror` eingesetzt werden. Um das Board mit dem USB-Kabel nach unten zu halten, muss `set_vflip(True)` aufgerufen werden. Wenn das Bild so angezeigt werden soll, wie es mit Ihren Augen zu sehen ist, muss `sensor.set_hmirror(True)` aufgerufen werden. Andernfalls würden Elemente im Bild, wie z. B. Text, gespiegelt werden.

12.2.8. Festlegen der Farbschwellenwerte

Um den Blob-Erkennungsalgorithmus mit einem Bild zu füttern, müssen Sie einen Schnappschuss von der Kamera machen oder das Bild aus dem Speicher laden (z. B. von der SD-Karte oder dem internen Flash). In diesem Fall machen Sie einen Schnappschuss mit der Funktion `snapshot()`. Das resultierende Bild muss dann mit Hilfe der Funktion `find_blobs` in den Algorithmus eingespeist werden. Sie werden feststellen, dass eine Liste von Tupeln an den Algorithmus übergeben wird. In dieser Liste können Sie die LAB-Farbwerthe angeben, die in dem Objekt, das Sie verfolgen möchten, hauptsächlich enthalten sind. Wenn Sie z. B. rein rote Objekte auf schwarzem Hintergrund erkennen wollen, wäre der resultierende Farbbereich sehr schmal. Der entsprechende LAB-Wert für reines Rot ist ungefähr (53,80,67). Ein etwas helleres Rot könnte (55,73,50) sein. Der LAB-Bereich wäre also L: 53-55 A: 73-80 B: 50-67. OpenMV bietet ein praktisches Werkzeug, um die gewünschten Farbbereiche zu ermitteln: Threshold Editor. Sie finden ihn in der OpenMV IDE im Menü unter `Tools > Machine Vision > Threshold Editor`. Platzieren Sie das gewünschte Objekt vor der Kamera und öffnen Sie das Werkzeug. Wenn Sie nach dem Ort des Quellbildes gefragt werden, wählen Sie „Frame Buffer“. In dem sich öffnenden Fenster sehen Sie einen Schnappschuss der Kamera und einige Schieberegler zur Anpassung der LAB-Farbbereiche. Wenn Sie die Schieberegler verschieben, sehen Sie im Schwarz-Weiß-Bild auf der rechten Seite, welche der Pixel dem eingestellten Farbbereich entsprechen würden. Weiße Pixel kennzeichnen die passenden Pixel. Wie Sie im folgenden Beispiel sehen können, sind die Pixel eines schönen roten Apfels auf braunem Hintergrund sehr schön gebündelt. Sie ergeben meist einen großen Klecks, siehe Abbildung 12.5.

Um eine ungefähre Vorstellung vom LAB-Farbbereich des Zielobjekts zu erhalten, können Sie die Histogrammansicht in OpenMV verwenden. Stellen Sie sicher, dass Sie das Histogramm auf den LAB-Farbmodus eingestellt haben. Ziehen Sie mit dem

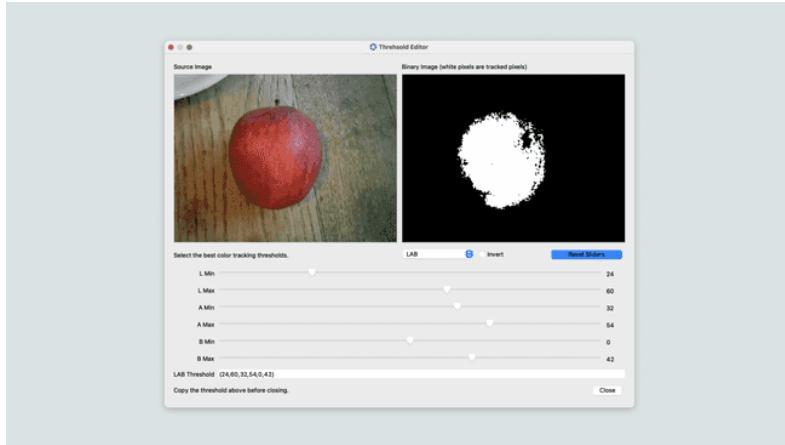


Figure 12.5.: LAB-Schwellenwerte für einen Apfel im Schwellenwert-Editor

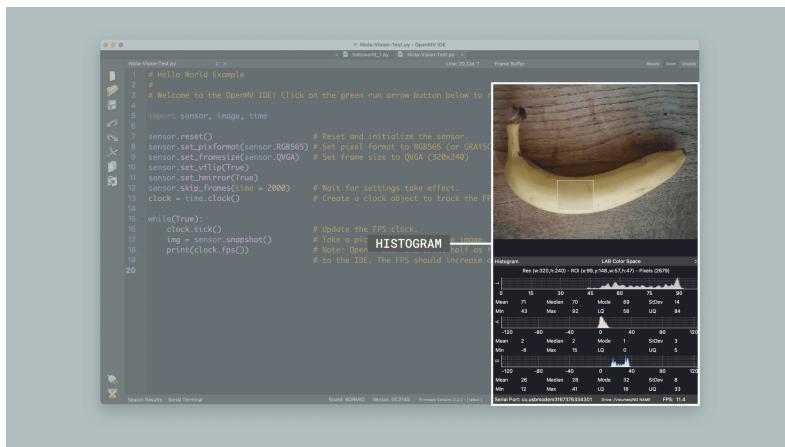


Figure 12.6.: LAB-Farbhistogramm des Bildpuffers

Mauszeiger ein Rechteck direkt über dem Zielobjekt in der Bildpufferansicht auf. Im Histogramm können Sie sehen, welche Farbwerte am häufigsten vorkommen. Sie können die Zielfarbgebiete auf die Minimal- und Maximalwerte der entsprechenden Farbkomponente einstellen. 12.6

Im Gegensatz zum obigen Beispiel mit dem Apfel ist die Clusterbildung der Bananenpixel etwas weniger kohärent. Das liegt daran, dass die Banane auf einem Hintergrund liegt, der eine leicht ähnliche Farbe hat. Das bedeutet, dass der Algorithmus empfindlich auf die Hintergrundpixel reagiert. Um Blobs, die nicht zum Zielobjekt gehören, auszuschließen, ist eine zusätzliche Filterung erforderlich. Sie können z. B. eine Mindestgröße des Begrenzungsrahmens (Bounding Box) oder eine Blob-Pixeldichte festlegen, die Dehnung des Objekts oder seine Rundheit definieren oder auch nur nach Objekten in einem bestimmten Teil des Bildes suchen. 12.7

12.2.9. Erkennung von Blobs

Da Sie nun den Bereich der Farbwerte kennen, der für die Suche nach den Blobs verwendet werden soll, können Sie diese 6 Tupel als Liste an die Funktion `find_blobs` übergeben 12.3

Sobald die Kleckse erkannt sind, möchten Sie vielleicht sehen, wo im Bild sie gefunden wurden. Dies kann durch direktes Zeichnen auf das Kamerabild geschehen, siehe 12.4.

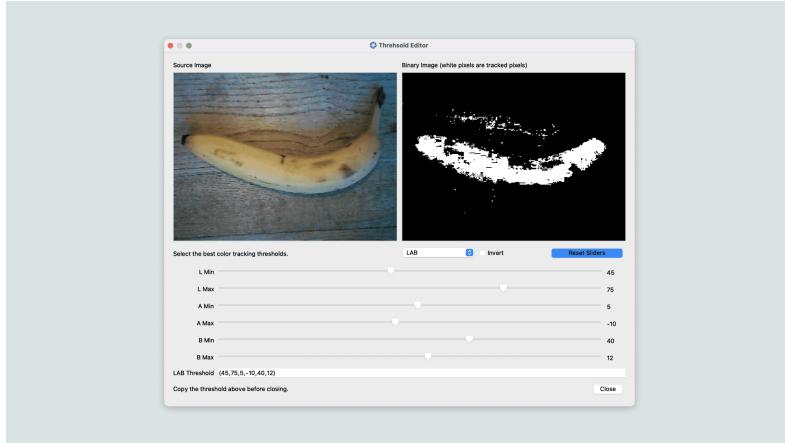


Figure 12.7.: LAB-Schwellenwerte für eine Banane im Schwellenwert-Editor

```
# Define the min/max LAB values we're looking for
thresholdsApple = (24, 60, 32, 54, 0, 42)
thresholdsBanana = (45, 75, 5, -10, 40, 12)
```

```
img = sensor.snapshot() # Takes a snapshot and saves it in memory

# Find blobs with a minimal area of 50x50 = 2500 px
# Overlapping blobs will be merged
blobs = img.find_blobs([thresholdsApple, thresholdsBanana], area_threshold=2500)
```

Listing 12.3.: Detektierung von Blobs

```
# Draw blobs
for blob in blobs:
    # Draw a rectangle where the blob was found
    img.draw_rectangle(blob.rect(), color=(0,255,0))
    # Draw a cross in the middle of the blob
    img.draw_cross(blob.cx(), blob.cy(), color=(0,255,0))
```

Listing 12.4.: Anzeigen von Blobs

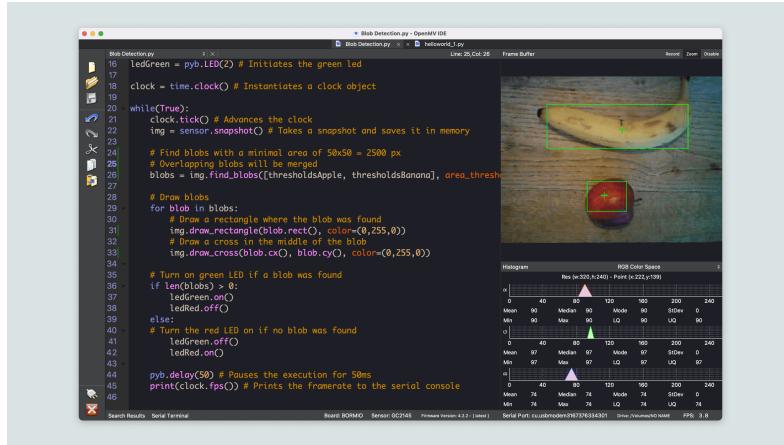


Figure 12.8.: Visualisierung der Blobs in der Bildpuffervorschau

```
ledRed = pyb.LED(1) # Initiates the red led
ledGreen = pyb.LED(2) # Initiates the green led
```

Listing 12.5.: Initialisierung der LEDs

Wenn Sie wissen müssen, welcher Blob welcher Farbschwelle entspricht, können Sie die Funktion `blob.code()` verwenden (weitere Informationen finden Sie [hier](#)).

Das Ergebnis wird im Frame Buffer-Vorschaufenster auf der rechten Seite der Open-MV-IDE, siehe 12.8, sichtbar sein.

12.2.10. LEDs umschalten

What if you want some visual feedback from the blob detection without any computer connected to your board? You could use for example the built-in LEDs to indicate whether or not a blob was found in the camera image. Let's initialise the red and the green LEDs with the following code 12.5

Fügen Sie dann die Logik hinzu, die die entsprechende LED einschaltet, wenn ein Blob vorhanden ist. Dieser Teil des Codes wird nach der Logik „Draw Blobs“ hinzugefügt 12.6.

In diesem Beispiel leuchtet die grüne LED auf, wenn mindestens ein Blob im Bild gefunden wurde. Die rote LED leuchtet, wenn kein Blob gefunden werden konnte.

```
# Turn on green LED if a blob was found
if len(blobs) > 0:
    ledGreen.on()
    ledRed.off()
else:
# Turn the red LED on if no blob was found
    ledGreen.off()
    ledRed.on()
```

Listing 12.6.: Initialisierung der LEDs

12.2.11. Hochladen des Skripts

Programmieren wir das Board mit dem vollständigen Skript und testen wir, ob der Algorithmus funktioniert. Kopieren Sie das folgende Skript 12.7 und fügen Sie es in die neue Skriptdatei ein, die Sie erstellt haben.

Klicken Sie auf die Schaltfläche „Play“ unten in der linken Symbolleiste. Legen Sie einige Objekte auf Ihren Schreibtisch und prüfen Sie, ob Portenta bzw. Nicla Vision sie erkennen kann.

Achtung: Das MicroPython-Skript wird nicht kompiliert und in eine eigentliche Firmware eingebunden. Stattdessen wird es in den internen Flash des Boards kopiert, wo es interpretiert und direkt ausgeführt wird.

12.3. Edge Impulse

Edge Impulse is a platform which makes the process of machine learning easier by choosing reasonable defaults for the parameters to be set when creating a model. It was founded in 2019 by Zach Shelby and Jan Jongboom. It provides a user friendly interface and we can train, test and deploy the model. Edge Impulse uses the TensorFlow ecosystem for training , optimizing and deploying deep learning models in to the embedded devices for example Arduino Portenta H7 [Tea21a]. The TensorFlow ecosystem provides a set of standards and integration points that can be used to make our own improvements in the model.

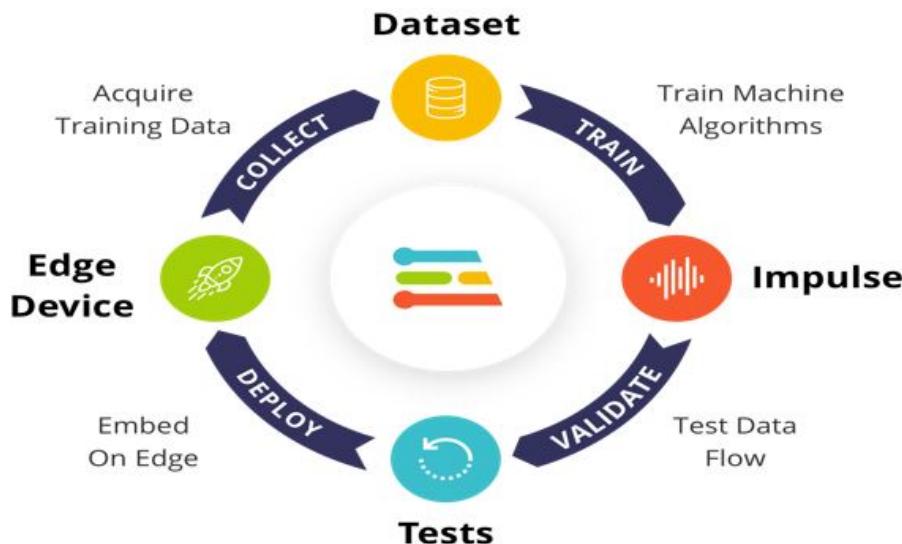


Figure 12.9.: Edge Impulse process [Tea21b]

The process involves collecting the data from the devices and creating a training data set in the Edge Impulse cloud. Then it will undergo the training process for the model and after the training process is completed, the model is tested using the testing data provided or the model can also be validated using live classification by loading the testing sample directly from a device. Then finally after the model is tested, it is deployed back in the device. In the background, Edge Impulse generates a python implementation of the model using TensorFlow ‘s Keras APIs. By switching into the expert mode we can edit the code and make changes in the parameters. The TensorFlow libraries and APIs are used by Edge Impulse and hence it makes it easy to extend the built-in training code with our own logic. The training process is done

```

import pyb # Import module for board related functions
import sensor # Import the module for sensor related functions
import image # Import module containing machine vision algorithms
import time # Import module for tracking elapsed time

sensor.reset() # Resets the sensor
sensor.set_pixformat(sensor.RGB565) # Sets the sensor to RGB
sensor.set_framesize(sensor.QVGA) # Sets the resolution to 320x240 px
sensor.set_vflip(True) # Flips the image vertically
sensor.set_hmirror(True) # Mirrors the image horizontally
sensor.skip_frames(time = 2000) # Skip some frames to let the image stabilize

# Define the min/max LAB values we're looking for
thresholdsApple = (24, 60, 32, 54, 0, 42)
thresholdsBanana = (45, 75, 5, -10, 40, 12)

ledRed = pyb.LED(1) # Initiates the red led
ledGreen = pyb.LED(2) # Initiates the green led

clock = time.clock() # Instantiates a clock object

while(True):
    clock.tick() # Advances the clock
    img = sensor.snapshot() # Takes a snapshot and saves it in memory

    # Find blobs with a minimal area of 50x50 = 2500 px
    # Overlapping blobs will be merged
    blobs = img.find_blobs([thresholdsApple, thresholdsBanana], area_threshold=2500)

    # Draw blobs
    for blob in blobs:
        # Draw a rectangle where the blob was found
        img.draw_rectangle(blob.rect(), color=(0,255,0))
        # Draw a cross in the middle of the blob
        img.draw_cross(blob.cx(), blob.cy(), color=(0,255,0))

    # Turn on green LED if a blob was found
    if len(blobs) > 0:
        ledGreen.on()
        ledRed.off()
    else:
        # Turn the red LED on if no blob was found
        ledGreen.off()
        ledRed.on()

    pyb.delay(50) # Pauses the execution for 50ms
    print(clock.fps()) # Prints the framerate to the serial console

```

Listing 12.7.: Initialisierung der LEDs

in the cloud and the trained model are automatically optimized for deployment to the microcontrollers.

Data Acquisition format : In order to send data from any sensor to the Edge Impulse platform we need to send the data in data acquisition format. If the data we have is in CSV or WAV then we can directly upload in the Edge Impulse platform. For images, the input image formats accepted by the Edge impulse platform are JPG and PNG. Images can be in RGB or Grayscale. There is also an option available if we need to convert the input image into grayscale.

Image Processing Block : In this processing block, preprocessing and normalization of image data is done. In addition, color depth can also be reduced. After the input image is loaded, this processing block is responsible for extracting features from the image.

Learning Block : In this learning block we have options to choose between classification and object detection. The object detection model is a pre-trained model and we can use this pre-trained model and do fine tuning to obtain desired results. This pre-trained model is based on MobileNetV2-SSD FPN-lite architecture which is designed for mobile and embedded vision application. This model is trained on images of 320X320 resolution.

Advantages

- Connect multiple devices to the platform. Even a mobile phone can also be used as a device.
- Multiple options to upload our input data from different devices.
- Drag and drop UI makes it user friendly for training, testing and deploying the model with little to no coding.

Disadvantages

- Difficult for fine grain customization of the model.
- Data security issues.

13. Google Colaboratory

13.1. Introduction

Google Colaboratory, or “Colab” for short, is a product from Google Research and is a free Jupyter notebook environment running completely in the cloud. Colab allows anybody to write and execute arbitrary python code through the browser, but it is typically well suited to machine learning and data analysis. The good thing is it requires no set up and provides free access to computing resources like GPU and TPU [Goo21]. The Chrome browser is a specific requirement primarily for Google Colab which is going to rely on working well with Chrome. Google Colab is an online integrated developer environment to design, train, and test our machine learning models. There is a nice introduction on youtube.com made by Jake Van der Plas. [Pla19] The next step is the practical use of Google Colab. Google offers a notebook [Introductory Colab](#). In it they provide a handful of quick tips and hands on exercises to get you started using the Colab environment.

13.2. Tips for using Colab

Here, there are a couple of tips and tricks for using Colab:

1. Each Colab instance will stay alive as long as the user interacts with it once every 90 minutes.
2. Each instance will only last for a maximum of 12 hours so it is important to download as soon as possible all of the files which are needed to avoid losing them. That’s not so easy if long notebooks are used.
3. The user can access the files in the screen by clicking the folder icon as shown in figure 13.1.
4. By default all user’s files are saved under [/content](#) and by default the user starts only seeing that folder.

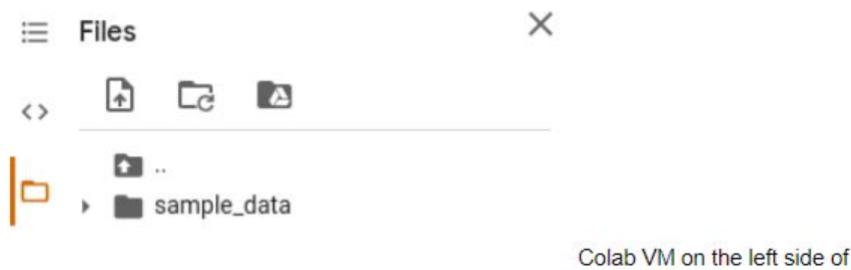


Figure 13.1.: This image is a screenshot of the Colab UI showing one portion of the interface. In this portion, there is a folder icon highlighted in yellow, which can be used to get to the files.

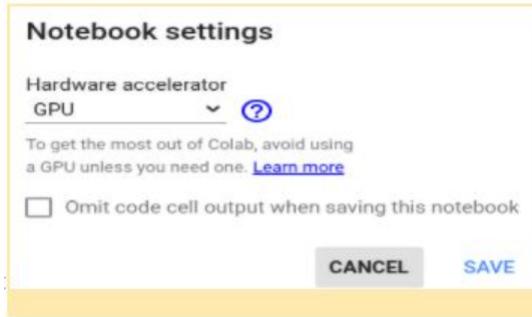


Figure 13.2.: Selecting GPU under Hardware accelerator

5. While the code cells in Colab default to python, any bash command can be executed by prepending the command with `!`. For example, the user can unzip a file with `!unzip file.zip -d destination_folder`.
6. To download a file from Colab the user has simply to click the three dots next to the file's name and then select `download`.
7. To upload a file to Colab the user has simply to click the icon with the piece of paper with a folded corner and an arrow on it below the word `Files`.
8. To make permanent changes to a Colab one needs to select `file→Save` a copy in `Drive` which will make a copy of the Colab in the user's drive with your changes saved.
9. Activating a GPU instance drastically speeds up Neural Network training. This can be done through `runtime→Change runtime type` and then selecting `GPU` under `Hardware accelerator`, see figure 13.2.

13.3. Advantages of Colab

- **Pre installed libraries:** It provides pre-installed machine learning libraries including PyTorch, Keras, TensorFlow
- **Saved on the Cloud :** If we use a Jupyter notebook, then everything is saved on a local device. However, when we use Google Colab, we can save it on the cloud and can access from any device by just signing in Google Drive account.
- **Collaboration :** It helps in providing access to multiple developers working on the project to edit the code or code together and also share the completed code with the developers
- **Free GPU and TPU use :** This is probably the best feature as Google Research lets us use their dedicated GPUs and TPUs for our personal machine learning projects. Therefore, our computers don't have to undergo the complex computations while training the model because it is done on the cloud

13.4. Selection of the Model: YOLO V4

You Only Look Once (YOLO) is a state of art Object Detector which can perform object detection in real-time with a good accuracy. They belong to the family of single stage detection also referred to as one shot detection.

YOLO V4 also based on the Darknet and has obtained an Average Precision (AP) value of 43.5 percent on the dataset COCO along with a real-time speed of 65 frames per second (fps) on the Tesla V100, beating the fastest and most accurate detectors in terms of both speed and accuracy.

When compared with YOLO V3, the AP andfps have increased by 10 percent and 12 percent, respectively.

Note: Further information about the model YOLO V4 such as the detailed architecture, the different layers and the hyper parameters can be found in the file [YOLOV4.tex](#)in the folder “Software”.

13.5. Training the model

The below are the steps to train the model with Google Colab.

- Create yolov4 and training folders in the google drive
- Mount drive, link the folder, and navigate to the yolov4 folder
- Clone the Darknet git repository [BWL20]
- Create & upload the files we need for training (i.e. [obj.zip](#) , [yolov4-custom.cfg](#) , [obj.data](#) , [obj.names](#) and [process.py](#)) to the drive
- Make changes in the Makefile to enable OPENCV and GPU
- Run make command to build darknet Copy the files [obj.zip](#), [yolov4-custom.cfg](#) [obj.data](#) , [obj.names](#) from the yolov4 folder to the darknet directory
- Run the process.py python script to create the [train.txt](#) & [test.txt](#) files
- Download the pre-trained YOLOv4 weights
- Train the detector
- Check performance

13.5.1. Architecture of YOLO V4

This section gives a brief overview of the YOLO V4 model: The YOLO V4 [BWL20] consists of:

- Backbone : CSP Darknet53.

Backbone is a deep neural network composed mainly of convolution layers. The main objective of the backbone is to extract the essential features. YOLO V4 applies the Cross Stage Partial Network (CSP) to Darknet 53 which uses 53 convolution layers.

- Neck: Streaming Processor (SP) and Path Aggregation Network (PAN)

The neck is always added between the backbone and head which can detect objects at different scales and spatial resolutions. SP applies maximum pooling whereas PAN is applied to attract features in a heirarchical structure.

- Head: YOLO V3

The head of an object detector is responsible for classification and localization.

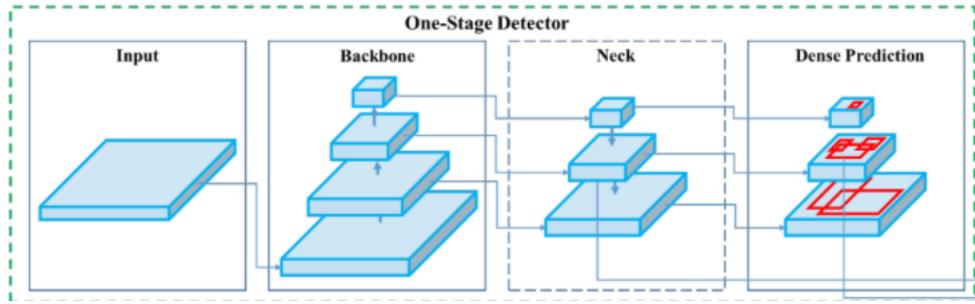


Figure 13.3.: Architecture of a single shot detector [BWL20]

13.5.2. Enabling GPU in Google Colab

In the Colab Notebook, the GPU can be enabled by clicking on Runtime and changing the Runtime type, hardware accelerator to GPU. Using the GPU will enable the YOLO V4 to make much faster predictions than the CPU.

13.5.3. Building Darknet

Darknet is an open source neural network framework written in C and CUDA. [BWL20] It is fast, easy to install, and supports CPU and GPU computation. The repository can be cloned from the github using the below command.

```
!git clone https://github.com/AlexeyAB/darknet
```

The next step is to navigate to the folder of the darknet and make necessary changes to enable the GPU and OpenCV.

```
1 GPU=1
2 CUDNN=1
3 CUDNN_HALF=1
4 OPENCV=1
5 AVX=0
6 OPENMP=0
7 LIBSO=0
8 ZED_CAMERA=0
9 ZED_CAMERA_v2_8=0
```

Figure 13.4.: Setting GPU and OpenCV to 1 in Makefile

The CUDA version can be verified using the below command

```
!/usr/local/cuda/bin/nvcc -version
```

```
[3] # verify CUDA
!/usr/local/cuda/bin/nvcc --version

nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Wed_Jul_22_19:09:09_PDT_2020
Cuda compilation tools, release 11.0, V11.0.221
Build cuda_11.0_bu.TC445_37.28845127_0
```

Figure 13.5.: Verification of CUDA version

The next step is to execute `!make`. This command builds darknet so that you can then use the darknet executable file to run or train object detectors. The successful compilation of the above command can be verified by checking the "darknet" file with no extensions in the darknet folder which will be created by this command.

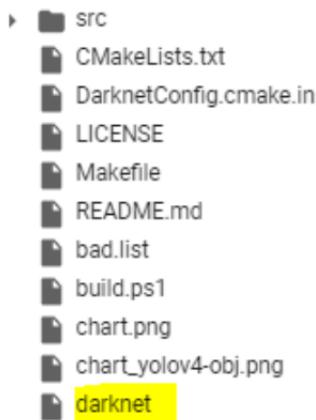


Figure 13.6.: darknet file with no extensions created after successful execution of make command

13.5.4. Define Helper Functions

These three functions are helper functions that will allow us to show the image in your Colab Notebook after running our detections, as well as upload and download images to and from our Cloud Virtual Machine (VM).

```
# define helper functions
def imShow(path):
    import cv2
    import matplotlib.pyplot as plt
    %matplotlib inline
    """%matplotlib inline, represents the magic function %matplotlib,
    which specifies the backend for matplotlib, and with the argument
    inline you can display the graph and make the plot interactive."""
    image = cv2.imread(path)
    height, width = image.shape[:2]
    """img.shape returns (Height, Width, Number of Channels)
    where Height represents the number of pixel rows in the image or the number of
    resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_
    #where 3*width and 3*height scale factors along x and y
    #the interpolation flag refers to which method we are going to use
    fig = plt.gcf()
    #plt.gcf() allows you to get a reference to the current figure when using pyp
    fig.set_size_inches(18, 10)
    plt.axis("off")
    plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
    plt.show()

# use this to upload files to the cloud Virtual Machine
def upload():
    from google.colab import files
    uploaded = files.upload()
    for name, data in uploaded.items():
        with open(name, 'wb') as f:
            f.write(data)
            print ('saved_file', name)

# use this to download a file from cloud VM
def download(path):
    from google.colab import files
    files.download(path)
```

Listing 13.1.: Useful Functions to show images, upload or download the images from Cloud VM

13.5.5. Moving the Datasets to Cloud VM

In order to create a custom YOLOv4 detector we will need the following:

- Labeled Custom Dataset
- Custom file `.cfg`
- `obj.data` and files `obj.names`
- `train.txt` file and `test.txt` files

The contents of these files are further explained below.

Firstly two ZIP folders are created for training `obj.zip` and validation datasets `test.zip`.

For training - Count of images

For validation - These should be uploaded to drive which can be later moved to the Cloud VM and unzipped in the directory “data”.

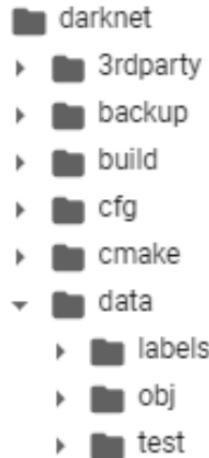


Figure 13.7.: Obj and Test folders containing training and validation datasets respectively

This can be done with the below commands:

To copy over both datasets into the root directory of the Colab VM:

```
!cp /mydrive/yolov4/obj.zip ../
!cp /mydrive/yolov4/test.zip ../
```

To unzip the datasets and their contents so that they are now in /darknet/data/ folder

```
!unzip ../obj.zip -d data/
!unzip ../test.zip -d data/
```

13.5.6. Configuring Files for Training

This step involves properly configuring your custom `.cfg`, `obj.data`, `obj.names`, `train.txt` and `test.txt` files.

It is important to configure all these files with extreme caution as typos or small errors can cause major problems with the custom training.

Cfg file: Firstly the `yolov4.cfg` is copied to the Google drive and modified.

The following modifications are made:

- batch = 64 and subdivisions = 16
- max_batches = 6000, steps = 4800, 5400 as the number of classes= 2. Usually this is decided by max_batches = (# of classes) * 2000 but the value should not be less than 6000. So I have chosen to use 6000. The Steps are decided by calculating (80 percent of max_batches), (90 percent of max_batches)
- The number of filters should be modified as filters = (# of classes + 5) * 3. As the number of classes=2, I have set this value as 21.
- The number of classes should be set in each YOLO Layer. There are 3 Yolo Layers in the `.cfg` file. Also the number of filters should be set in each Convolution layer just above every yolo layer.
- learning_rate=0.001
- width=416 (of the image)

- height=416

`obj.names` and `obj.data` files: These files should be created with the following contents. The `obj.names` file should have the classes in the order specified in the original `label.txt` file.

```
cylinder
cuboid
```

Figure 13.8.: Contents of the obj.names file

A backup folder should be created in our drive so as to save the weights. This will be The `obj.data` file has the paths to the backup, the number of classes, the paths to the `train.txt` and `test.txt` etc., The generation of `train.txt` and `test.txt` is further discussed below.

```
classes = 2
train = data/train.txt
valid = data/test.txt
names = data/obj.names
backup = content/gdrive/MyDrive/yolov4/backup
```

Figure 13.9.: Contents of the obj.data file

The files `obj.data` and `obj.names` should be uploaded to the cloud VM from our drive.

```
!cp /mydrive/yolov4/obj.names ./data
!cp /mydrive/yolov4/obj.data ./data
```

Generating `train.txt` and `test.txt` :

The last configuration files needed before we can begin to train our custom detector are the `train.txt` and `test.txt` files which hold the relative paths to all our training images and validation images.The scripts to generate these files are shown below.[The21]

```
import os

image_files = []
os.chdir(os.path.join("data", "obj"))
for filename in os.listdir(os.getcwd()):
    """
    os.listdir() method in python is used to get the list of all files and direct
    If we don't specify any directory,
    then list of files and directories in the current working directory will be return
    if filename.endswith(".jpg"):
        image_files.append("data/obj/" + filename)
    os.chdir("..")
    """
    Python method chdir() changes the current working directory to the given path.
    with open("train.txt", "w") as outfile:
        for image in image_files:
            outfile.write(image)
            ##Out-File uses the FilePath parameter and creates a file in the current
            outfile.write("\n")
        outfile.close()
    os.chdir("..")
```

Listing 13.2.: Script to generate `train.txt` file with paths to the training images

```

import os
image_files = []
os.chdir(os.path.join("data", "test"))
for filename in os.listdir(os.getcwd()):
    """ os.listdir() method in python is used to get the list of all files and direct
    If we don't specify any directory,
    then list of files and directories in the current working directory will be return
    if filename.endswith(".jpg"):
        image_files.append("data/test/" + filename)
os.chdir("../")
"""Python method chdir() changes the current working directory to the given path.
with open("test.txt", "w") as outfile:
    for image in image_files:
        outfile.write(image)
        #Out-File uses the FilePath parameter and creates a file in the current directory
        outfile.write("\n")
    outfile.close()
os.chdir("../")

```

Listing 13.3.: Script to generate test.txt file with paths to the testing images

These scripts can be executed by the commands:

```
!python generate_train.py
!python generate_test.py
```

13.5.7. Downloading pretrained weights for the convolution layers

This step downloads the weights for the convolutional layers of the YOLO V4 network. By using these weights it helps our custom object detector to be way more accurate and not have to train as long.

The command to download these weights [BWL20]

```
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet-
yolo_v3_optimal/yolov4.conv.137
```

13.5.8. Training the custom object detector

To train the model, we should execute the below command. `!./darknet detector train data/obj.data cfg/yolov4-obj.cfg yolov4.conv.137 -dont_show -map`

```
# train your custom detector! (uncomment %%capture below if you run into memory issues or your Colab is crashing)
%%capture
!./darknet detector train data/obj.data cfg/yolov4-obj.cfg yolov4.conv.137 -dont_show -map
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.828443), count: 23, class_loss = 0.617037, iou_lo
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.748139), count: 13, class_loss = 0.397307, iou_lo
total_box = 717874, rewritten_box = 0.000000 %
(next mAP calculation at 1600 iterations)
Tensor Cores are disabled until the first 3000 iterations are reached.
Last accuracy mAP@0.50 = 100.00 %, best = 100.00 %
1580: 0.384628, 0.452159 avg loss, 0.001000 rate, 6.973471 seconds, 101120 images, 9.435470 hours left
Resizing, random_coef = 1.40
```

Figure 13.10.: Training the model in Google Colab

13.5.9. Challenges

- Runtime restrictions : The Google Colab stops after 2000 iterations. We need to set up the google Colab VM after every interruption to train the model again.
- Error occurred that the GPU cannot be used any further.

- Lot of time taken for 2000 iterations approx 6 hours.

```
# train your custom detector! (uncomment %%capture below if you run into memory issues or you
# %%capture
!./darknet detector train data/obj.data cfg/yolov4-obj.cfg yolov4.conv.137 -dont_show -map
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.000000), c
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.891241), c
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.859803), c
total_bbox = 925474, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.000000), c
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.878810), c
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.862674), c
total bbox = 925488. rewritten bbox = 0.000000 %
```

Figure 13.11.: Training Failed after 2000 iterations due to Run-time limitations of Google Colab

Cannot connect to GPU backend

You cannot currently connect to a GPU due to usage limits in Colab. [Learn more](#)

If you are interested in priority access to GPUs and higher usage limits, you may want to take a look at [Colab Pro](#).

[Close](#)

[Connect without GPU](#)

Figure 13.12.: Unable to access the GPU

14. Training eines Gesichtserkennungsmodells mit Google Colaboratory

In diesem Kapitel werden wir Google Colab verwenden, um ein Objekterkennungsmodell zu trainieren und es dann in das TensorFlow Lite Format zu konvertieren, um es auf den Arduino Portenta H7 hochzuladen und zu validieren. TensorFlow Lite für Mikrocontroller wurde entwickelt, um maschinelle Lernmodelle mit begrenztem Speicher auszuführen. TensorFlow Lite für Mikrocontroller ist in C++ 11 geschrieben und benötigt eine 32-Bit-Plattform. Das Arduino Portenta H7 Board wird noch nicht unterstützt, um TensorFlow Lite Modelle mit der Arduino IDE [Goo19b] auszuführen, aber es ist einen Versuch wert.

14.0.1. Google Colaboratory

Google Colaboratory oder Colab ist ein Produkt von Google Research und ist eine kostenlose Jupyter-Notebook-Umgebung, die vollständig in der Cloud läuft. Es ist in der Regel gut für maschinelles Lernen und Datenanalyse geeignet. Das Gute daran ist, dass es keine Einrichtung erfordert und freien Zugang zu Rechenressourcen wie GPU und TPU bietet. [Goo21].

Vorteile von Colab

- **Vorinstallierte Bibliotheken:**

Es bietet vorinstallierte Bibliotheken für maschinelles Lernen, darunter PyTorch, Keras, TensorFlow

- **In der Cloud gespeichert:**

Wenn wir ein Jupyter-Notizbuch verwenden, dann wird alles auf einem lokalen Gerät gespeichert. Wenn wir jedoch Google Colab verwenden, können wir es in der Cloud speichern und von jedem Gerät aus darauf zugreifen, indem wir uns einfach beim Google Drive-Konto anmelden.

- **Zusammenarbeit:**

Es hilft dabei, mehreren Entwicklern, die an einem Projekt arbeiten, Zugang zu verschaffen, um den Code gemeinsam zu bearbeiten und den fertigen Code mit den Entwicklern zu teilen.

- **Kostenlose Nutzung von GPU und TPU:**

Ties ist wahrscheinlich die beste Funktion, denn Google Research lässt uns seine dedizierten GPUs und TPUs für unsere persönlichen Machine-Learning-Projekte nutzen. Daher müssen unsere Computer beim Training des Modells keine komplexen Berechnungen durchführen, da dies in der Cloud geschieht.

14.1. Datenbank

Die Datenbank für Gesichter ist die gleiche wie im vorherigen Kapitel, nämlich der UTKFace-Datensatz. Wir werden das Modell auf der Grundlage der Bildklassifizierungsstechnik trainieren, also werden wir eine andere Klasse verwenden, die in unserem Fall Blumen sind, und die Datenbank dafür kann von Google heruntergeladen werden.

Auswahl der Datensätze:

Bilder aus dem Blumendatensatz: 100 Bilder von Rosen, die in einem Ordner mit dem Namen „Roses“ gespeichert sind. Die Abmessungen der Bilder im Datensatz variieren, aber wir wählen Bilder mit Abmessungen im Bereich von 224×224 -Pixel. 14.1



Figure 14.1.: Beispielbilder aus dem Blumendatensatz

Bilder aus dem UTKFace-Datensatz: 120 Bilder, bestehend aus Gesichtern in einem Ordner, der als Faces gespeichert ist. Die Bilder haben eine Größe von 200×200 Pixel.

14.1.1. Verbinden mit Google Drive

Um ein Notizbuch in Colab zu erstellen und zu speichern, müssen wir es mit unserem Google Drive verbinden. Außerdem müssen wir unseren Bilddatensatz in unser Google Drive hochladen, damit wir von Colab aus darauf zugreifen können. Hierfür verwenden wir die Funktion `drive.mount('/content/gdrive')`. Diese leitet uns zu unserer Google-Anmeldeseite weiter und gibt uns den Autorisierungsschlüssel, den wir benötigen, um unser Google Drive mit Colab zu verbinden. Nach der Eingabe des Schlüssels werden wir mit unserem Google Drive verbunden.

14.1.2. Einrichtung des Datensatzes

Dieser Abschnitt ist ein wichtiger Teil davon, wo wir unsere gezippte Bilddatei und diese .ipynb-Skriptdatei (base_path) auf unserem Google Drive platziert haben, wie wir unsere gezippte Datei (imagezipped) benannt haben, die in unserem Fall `images_RF.zip` ist, und der Name der `data_dir`, der `data_dir='images'` ist, wo wir unsere Bilder vor der Komprimierung kompiliert haben. 14.2

```

[1] #set variables
data_dir = 'images'
base_path = '/mydrive/train'
imagezipped = 'images_RF.zip'
epochs = 10

[6] #first of all, we need to delete any images or saved model from the previous run, if any.
os.system("rm -rf {data_dir}".format(data_dir=data_dir))
# !rm -rf images
!rm -rf /tmp/saved_model

[8] #create output path if not yet existing
os.system("mkdir {base_path}/output".format(base_path=base_path))

```

Figure 14.2.: Einrichtung des Datensatzes

14.2. Vorverarbeitung von Daten

Rescaling:

Im Vorverarbeitungsteil werden wir das Bild zunächst mit der Skalierungsfunktion `rescale=1./255` umskalieren, da unsere Originalbilder aus RGB-Koeffizienten im Bereich von 0-255 bestehen, diese Werte aber für unser Modell zu hoch wären (bei einer typischen Lernrate), so dass wir stattdessen Werte zwischen 0 und 1 anstreben, indem wir mit einem Faktor von 1/255 skalieren.

Datenerweiterung:

Im Teil der Datenerweiterung werden wir Techniken zur Bildvergrößerung wie das Drehen von Bildern und das horizontale Spiegeln von Bildern verwenden. In diesem Fall beträgt der Drehbereich der Bilder 40 Grad, wie in Abbildung 14.3 dargestellt.

```
datagen_kwargs = dict(rescale=1./255, validation_split=.20)
dataflow_kwargs = dict(target_size=IMAGE_SIZE, batch_size=BATCH_SIZE,
                      interpolation="bilinear")

valid_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    **datagen_kwargs)
valid_generator = valid_datagen.flow_from_directory(
    data_dir, subset="validation", shuffle=False, **dataflow_kwargs)

do_data_augmentation = True #@param {type:"boolean"}
if do_data_augmentation:
    train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
        rotation_range=40,
        horizontal_flip=True,
        width_shift_range=0.2, height_shift_range=0.2,
        shear_range=0.2, zoom_range=0.2,
        **datagen_kwargs)
else:
    train_datagen = valid_datagen
train_generator = train_datagen.flow_from_directory(
    data_dir, subset="training", shuffle=True, **dataflow_kwargs)
```

Figure 14.3.: Neuskalierung und Datenerweiterung

14.3. Training des Modells

In diesem Fall haben wir ein vortrainiertes Bildklassifizierungsmodell verwendet, das auf MobileNet V2(224×224) basiert. Das Eingabebild sollte 224×224 Pixel groß sein. Das Modell wurde mit sequenziellen Keras-Schichten erstellt und die Dropout-Rate beträgt 0,2. Die Stapelgröße beträgt 32 und der in diesem Modell verwendete Optimierer ist Stochastic gradient descent (SGD) mit einer Lernrate von 0,005. Die hier verwendete Verlustfunktion ist die kategoriale Kreuzentropie. Wir haben den Epochenzähler auf 10 gesetzt. Epoche ist die Anzahl der Iterationen für die Rückwärtsfortpflanzung, um die Fehler im früheren Teil zu korrigieren, so dass die Genauigkeit erhöht wird. Je größer die Epoche, desto mehr Iterationen und desto länger der Trainingsprozess. 14.4

```

model.compile(
    optimizer=tf.keras.optimizers.SGD(lr=0.005, momentum=0.9),
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True, label_smoothing=0.1),
    metrics=['accuracy'])

steps_per_epoch = train_generator.samples // train_generator.batch_size
validation_steps = valid_generator.samples // valid_generator.batch_size
hist = model.fit(
    train_generator,
    epochs=epochs, steps_per_epoch=steps_per_epoch,
    validation_data=valid_generator,
    validation_steps=validation_steps).history

```

Figure 14.4.: Training des Modells

14.4. Prüfung des Modells

Hier testen wir das Modell an einem Bild aus dem Validierungsdatensatz. Wie in der Abbildung 14.5 zu sehen ist. sagt das Modell die Ausgabe, d. h. das Gesicht im Bild, genau voraus.

```
[ ] # Expand the validation image to (1, 224, 224, 3) before predicting the label
prediction_scores = model.predict(np.expand_dims(image, axis=0))
predicted_index = np.argmax(prediction_scores)
print("True label: " + get_class_string_from_index(true_index))
print("Predicted label: " + get_class_string_from_index(predicted_index))
```



True label: Faces
Predicted label: Faces

Figure 14.5.: Prüfung des Modells

Das trainierte Modell kann dann auf dem Sitzungsspeicher abgelegt werden.

14.5. Umwandlung des gespeicherten Modells in das tflite-Format

Um das gespeicherte Modell in das TFLite-Format zu konvertieren, verwenden wir die folgende Funktion `converter = tf.lite.TFLiteConverter.from_saved_model(saved_model)`

Nach der Konvertierung des Modells in tflite speichern wir die tflite-Datei auf unserem Laufwerk. Der nächste Schritt ist die Konvertierung der tflite-Datei in das C++-Format, damit wir das Programm in die Arduino-IDE hochladen können.

14.6. Konvertierung der tflite-Datei in eine Arduino-Header-Datei

Um das Objekterkennungsmodell in der Arduino-IDE ausführen zu können, müssen wir unser Modell im C++-Format programmieren und außerdem die aus unserem tflite-Modell erstellte Arduino-Header-Datei hinzufügen. Um eine Arduino-Header-Datei zu erstellen, verwenden wir Gitpod, um die tflite-Datei in eine Header-Datei (.h-Datei) zu konvertieren. Dazu müssen wir uns zunächst mit unseren GitHub-Anmeldedaten bei GitPod anmelden. Dann müssen wir einen neuen Arbeitsbereich erstellen. Im Befehlsterminal müssen wir diesen Befehl `xxd -i model.tflite model.h` verwenden. Dadurch wird eine Header-Datei erzeugt, die wir in unserem C++-Programm benötigen, um sie in die Arduino-IDE hochzuladen. Nachdem wir die Header-Datei heruntergeladen haben, müssen wir den Inhalt der Datei kopieren und mit dem Befehl `const unsigned char model_tflite[] =` in unser Programm einfügen. Dies stellt die Merkmale unseres Gesichtserkennungsmodells dar. Dann müssen wir die Gesichtserkennungsfunktion in C++-Code implementieren.

14.7. Offene Fragen

14.7.1. Arduino Portenta H7-Board nicht kompatibel für Objekterkennung mit Arduino IDE

Das Problem, das bei der Implementierung des Gesichtserkennungsmodells mit der Arduino IDE auftrat, ist, dass es keine früheren Arbeiten zur Objekterkennung mit dem Arduino Portenta H7-Board gab, da es nicht kompatibler war, so dass keine Daten verfügbar waren, die als Referenz für unser Gesichtserkennungsmodell verwendet werden konnten.[Goo19b] Auch die Implementierung eines TensorFlowLite Modells in der Arduino IDE ist eine schwierige Aufgabe, da unser Board nicht schnell genug ist und die Zeit zum Kompilieren eines Programms mehr als 20 Minuten beträgt und es keine Bibliotheken zur Implementierung des Objekterkennungsmodells gibt. Es gibt jedoch nur ein einziges online verfügbares Beispiel, das Hello World-Beispiel von Arduino, das auch als Sinuswellenfunktion im Tflite-Format bezeichnet wird und mit der Arduino-IDE ausgeführt werden kann, aber wir können dies nicht als Referenz nehmen, da das Objekterkennungsmodell und das Sinuswellenmodell völlig unterschiedlich sind.

Mögliche Lösung

Die mögliche Lösung könnte die Verwendung von OpenMV IDE anstelle von Arduino IDE sein. OpenMV IDE hat im Vergleich zu Arduino IDE mehr Funktionen, wie z.B. das Hochladen von Datensätzen in Echtzeit durch die Erfassung von Bildern vom Vision Shield. Es hat auch viele weitere Beispielprogramme wie Gesichtsverfolgung, Gesichtsfilter. Daher könnte die Verwendung von OpenMV IDE eine bessere und einfachere Lösung für dieses Problem sein.

14.7.2. Einsatz des auf Edge Impulse trainierten Objekterkennungsmodells mit Arduino IDE nicht möglich

Das Objekterkennungsmodell, das für die Gesichtserkennung in der Edge-Impulse-Plattform trainiert wurde, und die online generierte tflite-Datei konnten nicht auf der Arduino IDE eingesetzt werden. Das Modell funktioniert auf der Plattform einwandfrei, doch bei der Bereitstellung auf dem Arduino Portenta H7-Board unter Verwendung der Arduino IDE traten verschiedene Fehler auf. Nach Rücksprache mit

den Edge-Impulse-Plattform Schöpfer haben wir zu wissen, dass sie noch nicht bereit sind, in den Einsatz des Modells von ihrer Plattform mit Arduino IDE generiert.

Mögliche Lösung

Das von der Edge Impulse-Plattform generierte Modell ist in der Programmiersprache Python kodiert. Für die Arduino IDE ist die Programmiersprache C++. Daher hat Edge Impulse ab sofort einige Probleme mit der Arduino IDE, da einige Änderungen vorgenommen werden müssen. Die bessere Lösung wäre also die Verwendung der OpenMV IDE, da sie MicroPython unterstützt und das Edge Impulse-Modell eine Python-Datei erzeugt, die wir in die OpenMV IDE hochladen und unser Gesichtserkennungsmodell implementieren können.

15. Quellcode

15.1. Face Detection using Edge Impulse and OpenMV

```
import sensor, image, time, os, tf

sensor.reset()                      # Reset and initialize the sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # Set pixel format to RGB565 (or GRayscale)
sensor.set_framesize(sensor.QVGA)     # Set frame size to QVGA (320x240)
sensor.set_windowing((240, 240))      # Set 240x240 window.
sensor.skip_frames(time=2000)         # Let the camera adjust.

net = "trained.tflite"
labels = [line.rstrip('\n') for line in open("labels.txt")]

clock = time.clock()
while(True):
    clock.tick()

    img = sensor.snapshot()

    # default settings just do one detection... change them to search the image...
    for obj in tf.classify(net, img, min_scale=1.0, scale_mul=0.8, x_overlap=0.5, y_overlap=0.5):
        print("*****\nPredictions at [x=%d,y=%d,w=%d,h=%d]" % obj.rect())
        img.draw_rectangle(obj.rect())
    # This combines the labels and confidence values into a list of tuples
    predictions_list = list(zip(labels, obj.output()))

    for i in range(len(predictions_list)):
        print("%s = %f" % (predictions_list[i][0], predictions_list[i][1]))

    print(clock.fps(), "fps")
```


Part III.

Portenta H7

16. Einleitung

Quelle: <https://www.arduino.cc/pro/tutorials/portenta-h7>

17. Introduction

Quelle: <https://www.arduino.cc/pro/tutorials/portenta-h7>

Arduino Portenta H7 board is a dual core unit which has STM32H7 ARM Cortex®-M7 running at 480Mhz and Cortex®-M4 running at 240Mhz. This means it is capable of reading and executing two instructions at the same time. These two processors run the Mbed OS , which is an embedded real time operating system(RTOS) that is optimized for low-power microcontrollers. The interesting part of this dual core processor is that both the processors can communicate with each other and other peripherals on the board. This is done by a mechanism called Remote procedure Call(RPC) which helps in calling functions on the other processor. The two processors can run Arduino sketches on top of the Mbed OS, MicroPython/JavaScript via an interpreter, native Mbed applications and TensorFlow Lite.

The Arduino core sits on top of the Mbed OS and acts as a middleware which allows programs to leverage the Mbed OS APIs for storage, communication, security, and other hardware interfaces.

The portenta H7 also has an on-chip GPU Chrom-ART Accelerator, which makes it able to connect an external monitor to build a dedicated embedded computer. In addition, it also has a wireless module which is able to connect WiFi and Bluetooth simultaneously. The WiFi interface can also be utilized as an access point thereby creating its own WiFi network and allowing other devices to connect to it. Another interesting feature of portenta H7 is it has two 80-pin high density connectors at the bottom of the board in order to connect to other devices. It also has a USB type-C port which is used to power the board, connect to a display and can also be utilized as a USB hub. Some of the Industrial applications of the Arduino Portenta h7 include[Ard21a]

- High-end industrial machinery
- Laboratory equipment
- Computer vision
- PLCs
- Industry-ready user interfaces
- Robotics controller
- Mission-critical devices
- Dedicated stationary computer
- High-speed booting computation

18. Hardware

18.1. Portenta H7 - Hardware

The Arduino portenta H7 can simultaneously execute high-level code together with real time task. For instance, one core reads data from sensors or other devices and the other core could be used for applying machine learning techniques from the acquired data. It has two 80-pin high density connectors which enables scalability for different applications by just attaching the board with other compatible boards like Vision shield.[Ard21a]

Table 18.1.: Specifications Table

Main Processor	STM32H747XI dual Cortex®-M7+M4 32bit low power Arm®MCU
SDRAM	8-64 MByte option
QSPI Flash	2-128 MByte option
Ethernet	10/100 Phy option
Wireless	BT5.0 + WiFi 802.11 b/g/n 65Mbps option
Crypto chip	ECC608 or SE050C2 (Common Criteria EAL 6+) option
Display Connector	MIPI DSI host and MIPI D-PHY to interface with low-pin count large displays
GPU	Chrom-ART graphical hardware Accelerator
Timers	22x timers and watchdogs
UART	4x ports (2 with flow control)
SD Card	Interface for SD Card connector (through expansion port only)
Operational Temperature	-40°C to +85°C
Power	Through USB-C connector or LiPo battery (integrated charger)
Current Consumption	2.95 micro-Ampere in Standby mode (Backup SRAM OFF, RTC/LSE ON)
USB-C	Host / Device, DisplayPort out, High / Full Speed, Power delivery
MKR Headers	Use any of the existing industrial MKR shields on it
High Density Connectors	Two 80 pin connectors will expose all of the board's peripherals to other devices
ESLOV Connector	Arduino's open connector standard for self-identifiable hardware
Camera Interface	8-bit, up to 80 MHz
ADC	3 × ADCs with 16-bit max. resolution (up to 36 channels, up to 3.6 MSPS)
DAC	2 × 12-bit DAC (1 MHz)

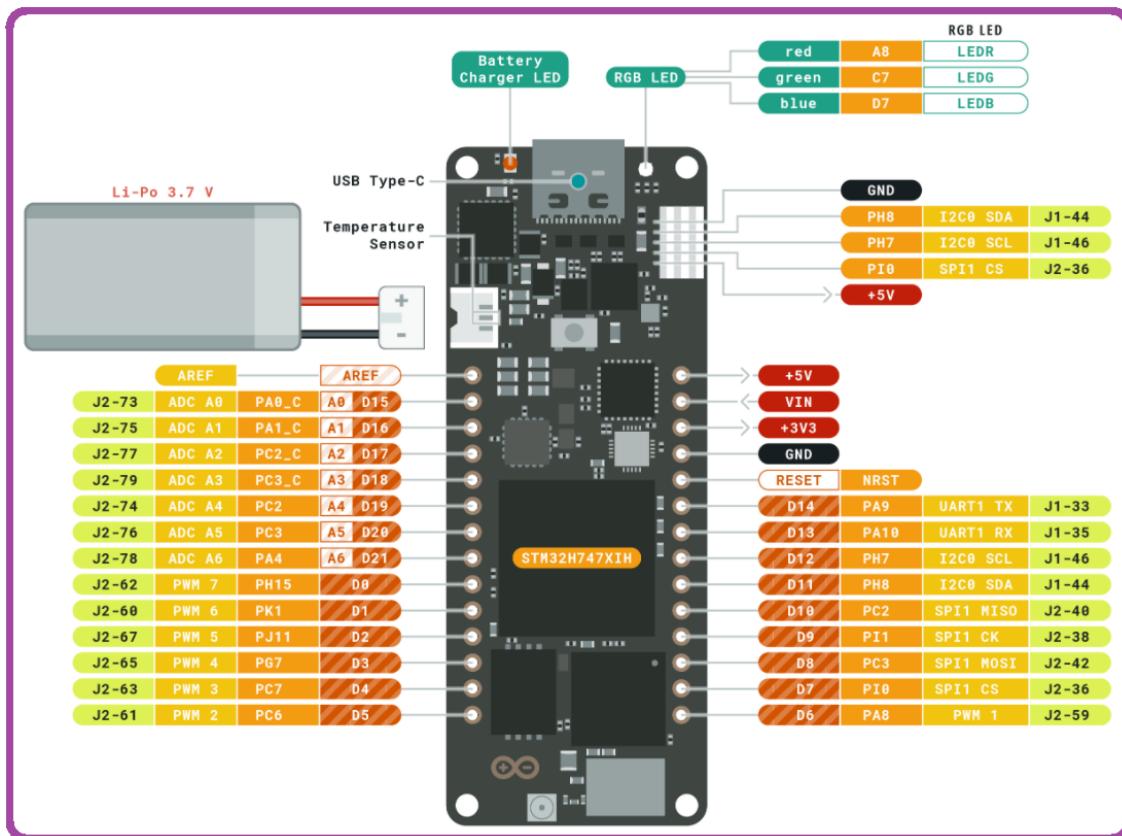


Figure 18.1.: Arduino Portenta Arduino Store

The above figure18.1 represents the Arduino Portenta H7 board layout. It has on the top a USB type-c connector along with two LEDs on its sides. One represents a battery charger LED which blinks while charging and the other LED glows when the Arduino Portenta is powered and ready to go. In addition to this, it has a temperature sensor and a blue reset button in the middle which resets the board. As can be seen from the figure, the pins A0 to A6 represents analog pins and D0 to D14 represents the digital pins. The RGB LED present on top of the board is useful while programming by changing the colors to represent different outputs. The notations to use in the Arduino sketches for the LED colors are represented as LEDB for blue, LEDG for green and LEDR for red. The ground pins (GND) are denoted in black color and power pins in red color. VIN is the input voltage pin. This pin is used to power the board when the USB source is disconnected. The 3v3 represents 3.3 volts output voltage pin and +5V pin is also a output voltage pin which outputs 5V from the board when powered using USB source or using the VIN pin. The SWD pin is a Serial Wire Debug pin which is used for debugging. On the back of the board, there are two 80-pin high density connectors to connect other boards. The pins for high density connectors are denoted as J1,J2 in the figure.18.1. There are PWM(Pulse Width Modulation) pins also present on the Arduino Portenta H7 board which are used in analog output demonstration like fading an LED. The ADC pins converts the analog input signal to digital output. It also has UART pins for communication with other serial devices. Apart from this, it also has I2C(Inter Integrated Circuit) port at the top right for communication between integrated circuits on the Arduino Portenta H7 board.

18.2. Tutorial

- <https://store.arduino.cc/portenta-h7>
- https://www.digikey.de/product-detail/de/arduino---bcmi-us-llc/ABX00042/1050-ABX00042-ND/11657626?utm_adgroup=Evaluation%20Boards%20-%20Embedded%20-%20MCU%2C%20DSP&utm_source=google&utm_medium=cpc&utm_campaign=Google%20Shopping_Development%20Boards%2C%20Kits%2C%20Programmers_New&utm_term=&productid=11657626&gclid=EAIAIQobChMIGprd57b06gIVCJzVCh2plwHSEAkYAiABEcBwE
- https://www.robotshop.com/de/de/arduino-portenta-h7-32-bit-arm-microcontroller.html?utm_source=google&utm_medium=surfaces&utm_campaign=surfaces_across_google_dede&gclid=EAIAIQobChMIGprd57b06gIVCJzVCh2plwHSEAkYASABEgLvtfd-BwE
- Tutorial

18.3. Hardware Installation

To use the Arduino Portenta H7 we need to power it by connecting it to the PC by a USB type-C cable or by connecting a battery to the slot provided on the board. It supports single cell Li-po or Li-Ion battery(3.7V). Here, we are going to connect the Arduino portenta H7 to a host laptop. Hardware and software requirements :

- Arduino Portenta H7 board
- USB type- C cable
- Arduino IDE 1.8.10 or newer version

To start with, we connect the board to the laptop using a USB C cable. The LED on the board starts blinking green when its connected to the laptop. In case the board doesn't respond via USB we can double press the reset button and it enters into bootloader mode and the LED starts to blink again indicating it as ready. In order to use Arduino vision shield, we should connect it to the Arduino Portenta H7 using the high density connectors. The Arduino vision shield is designed to sit on top of the Arduino Portenta boards

19. General Description of Softwares used in this project

19.1. Introduction

In this chapter we will be going through the description of the Arduino IDE and describe the features of it and what are all the options present in the IDE and how can we use it. Further we will be describing about the installation procedure of the Arduino IDE and how to get started with it.

19.2. Arduino IDE Description

It is an open source official Arduino software which used for editing, uploading and compiling codes in to the Arduino module. It is a cross-platform software which is available for Operating Systems like Windows, Linux, macOS. It runs on Java platform and supports a range of Arduino modules. It supports C and C++ languages. The microcontrollers present on the Arduino boards are programmed which accepts the information in the form of code. The program written in the IDE is called a sketch which will generate a Hex file which is then transferred and uploaded in the controller. The IDE environment is made up of two parts: an editor and a compiler. The editor is used to write the required code, while the compiler is used to compile and upload the code to the Arduino Module.[FAD18]

The Menu bar has options such as File in which there are many options including Opening a new file or existing, Examples-in which we can find sketches for different applications like Blink, Fade etc. There is an error console at the bottom of the screen for displaying errors.

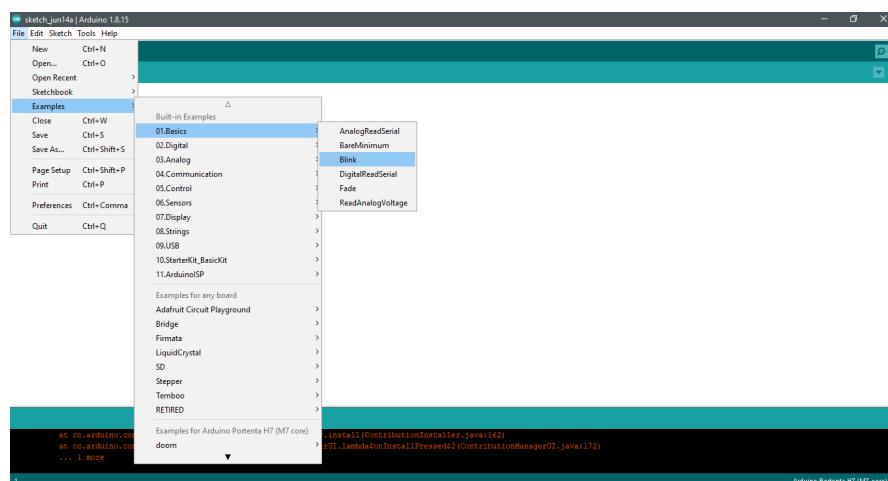


Figure 19.1.: Menu bar options

The 6 buttons are present on top of the screen are as follows

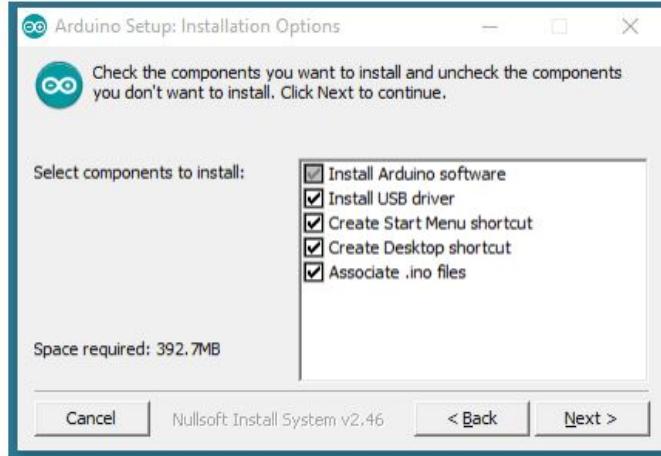


Figure 19.3.: Arduino Setup Installation options



Figure 19.2.: Menu buttons

- The check mark is used to verify your code. Click this once you have written your code.
- The arrow uploads your code to the Arduino to run.
- The dotted paper will create a new file.
- The upward arrow is used to open an existing Arduino project.
- The downward arrow is used to save the current file.
- The far right button is a serial monitor, which is useful for sending data from the Arduino to the PC for debugging purposes.

19.3. Installation

To install the Arduino IDE, we need to download the latest version from the Arduino webpage <https://www.arduino.cc/en/software>. We can select the version based on the operating system we are using. Here we are installing Arduino 1.8..15 for a Windows 10 operating system. The set up file name is arduino-1.8.15-windows.exe and the size of it is 1,17,470 KB. we can specify the path according to our needs. Here i have set the path as **C:/Program Files (x86)/Arduino**

After the download is done, open the setup file and proceed to install. Select all the components in the dialog box and click Next.

Select the destination folder and click Install

Once the installation is done, open the Arduino IDE and a default sketch appears on the screen as shown.

It can be seen from the figure 19.5 that the basic arduino sketch has two parts. The first part is a void setup() function which returns void and we do the intliazition such as the output LED color, specifying the core etc. The second part is the void loop() function where we define functions which are to be performed through out the loop. These codes are placed between paranthesis and each function has a return type, here it has void return type.

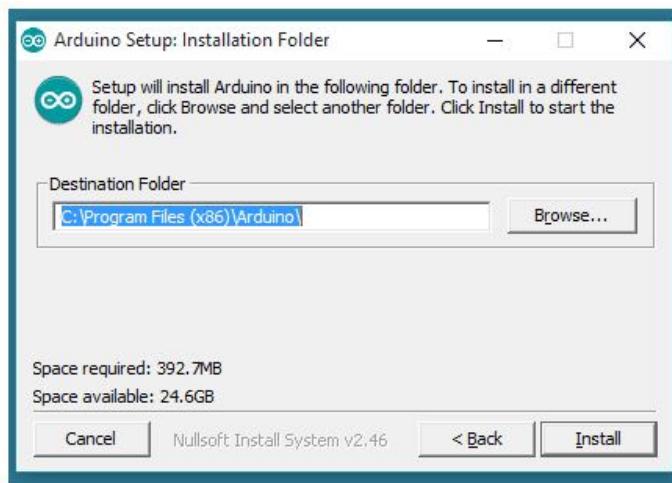


Figure 19.4.: Arduino Setup: Installation Folder

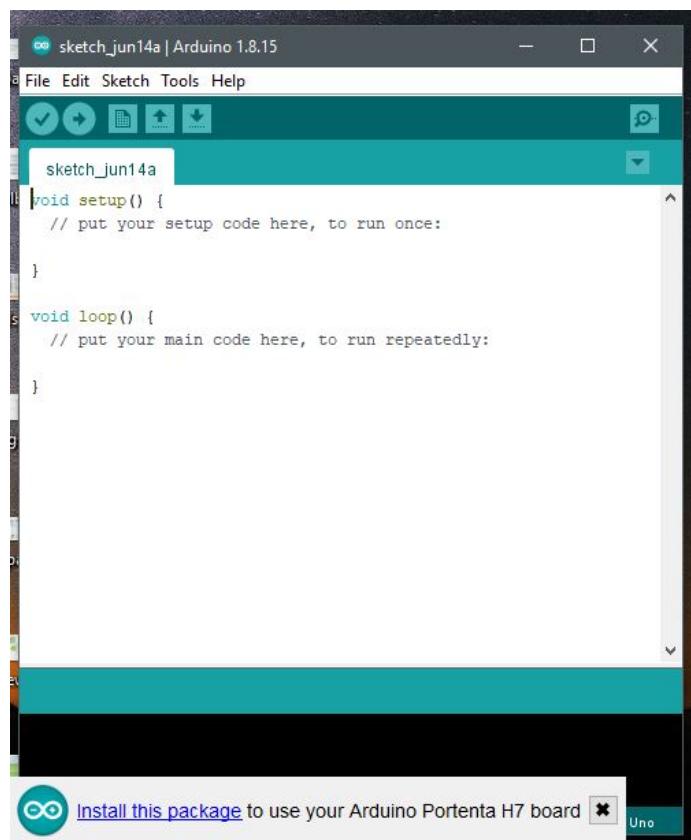


Figure 19.5.: Arduino Sketch

20. First Steps with the Portenta H7

20.1. Introduction

In this chapter we will be looking at how to connect the Arduino Portenta H7 with a PC/laptop in order to use the board. Then, we will be going through the first steps of installing the relevant packages to use the Arduino Portenta H7 board and then we will be implementing a basic example sketch from the Arduino IDE.

20.2. Configuration

Connect the Arduino Portenta H7 board to the computer via USB Type-C cable. Press the reset button twice on the board and the LED on the board starts blinking green as shown in the figure 20.1 indicating that it is ready.

First we need to install the packages in order to use the Arduino Portenta board. To do this search for Portenta in the boards and select Arduino Mbed OS Portenta Boards and click install as shown in figure 20.2. After the installation is done, we can start writing sketches in the IDE.

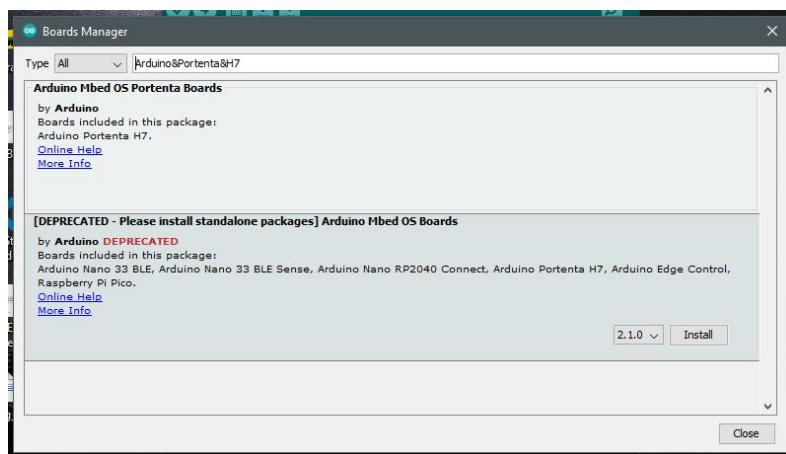


Figure 20.2.: Install packages for Arduino Portenta H7

20.2.1. Example Blink Sketch

To upload the example sketch, go to File then click on examples and then click on 01.Basics and then select blink as shown in figure 20.3.



Figure 20.1.: Arduino Portenta H7 Connected to a laptop

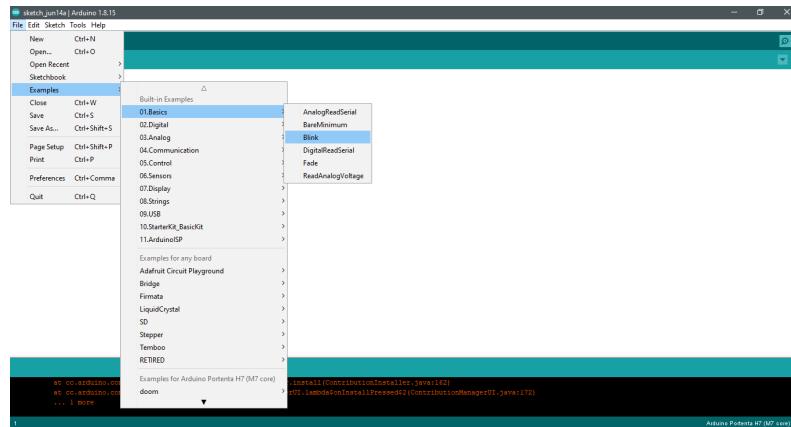


Figure 20.3.: Upload Basic Sketch

In the program, we initialize LED_BUILTIN as the output in the setup function. In the loop function we turn on the LED by using digitalWrite() function. Click on upload and wait for it to complete. After the upload is done, the green LED on the board starts blinking with a delay of 1000ms. The blink sketch appears on the screen as shown in figure 20.4.

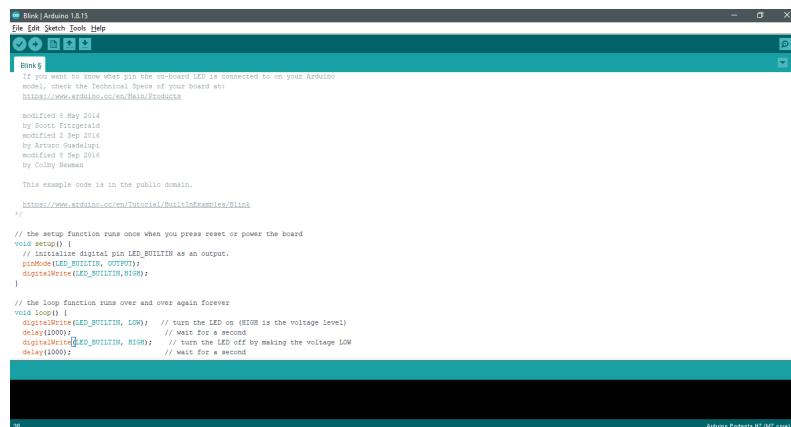


Figure 20.4.: Compile Blink Sketch

20.2.2. Serial Blink Example for Dual Core Processing

The following figure 20.5 represents the serial blink sketch. This sketch helps in demonstrating the serial data transmission by showing the LED blinking after a certain time period.

Here, Serial.begin(115200) command sets the data transmission rate to 115200 bits per second. The LEDB is the output which blinks in blue color and we have specified core M7. If we would like to use core M4 then we need to boot it first using bootM4() command. The ouput LED blinks blue in color after a delay of 3 seconds as defined in the sketch.

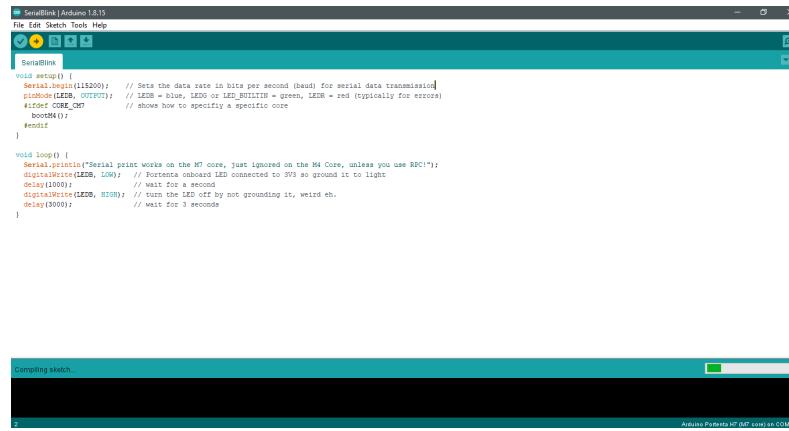


Figure 20.5.: Serial Blink Sketch

In order to use core M4, we need to de-select the core M7 from the board and select core M4. After selecting, we can see at the bottom the selected core for confirmation. Also, We need to write another sketch and now change the LED color to red in the sketch and add a delay of 4 seconds.

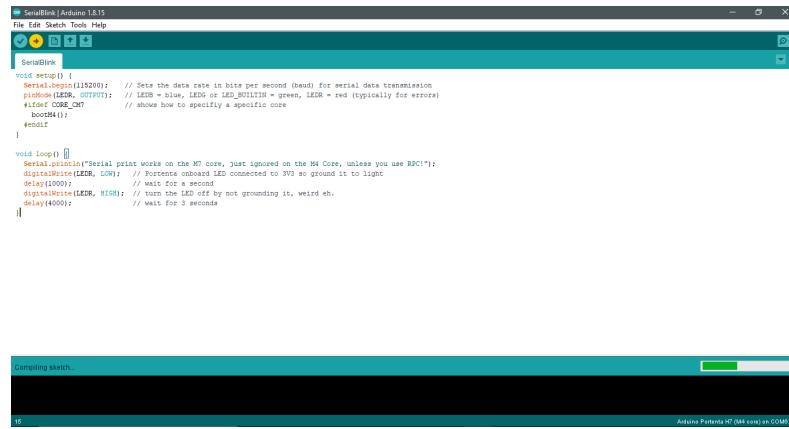


Figure 20.6.: Serial Blink Sketch

Notice that in the output both LED blue and green blinks and at a point in time they almost blink together resulting in a Lila color. This shows the dual core processing concept by using an LED.

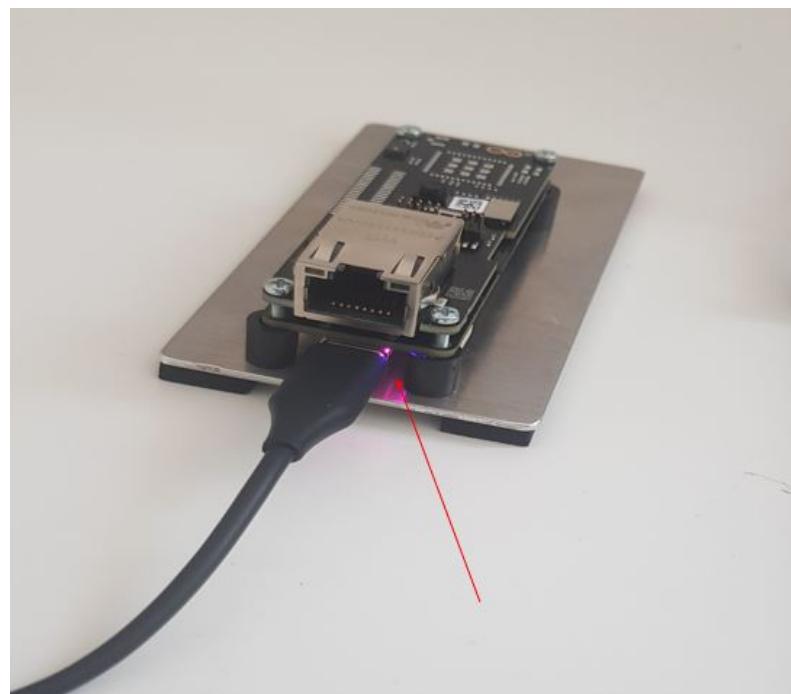


Figure 20.7.: Output LED color

It can be seen from the figure 20.7 that the output LED blinks in lila color due to the overlapping of the green and blue LED blinking at the same time.

21. First Steps with the Vision Shield

21.1. Introduction

In this chapter we will be going through the installation of OpenMV IDE and the steps required for connecting the Arduino Portenta H7 with the Vision Shield using the OpenMV IDE. Further, we will be looking at the overview of the OpenMV IDE and what are the tools provided in the IDE. Finally, we will be implementing some of the example programs provided in the OpenMV IDE.

21.2. Update Bootloader version

The OpenMV IDE is a premier integrated development environment which is a software tool to use the openMV camera present on the Arduino vision shield. It has a text editor ,frame buffer viewer with a histogram display and a debug terminal. It can be used across platforms, supporting Windows, Mac OS, Linux and Raspian. It was built for machine vision applications and is programmed in python[Tea21c]. Now we need to connect our Arduino Portenta board with the vision shield. To do this, first we need to update the bootloader version using the PortentaH7_updateBootloader sketch in the examples from the arduino IDE[RG20]. After connecting the Arduino Vision shield with Arduino Portenta H7 board and plugging in to the laptop , go to the Arduino IDE and select examples and click on Portenta system and then select PortentaH7_updateBootloader.

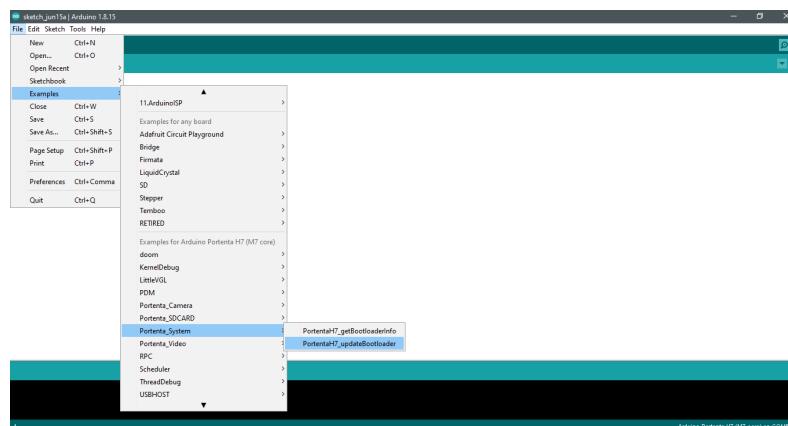


Figure 21.1.: Update Bootloader

Then run this sketch by selecting the right port.

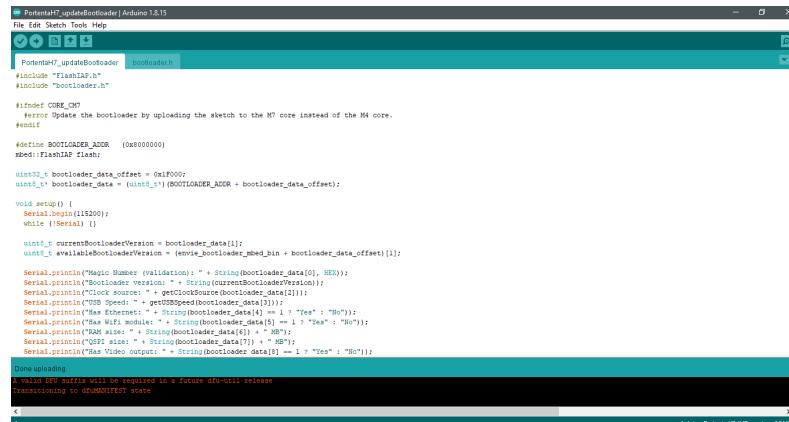


Figure 21.2.: Upload Bootloader sketch

After the upload is done, go to Tools and click on Serial monitor. A dialog box opens asking to update the bootloader version. Then press 'y' and enter. Now the bootloader version is updated.

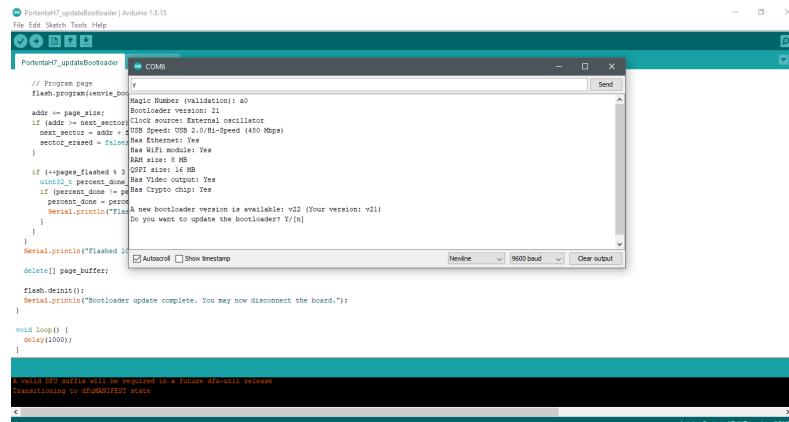


Figure 21.3.: Updating Bootloader

21.3. OpenMV Examples

21.3.1. Creating a basic face filter with OpenMV

In this example we use a smiley image in portable bitmap image(.pbm) format because OpenMV supports bmp, pgm or ppm image formats and not the other image formats such as .png [SR21]. This image will be overlaid on the face when it detects a face in the camera stream. Download the image and copy it into the flash drive. First we load the image into a variable named as `faceImage` using the `Image()` function. We use `copy_to_fb` to false because we don't want it to be copied automatically in to the frame buffer. Next we calculate the scale ratio to scale the bitmap image to match the detected face in the camera stream using the function `scale_ratio=faceWidth/faceImage.width()`. Then we draw the bitmap image on the detected face using the function `draw_image()`. Here the face detection is done using the Haar cascade algorithm. In this algorithm, it has multiple stages in which the output of one stage gives additional information to the input of next stage in the cascade. By doing this, it calculates the edges, lines, contrast checks in different stages and ultimately detects the face and draws a bounding box around it.

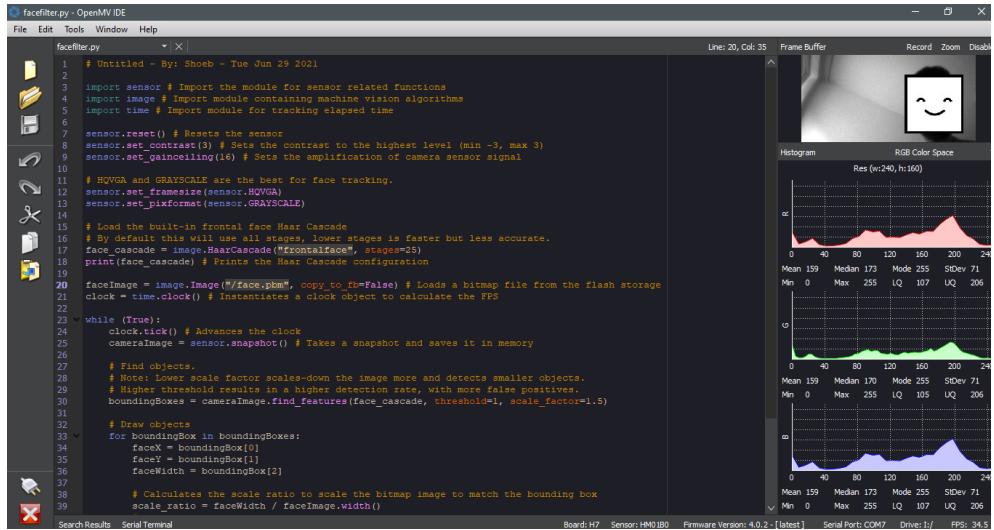


Figure 21.4.: Face Filter output

As we can see from the above figure 21.4 that as soon as a face is detected in the camera stream, the bitmap image which was loaded earlier is overlapped on to the detected face.

21.3.2. Face tracking

In this example, we are going to detect face and draw a bounding box around it. It uses the Haar cascade algorithm to detect faces. To do this, go to files in the OpenMV IDE and open examples and select `face_tracking`. In the program, firstly, we reset the sensor using `sensor.reset()` function and we set the frame size to QVGA and set the pixel format to grayscale. Next we load the Haar cascade using the function `image.HaarCascade()`. Next we use the keypoints feature of OpenMV camera to track a face after it is detected by Haar cascade algorithm. For this, first we set the keypoints to none and when the face is detected we will use the function `image.find_keypoints()` function. Then by using `draw_rectangle()` function we draw a rectangle around the detected face and draw keypoints using `draw_keypoints()` function. After this, we extract keypoints in the whole frame and then match this keypoints with first keypoints which was used only for face using `image.match_descriptor()` function. If the keypoints match is greater than 5 then we draw a rectangle around the matched feature.

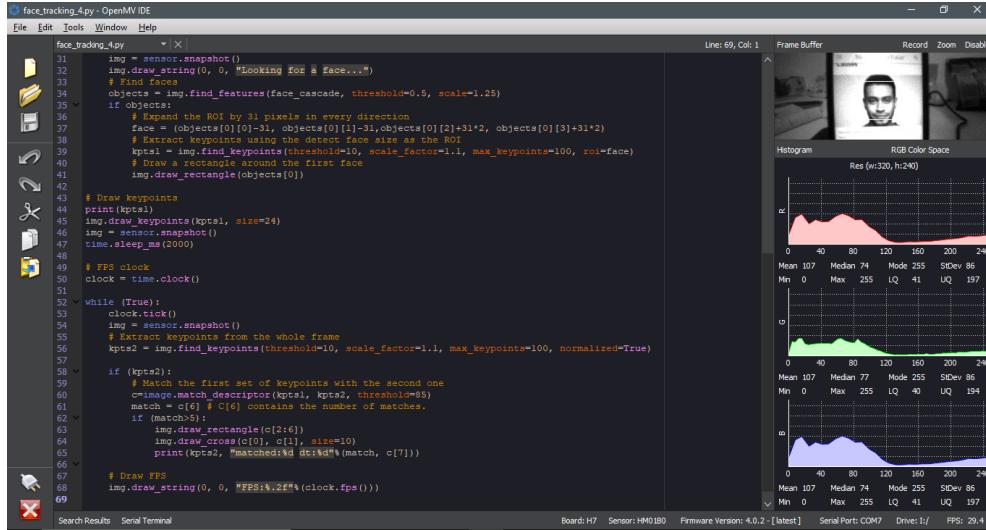


Figure 21.5.: Face tracking output

After we run this sketch, we need to point the vision shield camera towards a face and focus it properly. As soon as the face is detected in the camera stream , a rectangular box is formed around it indicating the detected face in the image. In this example, we can see that it clearly identifies the face and the bounding box is also generated around it.

21.3.3. Finding circles around the objects

In this example, we find the circles in an image using Hough transform algorithm. This algorithm is used in image processing for extracting the features such as circles or ellipses in an image. For programming this, we first reset the sensor as described in earlier examples. Then we take snapshot from the camera stream using `sensor.snapshot()` function. Then we find the circle in the image using `find_circles()` function. Lastly we draw circles using `draw_circles()` function.

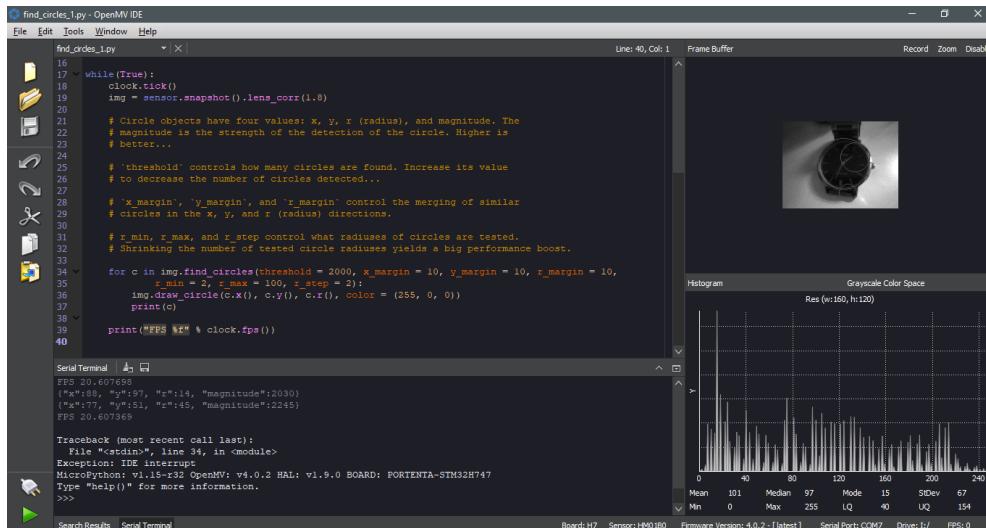


Figure 21.6.: Find circles output

As can be seen from the figure 21.6, the ouput shows that the circles are drawn on the

border of the watch. we need to bring the objects closer to the camera so that the focus is set properly in order to detect the circles around the objects.

21.3.4. QR Code information extraction

In this example we extract the information from QR code by using the vision shield. This is an interesting feature of the openMV camera present in the vision shield. In order to program this, we first follow the same initialization steps as mentioned in the earlier examples. We then take a snapshot from the camera stream using the `sensor.snapshot()` function and then we do the lens correction to 1.8 in order to get a better quality image using `lens_corr()` function. To extract the QR code we use `find_qrcodes()` function and then we draw a rectangle around the QR code using `draw_rectangle()` function.

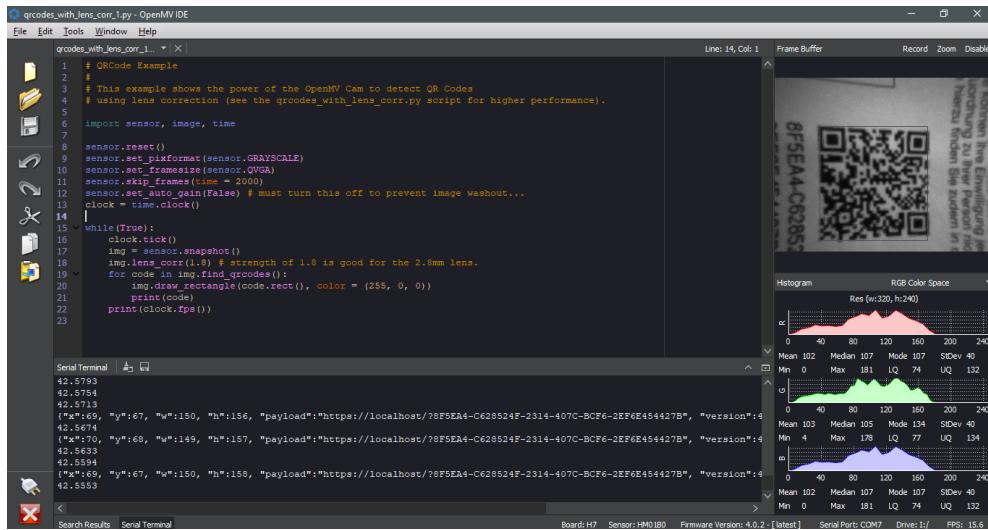


Figure 21.7.: QR Code output

In the output figure 21.7 we can see that a rectangular box is created around the QR Code and the information is extracted from it and it can be seen in the output serial terminal.

22. MicroPython & OpenMV

22.1. Introduction

In this chapter we will be looking at MicroPython and OpenMV in detail. We will be describing some of the features of the MicroPython programming language and how we can run MicroPython scripts on the Arduino Portenta H7 board by using the OpenMV IDE. We will be implementing a basic MicroPython script in the OpenMV IDE to blink LED's on the Arduino Portenta H7 board.

22.2. MicroPython

MicroPython is a tiny Python programming language interpreter that works on small embedded development boards and is open source. Instead of using complex low-level languages like C or C++, MicroPython allows us to write clean and simple Python code to control hardware (what Arduino uses for programming). It is a lean and efficient Python 3 implementation that includes only a tiny part of the Python standard library and is optimized for use on microcontrollers and in constrained environments and it is compact enough to fit and run within just 256k of code space and 16k of RAM.[RG20] It has unique features which makes it different from other embedded systems such as :

- **Interactive REPL(Read-Evaluate-Print-Loop)** : This helps in connecting to a board and execute the code without any need for compiling or uploading. It is a four step process in which it first reads the user input, evaluate the code, print any results and loop back to first step.
- **Extensive software library** : MicroPython like the normal python programming language has libraries built in to support many tasks. For example parsing JSON data from a web service, searching text with a regular expression, or even doing network socket programming is easy with built-in libraries for MicroPython.
- **Extensibility** : MicroPython is extensible with low-level C/C++ functions which means we can combine high level MicroPython code with faster low level code when needed.

The MicroPython programming language implements the core python 3 language but it can't implement the entire standard library of Python 3 because trying to fit big libraries into tiny boards with just kilobytes of memory is not possible. One thing to note is that MicroPython is only a programming language interpreter whereas Arduino is an entire ecosystem which has the Arduino IDE and the hardware which in this case is Arduino Portenta H7.

22.3. Implementing MicroPython on Portenta H7 with OpenMV IDE

MicroPython scripts can be run on the Arduino Portenta H7 with the help of OpenMV IDE[SR21]. OpenMV comes with its own firmware which is built on MicroPython.

We can explore the features of the Arduino Portenta H7 by using different classes and modules provided in MicroPython.

In order to implement MicroPython on the Arduino Portenta H7, we first need to connect the Arduino Portenta H7 with the computer using a USB -C cable and flash it by double pressing the reset button. After the flashing is done the bootloader gets updated while flashing the green LED. Now we open the OpenMV IDE and click on connect option provided at the bottom of the screen as done in the earlier chapters. This will install the latest OpenMV firmware to the Arduino Portenta H7 board. After the firmware is updated, we need to click on the play button to connect the board with OpenMV IDE and start programming in the IDE. To be able to check whether MicroPython program can run on the Arduino Portenta H7 we write a simple program in MicroPython to blink LED's on the board using the OpenMV IDE.

22.3.1. Example script for blinking builtin LED on the Arduino Portenta H7 with OpenMV

In the program we first import pyb module which contains board related functionality such as PIN handling. Then we create variables to control the built in LED's using `pyb.LED()` function. Then we switch on the red,blue and green LED's with a delay of 1000ms using `redLED.on()` function and off the LED using `redLED.off()`. Now after the program is complete we hit the play button in the bottom to upload the script.

```
import pyb
redLED = pyb.LED(1) # built-in red LED
greenLED = pyb.LED(2) # built-in green LED
blueLED = pyb.LED(3) # built-in blue LED
while True:
    # Turns on the red LED
    redLED.on()
    # Makes the script wait for 1 second (1000 milliseconds)
    pyb.delay(1000)
    # Turns off the red LED
    redLED.off()
    pyb.delay(1000)
    greenLED.on()
    pyb.delay(1000)
    greenLED.off()
    pyb.delay(1000)
    blueLED.on()
    pyb.delay(1000)
    blueLED.off()
    pyb.delay(1000)
```

We can see from the figure 22.1 that the output is flashing red,blue and green LED's at a time interval of 1000ms. Therefore we can say that the MicroPython scripts can be run on the Arduino Portenta H7 with the use of OpenMV IDE

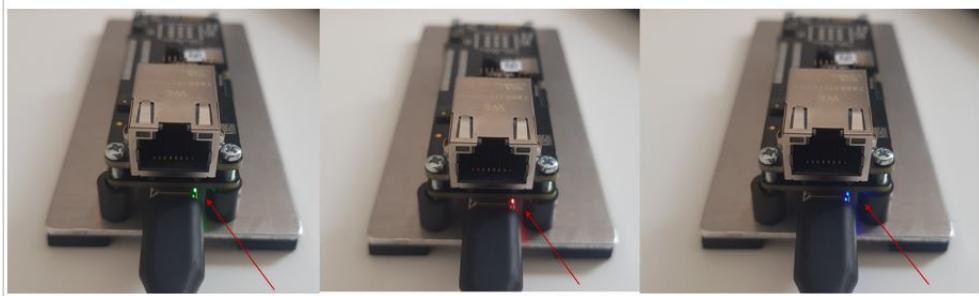


Figure 22.1.: Output RGB LED blinking red , blue and green

23. First Application: Object Detection

23.1. What is Face Detection

Face detection is Artificial Intelligence based computer technology which is used to detect human faces in images. It determines the location and size of human faces in an image or video stream and after the face is detected it should return a bounding box on detected faces in it. It is considered as a quite challenging task in the field of computer vision because human face is a dynamic object and has great degree of variations in it. Face detection is used in many applications such as biometric security systems in which it is possible to track and monitor people in real-time.

There are two types of approaches when detecting a face in an image, 1) Feature based approach 2) Image based approach.

1) Feature-based approach : In this type of method the different features of the face are found in the image for example detecting the eyes, nose eyebrows, mouth. As these features are consistent in an image which contains face despite other variabilities like lighting and poses. So the main idea behind this approach is to extract these features using edge detection techniques and based on these extracted features statistical models are built to describe the relationship and detect a presence of face in an image.
2) Image-based approach : The image based method try to learn templates from examples in images. That means this method relies on machine learning and statistical analysis techniques to find the facial characteristics to declare whether the image has face or no face present in it. In this method, neural networks are used for detection.

23.1.1. Challenges

The challenges in face detections are the reasons for less accuracy and detection rates in images. Some of the challenges include :

- Lighting Conditions : The lighting surrounding an image maybe low or high in parts of image which makes it difficult to detect faces.
- Distance : If the distance of the camera towards a face is very low or high then also it becomes difficult for detection.
- Orientation : The face orientation and angle to the camera can also cause detection failures.
- Complex background : If the background of the image contains more number of objects then it will become a difficult task for detection.
- Low resolution : If the image resolution is very low or the image contains noise then the accuracy of the detected face will be less.

23.1.2. Solutions

- Optimal lighting conditions should be considered when trying to detect a face in an image. A good evenly illuminated image helps in easy detection of face in an image.
- The distance between the camera and face should be optimum. In our case, a distance between 7cm-10cm is best suited for face detection.

- The number of training images with different angles and orientation should be taken to improve the detection rate in images with different orientations of faces.
- The background of the image should also be considered in such a way that the face which we are going to detect is not occluded because of more number of objects present in the image.
- Using state of the art methods like convolutional neural networks(CNN) for face detection helps significantly in detection rate.

23.1.3. Applications

- Quality Testing : Testing the quality of finished products in a production plant
- Surveillance : It is used for face detection in a crowd such as a public place or private place.
- Attendance : It can be used to detect the attendance of humans in a class and when combined with biometric security system it can also be used in access management in a building.
- Photography : Mobile phone cameras detect faces for autofocus in an image.

24. Training a Custom Machine Learning Model for Portenta H7

24.1. Introduction

In this chapter we are going to look at how to train a face detection model and what are all the steps required in training a model using the Edge Impulse platform. Further we will be looking at how to deploy the tflite model and implement in the OpenMV IDE the face detection model.

24.2. How to train a face detection model

Machine learning on microcontrollers is an interesting aspect as they can run on low power batteries for a long period of time. We can even put the microcontroller to sleep and only activate it when the camera or any other sensor notices an activity. The other interesting aspect is that machine learning models on a microcontroller can run without internet connection as they don't need to upload data to the cloud. Therefore the data is only present in the device which ensures data protection.[SR21]

24.3. KDD Process

24.3.1. KDD Process for the object detection model

Knowledge discovery in databases is the process of discovering useful knowledge from a collection of unstructured or structured data. It involves different phases such as data selection, preparation, timing data, mining, finding patterns, evaluation and verification. This process can be applied to the Edge Impulse platform in determining how the platform works in the background. The process starts with a database consisting of images of objects which are to be trained in the model in order to be detected. Then a training dataset is prepared from the database and data processing is done before going for actual training. Then the model is trained, in this case using a Convolutional Neural Network(CNN) and the model is evaluated. Depending on the output the model is either fine tuned or changes in training dataset is made accordingly. This KDD process is described in detail for the face detection model in the next sections.

24.3.2. Database

In order to train a machine learning model for face detection we first need to create a dataset of images consisting of faces. We can do this by capturing the images using the Vision Shield camera. We can also download a dataset online and use it to train the model. UTKFace is one dataset with long age span(0 to 116 years old) consisting of over 20k single face images. We can download this dataset from the link <https://susanqq.github.io/UTKFace/>

Selection of Dataset : Manually captured images from the camera present on the Vision shield, UTKFace dataset.

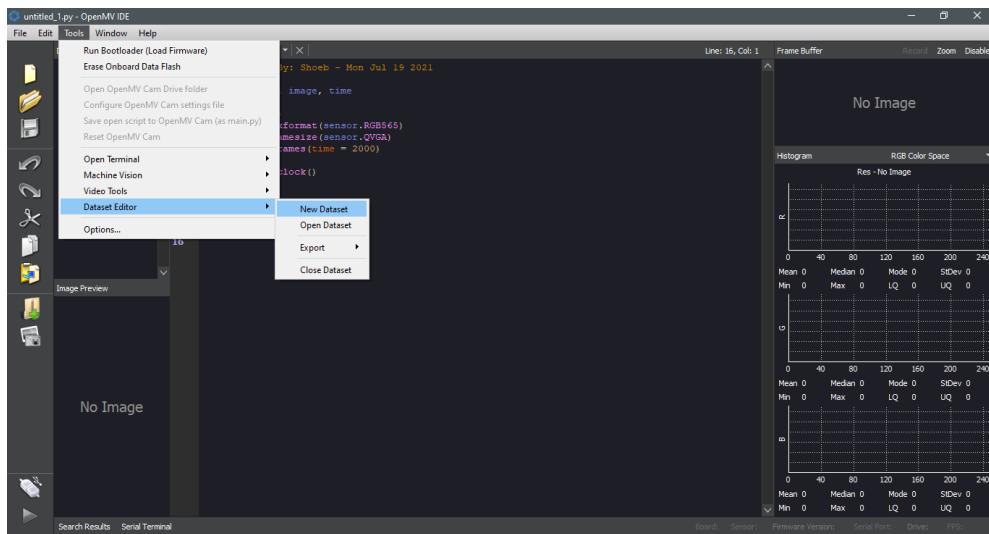


Figure 24.1.: Add Dataset

24.3.3. Add Dataset in the OpenMV IDE

In order to add dataset in the OpenMV IDE, go to Tools then select Dataset Editor and click on New Dataset as shown in the figure 24.1. Then select an empty folder in the drive where all the images will be stored.

Then we defined two classes of images. Firstly, we created face class and added images of face from the UTKFace dataset. Secondly, pen class is created by capturing pictures of pens of different types and at different positions using the Vision Shield directly from the OpenMV IDE. From the figure 24.2 we can see that two folders namely face.class and pens.class are created in the Dataset editor. We do this to help the model learn in distinguishing face from other objects, in this case a pen.

Now that our dataset is prepared, we can now go for training the model. To do this, we need to create a project in Edge Impulse platform.

Firstly, we need to sign up / login in the Edge Impulse account by going to their webpage <https://www.edgeimpulse.com/> and create a new project named Face_detector.

24.3.4. Export the Data to Edge Impulse platform

Now in the OpenMV IDE we need to export our data to the Edge Impulse website. To do this, click on Tools and then select Dataset Editor and then select Export and then select upload to Edge Impulse Project as shown in the figure 24.3

24.3.5. Data Selection

Now go to the Edge Impulse website and click on data acquisition. From the figure 24.4 we can see that all of our images with two labels have been exported from the OpenMV IDE to the Edge Impulse platform. It automatically divides the dataset in to training and testing data. Here it has divided into 487 items in training data and 51 items in testing data.

24.3.6. Data Transformation

Now click on Impulse design and set the image width and height in the Image data field, add processing block which is image in this case and apply transfer learning as

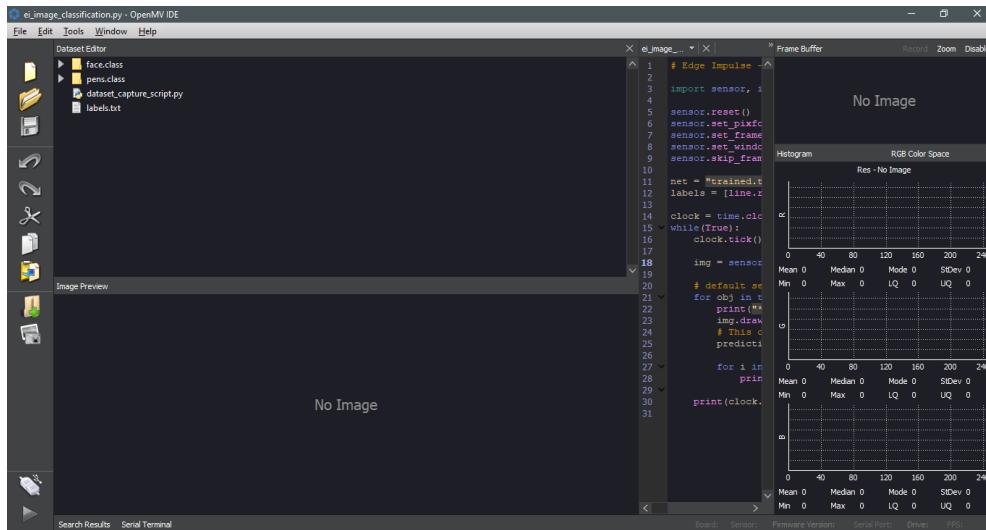


Figure 24.2.: Dataset class Folders

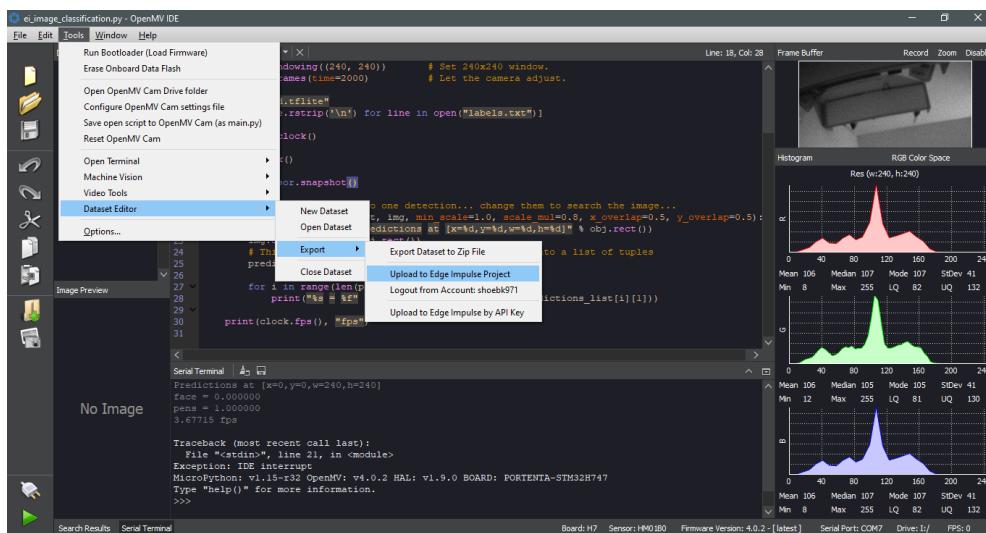


Figure 24.3.: export to Edge Impulse

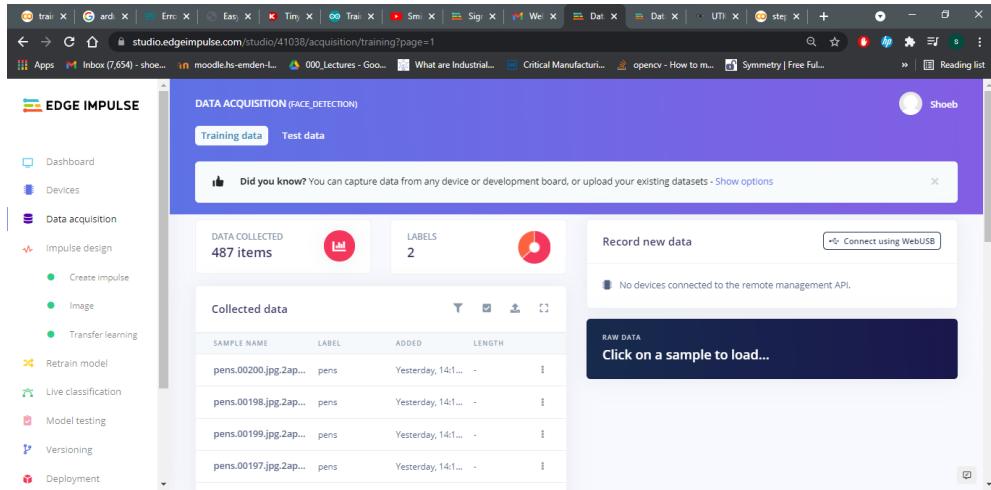


Figure 24.4.: Data Acquisition in Edge Impulse

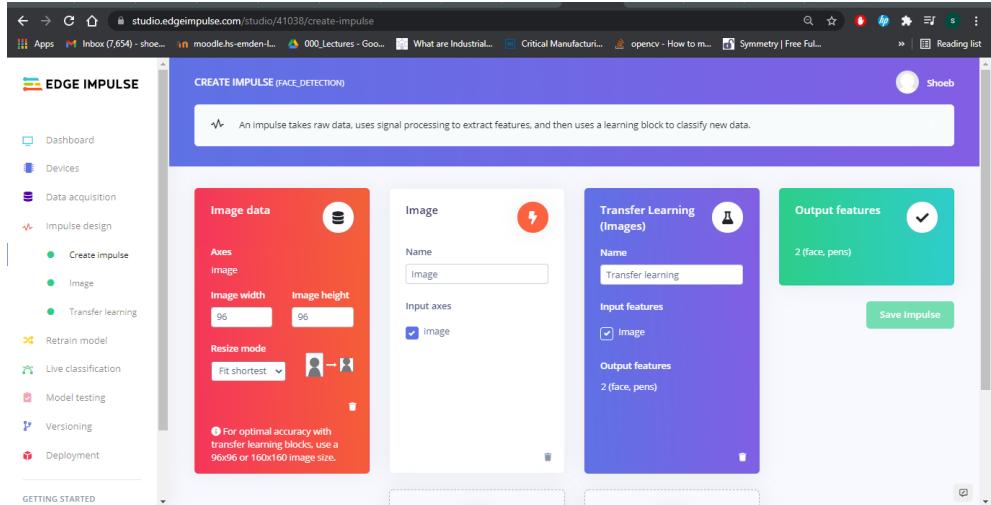


Figure 24.5.: Impulse Design

shown in the figure 24.5. After this, click on save impulse.

Now click on Image tab on the left side of the screen and select the color depth to Grayscale and click on save parameters as shown in the figure 24.6.

Now click on Generate features and here we can see the number of items in the training dataset and number of classes as depicted in the figure 24.7. click on Generate features. After the process is done we can see the feature explorer which contains the features of the dataset we have added and we can see that there is a clear difference between features of a face and a pen as shown in the figure 24.8.

24.3.7. Data training

Now click on Transfer learning and select the number of training cycles which is epochs(50 in this case). Then we select the architecture model for the convolutional neural network model which we chose MobilenetV2 here. We can also edit the python script to train the model by clicking on neural network settings and selecting switch to Keras(expert) mode option as shown in figure 24.9. The model uses Adam optimizer and Adadelta optimizer.

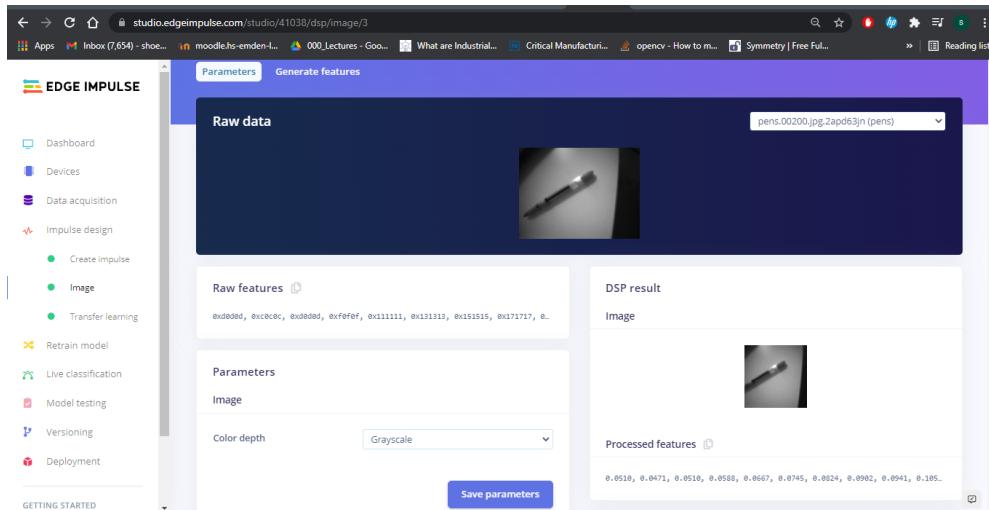


Figure 24.6.: Save Parameters

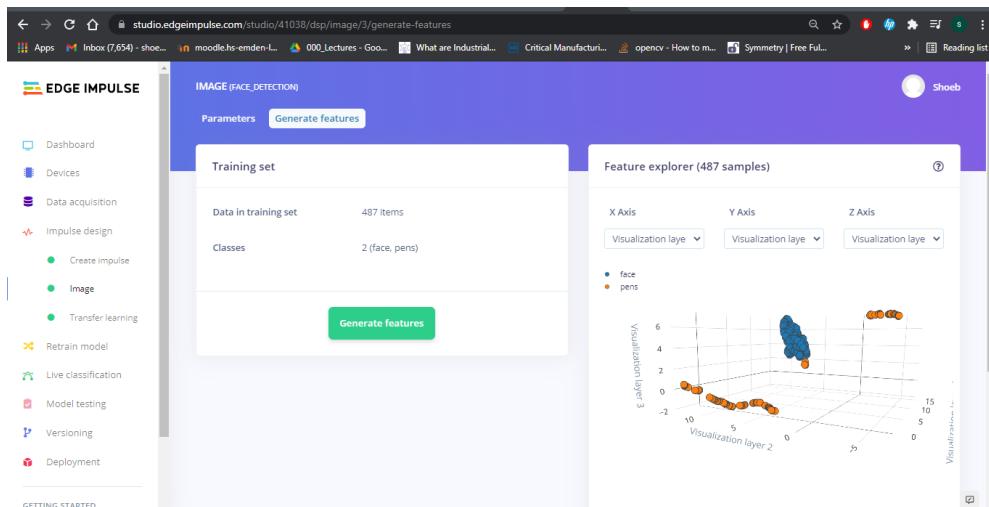


Figure 24.7.: Generate Features

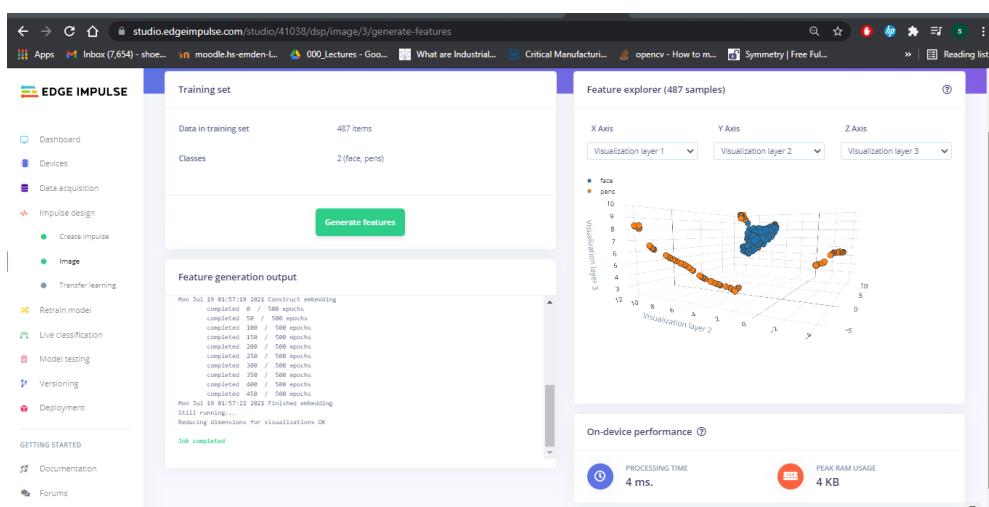


Figure 24.8.: Feature Explorer

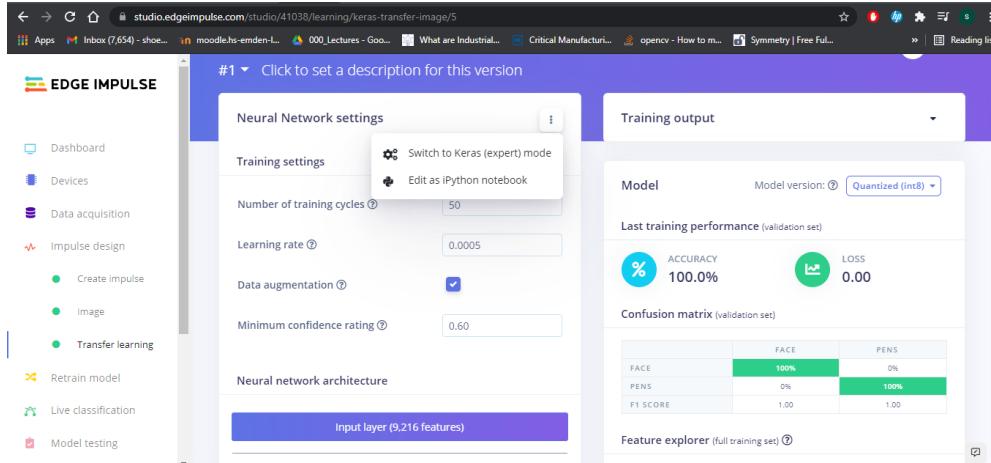


Figure 24.9.: Edit Neural Network Program

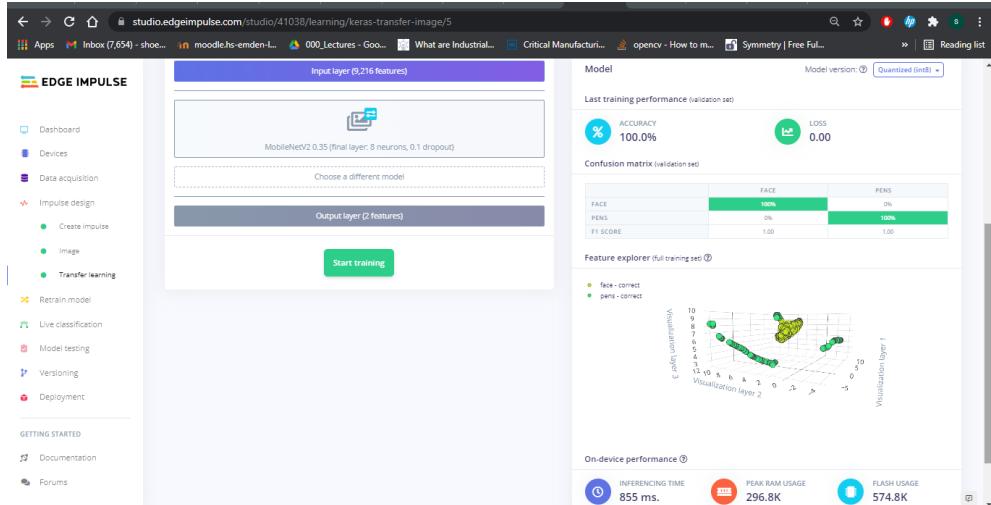


Figure 24.10.: Model output

Now click on start training. After the training process is completed we can see the output accuracy is 100% and the loss percentage is 0 as shown in the figure 24.10.

24.3.8. Deploy to Arduino Portenta H7

In order to deploy the trained model to the Arduino Portenta H7, we go to the Edge Impulse menu options and click on Deployment and then select OpenMV and click on Build as shown in the figure 24.11. This will create a zip file which consists of TensorFlow Lite file named as trained.tflite and lables.txt along with the python script ei_image_classification.py. After downloading this we need to copy trained.tflite file and labels.txt file and paste it in the USB drive where our Arduino Portenta H7 is connected to the laptop.

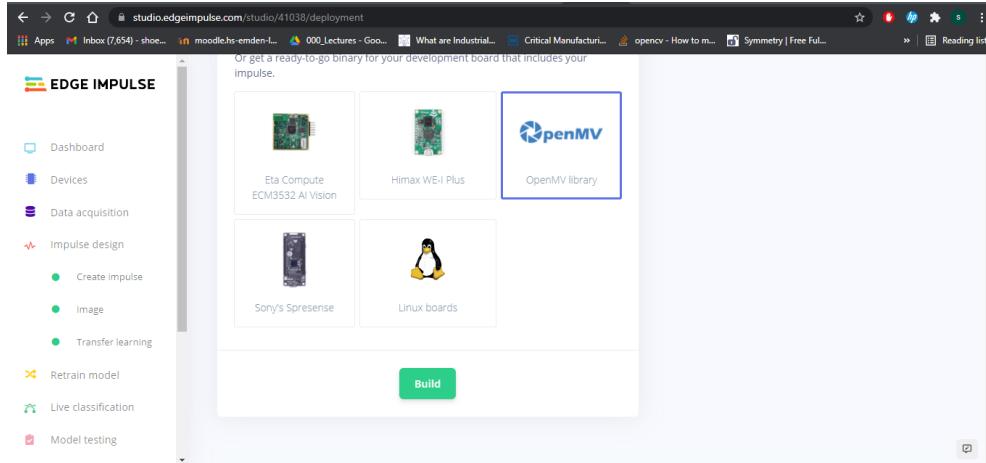


Figure 24.11.: Deployment to the Arduino Portenta H7

24.3.9. Testing the trained model using Vision shield and OpenMV IDE

To test the trained model, we go to the OpenMV IDE and open the downloaded python script `ei_image_classification.py` and click on play button. Now move the vision shield towards the objects and open the serial terminal to see the output predictions. Here we can see from the figure 24.12 that the face is being detected and the prediction is 1.0 which means it is 100%

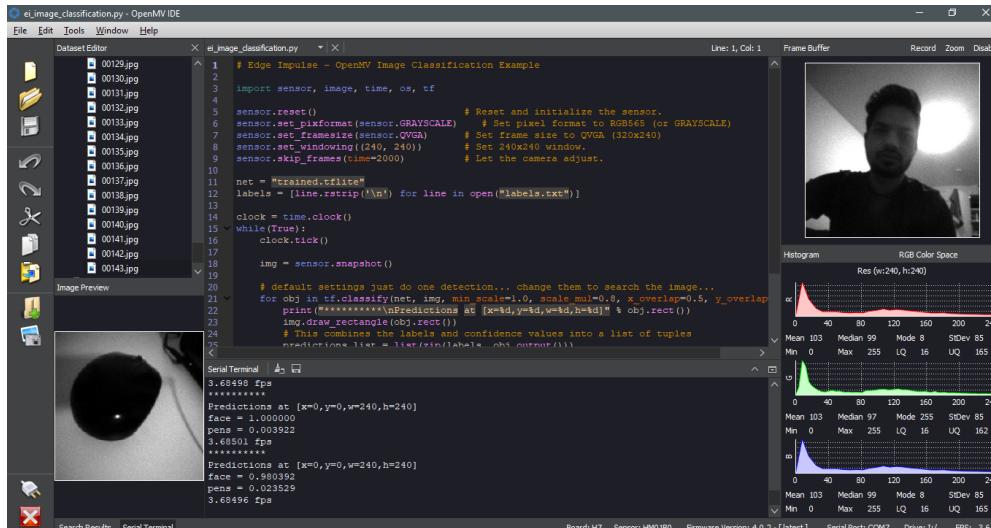


Figure 24.12.: Face detection output

In order to check whether the model differentiates between a face and other objects, we move the vision shield camera towards a pen and we can see from the figure 24.13 that the face prediction is 0 and it predicts that the object is a pen at 0.96 which means 96%. Therefore, we can say that the model is trained to be able to detect faces in an image or video stream captured by the Vision Shield.

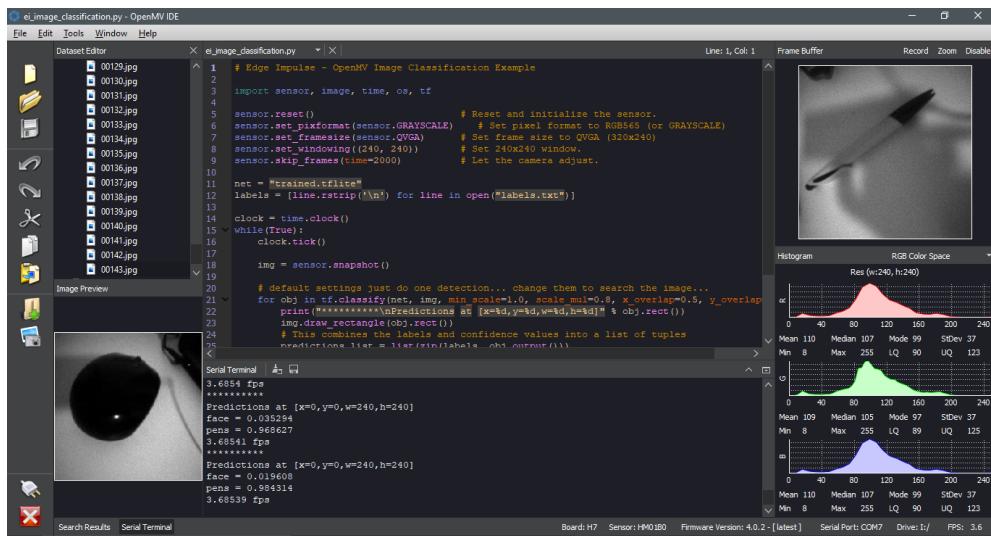


Figure 24.13.: Pen detection

24.4. Face Detection using Edge Impulse

In this model we will be detecting faces in an image without using the OpenMV IDE. We will be making use of Edge Impulse platform for training and testing the model. The flowchart for the overall process is shown in the below figure 24.14

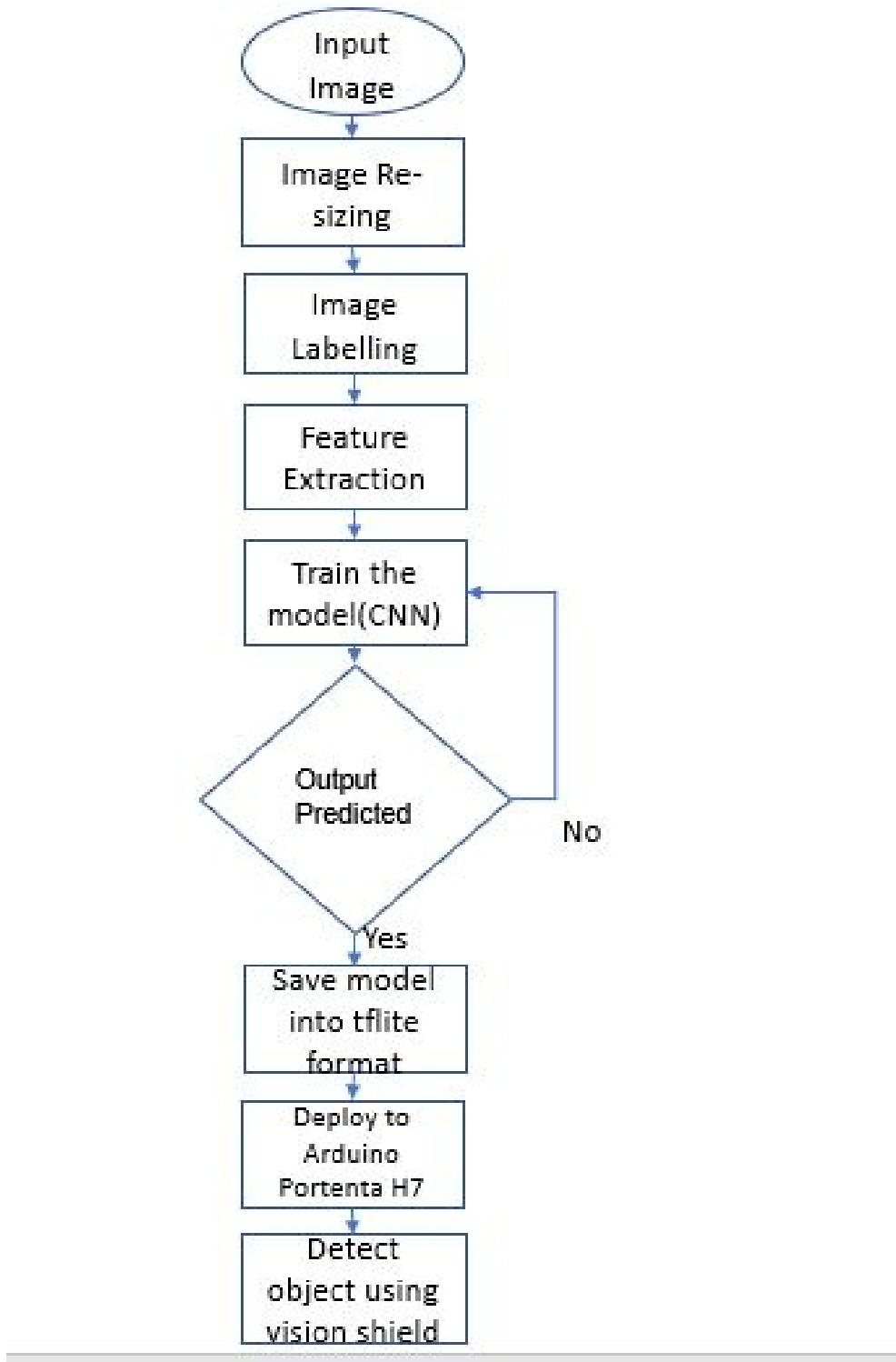


Figure 24.14.: Flowchart of the overall process

24.4.1. Database

To train a machine learning model for face detection we first need to find a database of images consisting of faces. We can create it by capturing the images using the

Vision Shield camera. We can also download a dataset online and use it to train the model. UTKFace is one dataset with long age span(0 to 116 years old) consisting of over 20k single face images. We can download this dataset from the link <https://susanqq.github.io/UTKFace/>

Selection of Dataset : Manually captured images from the camera present on the Vision shield, Images taken from UTKFace dataset. Training set : Consisting of 388 images testing set : consisting of 98 images In the below figure 24.15 we can see the sample images from UTKFace dataset. The dimension of the images in this dataset is 200x200 pixels.

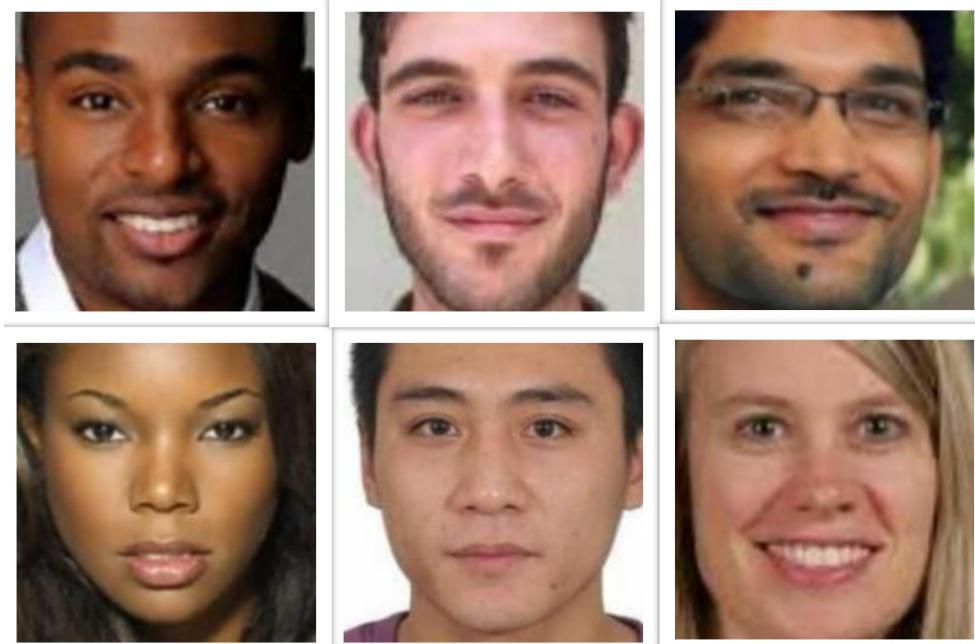


Figure 24.15.: Sample Images from UTKFace Dataset

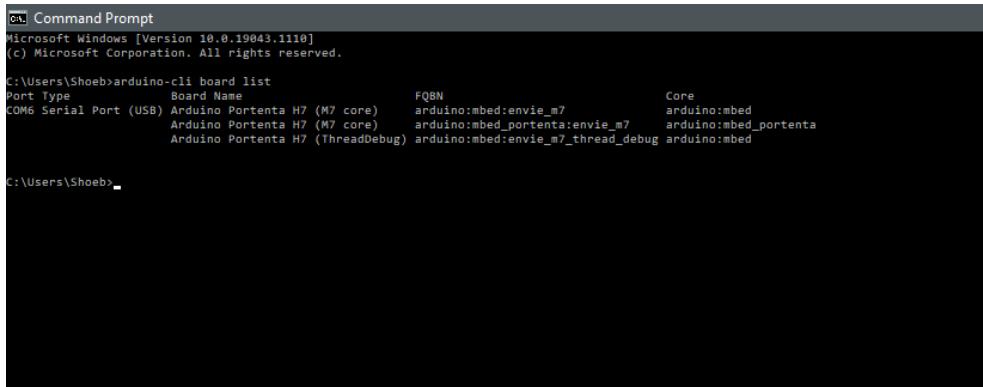
Uploading the dataset in the Edge Impulse : Firstly, we need to login in the Edge Impulse portal and create a project named Face_detection. In order to upload our image dataset , we need to go to the Edge Impulse dashboard and click on acquire data and select upload data. For uploading the dataset of images captured by Arduino Vision Shield, we need to connect it to the Edge Impulse Platform. This is described in the next section.

24.4.2. Connecting the Arduino Portenta H7 with Edge Impulse

In order to take pictures from the vision shield, we need to connect our device to the Edge Impulse portal and then start taking pictures. To do this, we need to install the following software.

- **Arduino CLI :** We need to download the Arduino CLI from the website <https://arduino.github.io/arduino-cli/latest/installation/> and extract the downloaded Zip file which is arduino-cli_0.18.3_Windows_64bit and copy the path of the file which is C:\Users\Shoeb\Downloads\arduino-cli_0.18.3_Windows_64bit.zip\. Then we need to add this path in the Environment Variables list on our computer. After adding the path, we need to open command prompt and type Arduino-cli and press enter to install it. After the installation is done, type arduino-cli board list in the command prompt. It displays the

device connected as Arduino Portenta H7 with port number and type as shown in the figure 24.16.



```
Command Prompt
Microsoft Windows [Version 10.0.19043.1118]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shoeb>arduino-cli board list
Port Type          Board Name      FQBN                  Core
COM6 Serial Port (USB) Arduino Portenta H7 (M7 core)  arduino:mbed:envie_m7    arduino:mbed
                                         Arduino Portenta H7 (M7 core)  arduino:mbed_portenta:envie_m7  arduino:mbed_portenta
                                         Arduino Portenta H7 (ThreadDebug) arduino:mbed:envie_m7_thread_debug arduino:mbed

C:\Users\Shoeb>
```

Figure 24.16.: Arduino-CLI Installation

- **Edge Impulse CLI** : We first need to install Python 3 and download Node.js version 14.17 or higher from the website <https://nodejs.org/en/>. The downloaded Windows installer file is named as node-v14.17.4-x64.msi. Then we need to begin installing by opening this Windows installer file. Click on install as shown in the figure.. and the installed path is C:\ProgramData\Microsoft\Windows\StartMenu\Programs\Node.js After the installation is done we need to open the command prompt and type npm install -g edge-impulse-cli –force to install it.

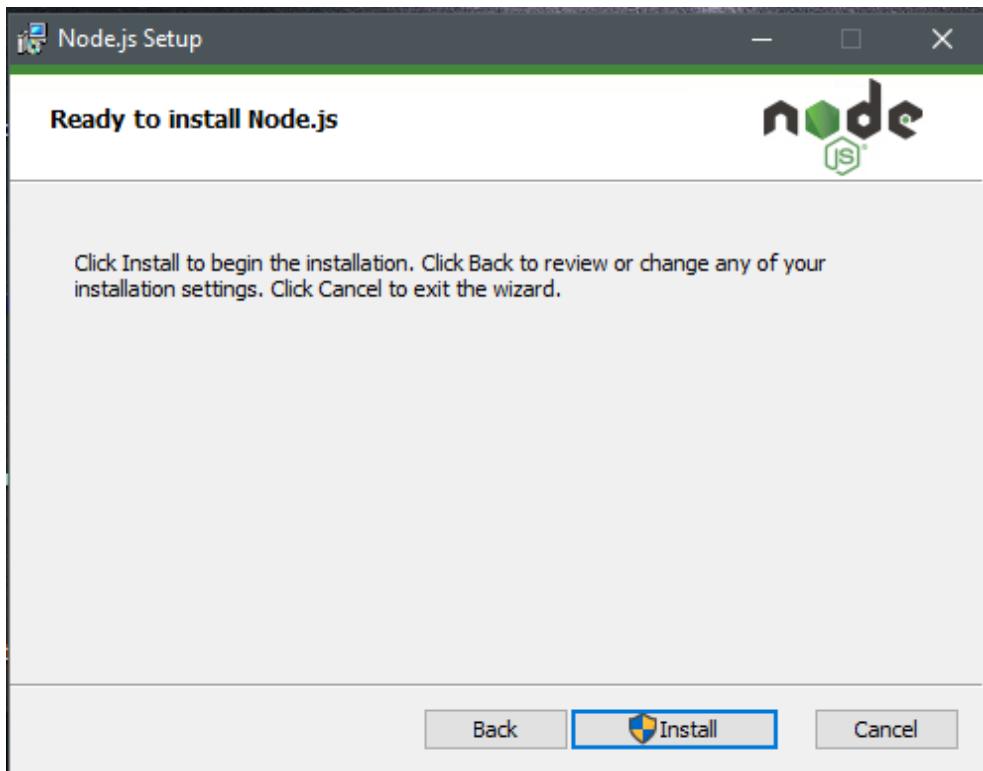


Figure 24.17.: Node.js Installation

After installing these two softwares, we need to connect the Arduino Portenta H7 with Edge Impulse. To do this, we need to download the latest Edge Impulse firmware from the website <https://docs.edgeimpulse.com/docs/arduino-portenta-h7>. The downloaded file is named as arduino-portenta-h7-preview.zip . We ned to extract this file and open the flash script for Windows which is present inside the folder named as flash_windows.bat. After the flashing is done, we need to press the reset button once to launch the new firmware. Now we need to open the command prompt and type edge-impulse-daemon. This prompts us to login with our Edge Impulse ID and Password in the command prompt. After the login is successful,we need to name the device, here i have named it as Arduino Portenta H7 as shown in the figure 24.18.

```

C:\ Command Prompt - edge-impulse-daemon
Microsoft Windows [Version 10.0.19043.1110]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shoeb>arduino-cli board list
Port Type          Board Name           FQBN                Core
COM6 Serial Port (USB) Arduino Portenta H7 (M7 core)    arduino:mbed:envie_m7      arduino:mbed
                                         Arduino Portenta H7 (M7 core)    arduino:mbed:portenta:envie_m7  arduino:mbed_portenta
                                         Arduino Portenta H7 (ThreadDebug) arduino:mbed:envie_m7_thread_debug arduino:mbed

C:\Users\Shoeb>edge-impulse daemon
'edge-impulse' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Shoeb>edge-impulse-daemon
Edge Impulse serial daemon v1.13.14
Endpoints:
  WebSocket: wss://remote-mgmt.edgeimpulse.com
  API:      https://studio.edgeimpulse.com/v1
  Ingestion: https://ingestion.edgeimpulse.com

[SER] Connecting to COM6
[SER] Serial is connected, trying to read config...
[SER] Retrieved configuration
[SER] Device is running AT command version 1.5.0

Setting upload host in device... OK
Configuring remote management settings... OK
Configuring API key in device... OK
Configuring HMAC key in device... OK
[SER] Device is not connected to remote management API, will use daemon
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com
[?] What name do you want to give this device? (4E:D3:0D:36:40:04) [SER] Entering snapshot stream mode...
[?] What name do you want to give this device? Arduino Portenta h7
[WS ] Device "Arduino Portenta h7" is now connected to project "Face_Detection"
[WS ] Go to https://studio.edgeimpulse.com/studio/41038/acquisition/training to build your machine learning model!

```

Figure 24.18.: Connecting Arduino Portenta H7 with Edge Impulse

To verify that the board is connected, go to the Edge Impulse website and click on devices,we can see that our device is connected as shown in the figure 24.19. This way we have successfully connected our board with the Edge Impulse platform and now we can go for training the model.

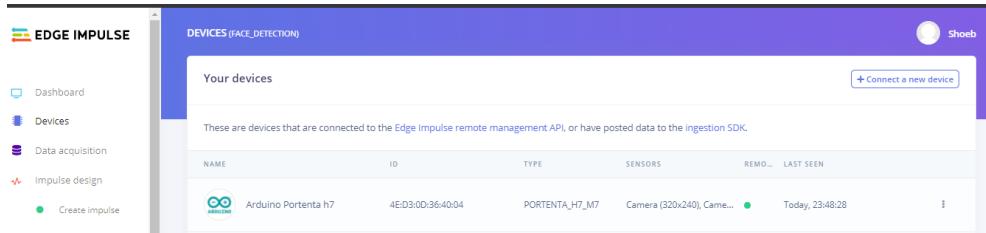


Figure 24.19.: Connected Devices in the Edge Impulse

24.4.3. Data Labelling

Now that our data is uploaded, we need to label the data. To do this, we go to Labeling queue option in Data Acquisition and start drawing bounding box around the object and label it. In our database we have two different objects which are faces

and pens. So we manually draw bounding box around the object and label them for all the images in the dataset.

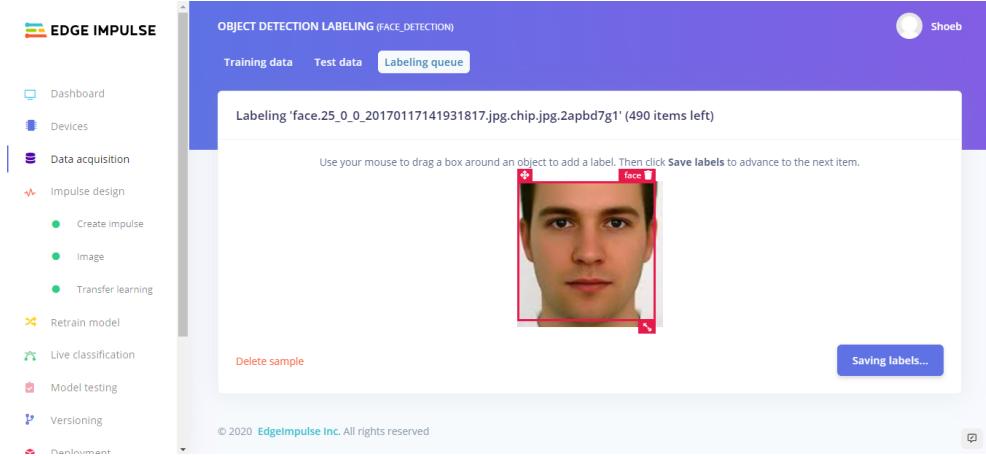


Figure 24.20.: Labelling an Image

24.5. Data transformation

24.5.1. Designing the Impulse

Image Preprocessing :

For designing the impulse, click on Impulse design and then set the image width and height to 320X320 because the object detection pre-trained model works only with images of size 320X320 [Tea21b]. Then in the processing block we click on Image. The image pre-processing block takes input image and can optionally convert it into grayscale and then turns the data into a features array. We then click on learning block and select object detection. Then click on save impulse as shown in the figure 24.21.

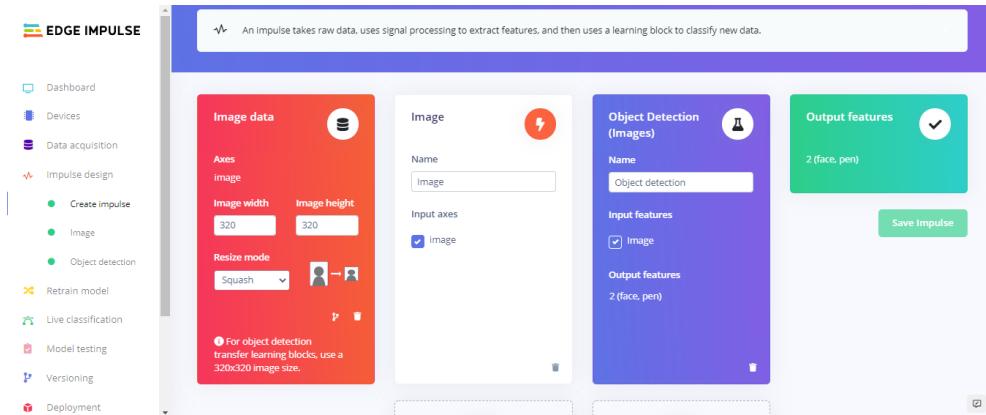


Figure 24.21.: Create Impulse

24.5.2. Configuring the Image processing block

To configure the processing block, click Images in the menu on the left. This will show us the raw data on top of the screen and the results of the processing step on the right. we can also use the options to switch between 'RGB' and 'Grayscale' mode. We then click on save parameters as shown in figure 24.22.

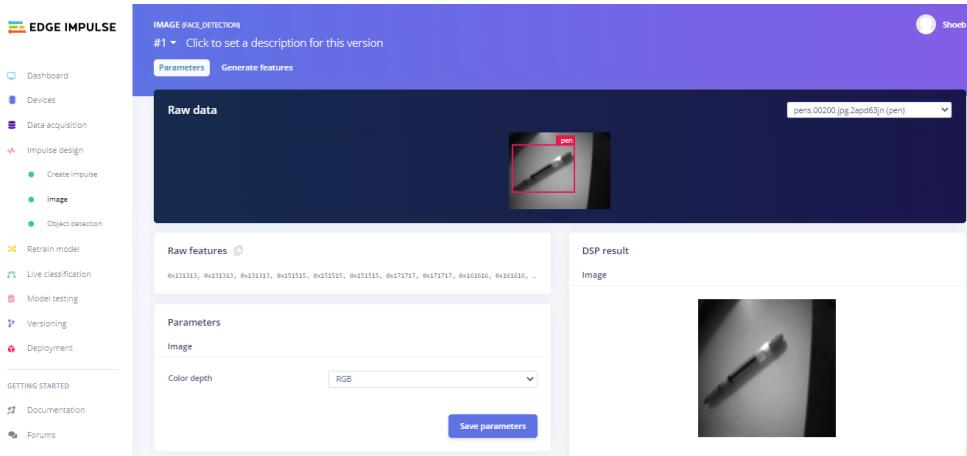


Figure 24.22.: Save parameters

Then click on generate features option on the top and then click on generate features. In the figure 24.23 we can see that features generated from the dataset of images consisting of pens and faces are easily distinguishable.

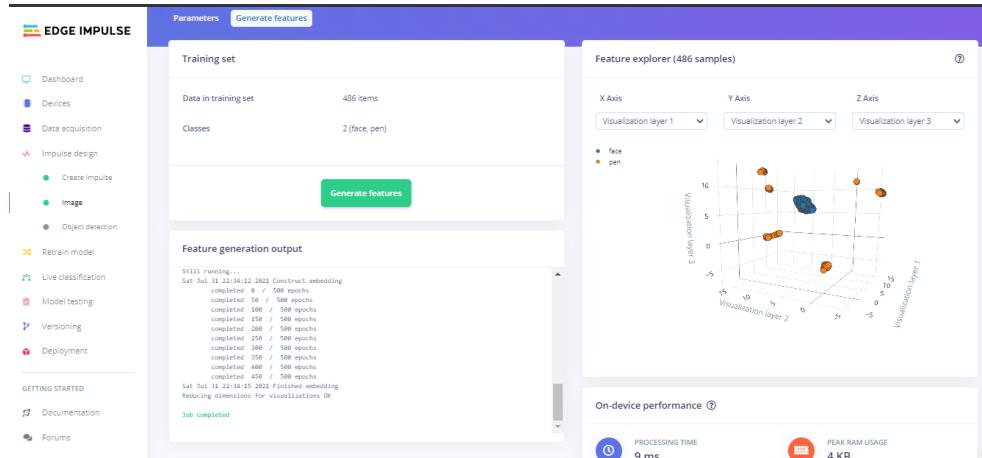


Figure 24.23.: Generate Features

24.6. Data training

Configuring the transfer learning model :

Here we are using transfer learning block because it is easier to use a pre-trained model by only retaining the upper layers of a neural network and then we can train the model in a short time. To configure the transfer learning model, click on object detection. Here, we can set the learning rate ,in our case we have set it to 0.15 and the number of training cycles to 25. Then we select the base model, here we have selected as MobileNetV2 and then click on start training. After the training process is completed we can see from the figure 24.24 the precision of it is 87%.

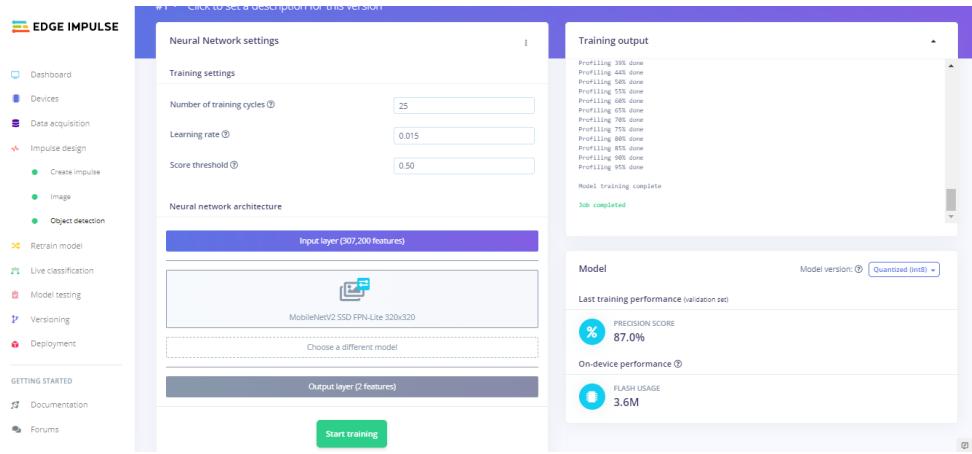


Figure 24.24.: Completed training

24.7. Testing the model

In order to test the model, we need to click on the option model testing and then click on classify all. Here, the dataset images were automatically divided into training and testing images. Here, the model divided the input images into training images consisting of 388 images and testing consisting of 98 images. The testing images are then used to check how well the model performs. As we can see from the figure 24.25 that the model achieved 87.1% accuracy.

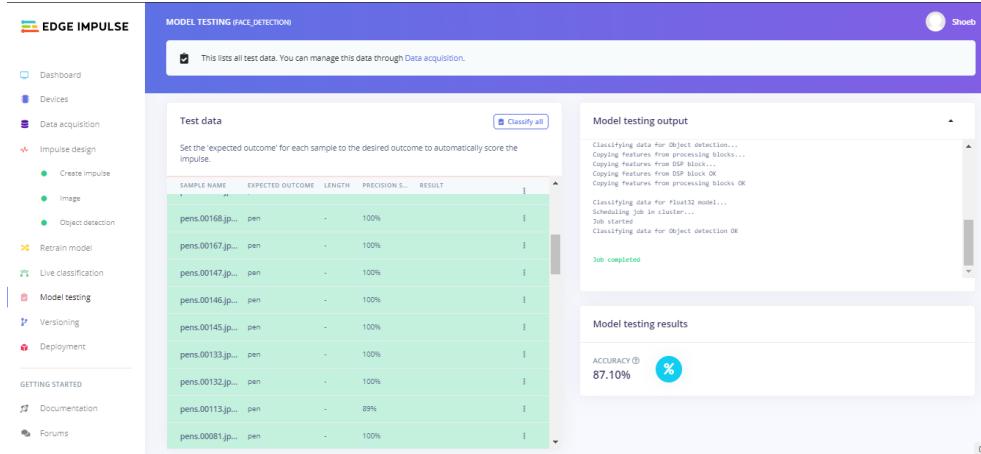


Figure 24.25.: Testing the model

We can also do the live classification and detect objects in real time using the vision shield. Click on the live classification option in the menu and connect our Portenta H7 device with the Edge Impulse by selecting the device. Then as soon as it is connected we can see the camera stream from vision shield. Here we have tested the model by focusing on face and as we can see from the figure 24.26 that it gives the output as face with 0.89 value which is a pretty good prediction.

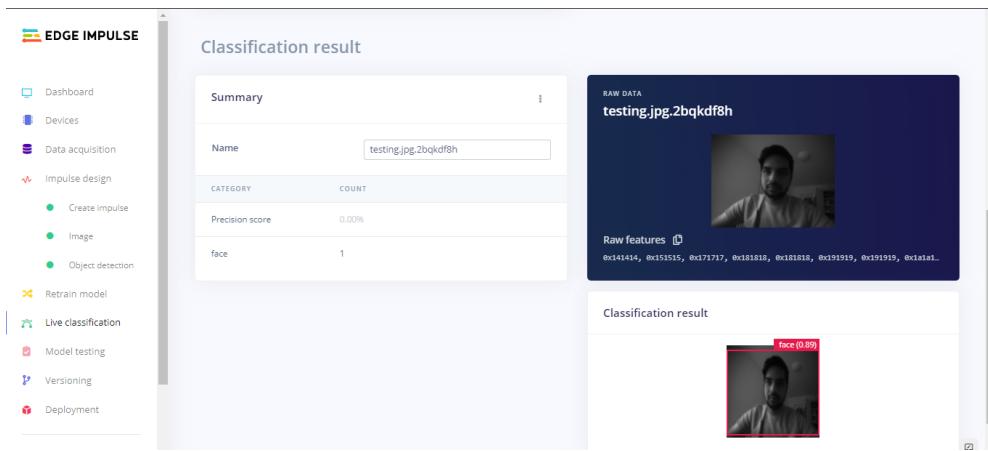


Figure 24.26.: Live classification by using vision shield camera

25. Use TensorFlow Lite with Portenta H7

25.1. Introduction

In this chapter, we will be using Google Colab to train an object detection model and then convert it to TensorFlow Lite format in order to upload and validate it on Arduino Portenta H7. TensorFlow Lite for microcontrollers is designed to run machine learning models with limited memory. TensorFlow Lite for microcontrollers is written in C++ 11 and needs a 32-bit platform. Arduino Portenta H7 board is not yet supported to run TensorFlow Lite models using Arduino IDE [Goo19b], but it is worth giving a try.

25.2. Training a Face detection model using Google Colaboratory

25.2.1. Google Colaboratory

Google Colaboratory or Colab is a product from Google Research and is a free Jupyter notebook environment running completely in the cloud. It is typically well suited to machine learning and data analysis. The good thing is it requires no set up and provides free access to computing resources like GPU and TPU [Goo21].

Advantages of Colab

- **Pre installed libraries:** It provides pre-installed machine learning libraries including PyTorch, Keras, TensorFlow
- **Saved on the Cloud :** If we use a Jupyter notebook, then everything is saved on a local device. However, when we use Google Colab, we can save it on the cloud and can access from any device by just signing in Google Drive account.
- **Collaboration :** It helps in providing access to multiple developers working on the project to edit the code or code together and also share the completed code with the developers
- **Free GPU and TPU use :** This is probably the best feature as Google Research lets us use their dedicated GPUs and TPUs for our personal machine learning projects. Therefore, our computers don't have to undergo the complex computations while training the model because it is done on the cloud

25.3. Database

The Database for faces is the same as in the previous chapter which is UTKFace Dataset. We will be training the model based on Image classification technique so we will be using another class that is Flowers in our case and the database for it can be downloaded from Google. **Selection of Dataset :** Images taken from Flowers dataset :100 images consisting of Roses saved in a folder named as Roses. The dimensions of the images in the dataset varies but we select images of dimensions in the range of 224 x 224 pixels.



Figure 25.1.: Sample Images from flowers dataset

Images taken from UTKFace dataset : 120 images consisting of Faces in a folder saved as Faces. The dimensions of the images is 200X200 pixels.

25.3.1. Connecting to Google Drive

In order to create and save notebook in Colab, we need to connect it to our Google Drive. Also, we need to upload our images dataset in our Google Drive in order to access it from Colab. To do this, we use the function `drive.mount('/content/gdrive')`. This will redirect to our google sign-in page and gives us the authorization key required to connect our Google Drive to Colab. After entering the key we will be connected to our Google Drive.

25.3.2. Setting up the Dataset

This section is a significant part of where we have placed our image zipped file and this .ipynb script file(base_path) on our Google Drive, how we have named our zipped file(imagezipped) which in our case is 'images_RF.zip'. and the name of the folder(data_dir) which is data_dir='images' where we compiled our images before compressing it.

```

[1] #set variables
data_dir = 'images'
base_path = '/mydrive/train'
imagezipped = 'images_RF.zip'
epochs = 10

[6] #first of all, we need to delete any images or saved model from the previous run, if any.
os.system("rm -rf {data_dir}".format(data_dir=data_dir))
# !rm -rf images
!rm -rf /tmp/saved_model

[8] #create output path if not yet existing
os.system("mkdir {base_path}/output".format(base_path=base_path))

```

Figure 25.2.: Setting up the dataset

25.4. Data Preprocessing

Rescaling :

In the preprocessing part we will be first rescaling the image using the rescaling function `rescale=1./255` because our original images consist in RGB coefficients in the 0-255, but such values would be too high for our model to process (given a typical learning rate), so we target values between 0 and 1 instead by scaling with a 1/255 factor.

Data Augmentation :

In the data augmentation part, we will be using image augmentation techniques such as rotation of images and horizontal flipping of images. Here, the rotation range of images is given as 40 degrees as shown in figure 25.3.

```

datagen_kwargs = dict(rescale=1./255, validation_split=.20)
dataflow_kwargs = dict(target_size=IMAGE_SIZE, batch_size=BATCH_SIZE,
                      interpolation="bilinear")

valid_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    **datagen_kwargs)
valid_generator = valid_datagen.flow_from_directory(
    data_dir, subset="validation", shuffle=False, **dataflow_kwargs)

do_data_augmentation = True #@param {type:"boolean"}
if do_data_augmentation:
    train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
        rotation_range=40,
        horizontal_flip=True,
        width_shift_range=0.2, height_shift_range=0.2,
        shear_range=0.2, zoom_range=0.2,
        **datagen_kwargs)
else:
    train_datagen = valid_datagen
train_generator = train_datagen.flow_from_directory(
    data_dir, subset="training", shuffle=True, **dataflow_kwargs)

```

Figure 25.3.: Rescaling and Data Augmentation

25.5. Training the model

In this we have used a pre-trained image classification model based on MobileNet V2(224x224). The input image size should be in 224x224 pixels dimension. The model here is build with keras sequential layers and the drop out rate given here is 0.2. The batch-size is 32 and the optimizer used in this model is Stochastic gradient descent(SGD) with a learning rate of 0.005. The loss function used here is Categorical cross entropy. we have set the epoch value to 10.Epoch is the number of iteration for back propagation to correct the errors on earlier part so that it would increase accuracy. The larger the epoch the more iteration thus the longer the training process.

```

model.compile(
    optimizer=tf.keras.optimizers.SGD(lr=0.005, momentum=0.9),
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True, label_smoothing=0.1),
    metrics=['accuracy'])

steps_per_epoch = train_generator.samples // train_generator.batch_size
validation_steps = valid_generator.samples // valid_generator.batch_size
hist = model.fit(
    train_generator,
    epochs=epochs, steps_per_epoch=steps_per_epoch,
    validation_data=valid_generator,
    validation_steps=validation_steps).history

```

Figure 25.4.: Training the model

25.6. Testing the model

Here, we will be testing the model on an image from validation dataset. As can be seen from the figure 25.5 , the model predicts the output which is face in the image accurately.

```
[ ] # Expand the validation image to (1, 224, 224, 3) before predicting the label
prediction_scores = model.predict(np.expand_dims(image, axis=0))
predicted_index = np.argmax(prediction_scores)
print("True label: " + get_class_string_from_index(true_index))
print("Predicted label: " + get_class_string_from_index(predicted_index))
```



True label: Faces
Predicted label: Faces

Figure 25.5.: Testing the model

The trained model can then be stored on session storage.

25.7. Converting the saved model to tflite format

To convert the saved model in to tflite format we use the following function `converter = tf.lite.TFLiteConverter.from_saved_model(saved_model)`

After converting the model into tflite we save the tflite file on our drive. The next step is to convert the tflite file to C++ format so that we can upload the program in the Arduino IDE.

25.8. Converting the tflite file to Arduino header file

In order to run the object detection model in Arduino IDE we need to program our model in C++ format and also add the arduino header file created from our tflite model. To create a arduino header file, we use Gitpod to convert tflite file to a header(.h file) file. TO do this, firstly we need to login to GitPod using our GitHub credentials. Then we need to create a new workspace. In the command terminal we need to use this command `xxd -i model.tflite model.h`. This will generate a header file we need in our C++ program to upload in the Arduino IDE. After downloading the header file, we need to copy the contents of the file and paste it in our program with the command `const unsigned char model_tflite[] =`. This represents the features of our face detection model. Then we need to implement the face detection function in C++ code.

25.9. Open Questions

25.9.1. Arduino Portenta H7 board not compatible for object detection using Arduino IDE

The problem occurred during the implementation of face detection model using Arduino IDE is that there is no previous work done on object detection using Arduino Portenta H7 board as it was not comptabile[Goo19b].Therefore, there was no data available which could be taken as a reference for our face detection model. Also to implement a TensorFlowLite model in the Arduino IDE is a herculean task as our board is not fast enough and the time to compile a program is more than 20 minutes and there are no libraries available to implement the object detection model. However, there is only one example available online which is the Hello World example of Arduino also called as a sine wave function in a tflite format which can be executed using the Arduino IDE but we cannot take this as a reference because object detection model and a sine wave model are completely different.

Possible Solution

The possible solution could be using OpenMV IDE instead of Arduino IDE. OpenMV IDE has more number of functions when compared to Arduino IDE such as uploading dataset real-time by capturing images from vision shield. It also has many more example programs like face tracking, face filter. So, using OpenMV IDE could be a better and easy solution for this issue.

25.9.2. Unable to Deploy object detection model trained on Edge Impulse using Arduino IDE

The object detection model which was trained for face detection in edge impulse platform and the tflite file generated online was unable to deploy on Arduino IDE. The model works completely fine on the platform, however during deployment on Arduino Portenta H7 board using Arduino IDE received various errors. Upon consulting the edge impulse platform creators we got to know that they are not yet ready in deployment of the model generated from their platform using arduino IDE.

Possible Solution

The model generated from Edge Impulse platform is coded in python programming language. For Arduino IDE, the programming language is C++. So, as of now Edge Impulse does have some issues when it comes to Arduino IDE as some changes has to be done. So, the better soution would be using OpenMV IDE, as it supports MicroPython and the Edge Impulse model generates a python file which we can upload in OpenMV IDE and implement our face detection model.

26. Source Code

26.1. Serial Blink Example for Arduino Portenta H7

```
void setup() {  
Serial.begin(115200); // Sets the data rate in bits per second (baud) for serial data transmission  
pinMode(LED_BUILTIN, OUTPUT); // LEDB = blue, LEDG or LED_BUILTIN = green, LEDR = red (typically for error)  
#ifdef CORE_CM7 // shows how to specify a specific core  
bootM4();  
#endif  
}  
  
void loop() {  
Serial.println("Serial print works on the M7 core, just ignored on the M4 Core, unless you use RP  
digitalWrite(LED_BUILTIN, LOW); // Portenta onboard LED connected to 3V3 so ground it to light  
delay(1000); // wait for a second  
digitalWrite(LED_BUILTIN, HIGH); // turn the LED off by not grounding it, weird eh.  
delay(4000); // wait for 3 seconds  
}
```

26.2. Example Script for blinking builtin RGB LED on the Arduino Portenta H7 with OpenMV

```
import pyb  
redLED = pyb.LED(1) # built-in red LED  
greenLED = pyb.LED(2) # built-in green LED  
blueLED = pyb.LED(3) # built-in blue LED  
while True:  
# Turns on the red LED  
redLED.on()  
# Makes the script wait for 1 second (1000 milliseconds)  
pyb.delay(1000)  
# Turns off the red LED  
redLED.off()  
pyb.delay(1000)  
greenLED.on()  
pyb.delay(1000)  
greenLED.off()  
pyb.delay(1000)  
blueLED.on()  
pyb.delay(1000)  
blueLED.off()  
pyb.delay(1000)
```

26.3. QR Code Information Extraction Program

```
import sensor, image, time
```

```
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(time = 2000)
sensor.set_auto_gain(False) # must turn this off to prevent image washout...
clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot()
    img.lens_corr(1.8) # strength of 1.8 is good for the 2.8mm lens.
    for code in img.find_qrcodes():
        img.draw_rectangle(code.rect(), color = (255, 0, 0))
    print(code)
    print(clock.fps())
```

26.4. Finding Circles around the Objects

```
import sensor, image, time

sensor.reset()
sensor.set_pixformat(sensor.GRAYSCALE) # grayscale is faster
sensor.set_framesize(sensor.QQVGA)
sensor.skip_frames(time = 2000)
clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot().lens_corr(1.8)

    # Circle objects have four values: x, y, r (radius), and magnitude. The
    # magnitude is the strength of the detection of the circle. Higher is
    # better...

    # 'threshold' controls how many circles are found. Increase its value
    # to decrease the number of circles detected...

    # 'x_margin', 'y_margin', and 'r_margin' control the merging of similar
    # circles in the x, y, and r (radius) directions.

    # r_min, r_max, and r_step control what radii of circles are tested.
    # Shrinking the number of tested circle radii yields a big performance boost.

    for c in img.find_circles(threshold = 2000, x_margin = 10, y_margin = 10, r_margin = 10,
        r_min = 2, r_max = 100, r_step = 2):
        img.draw_circle(c.x(), c.y(), c.r(), color = (255, 0, 0))
    print(c)

    print("FPS %f" % clock.fps())
```

26.5. Creating a basic face filter with OpenMV

```

import sensor # Import the module for sensor related functions
import image # Import module containing machine vision algorithms
import time # Import module for tracking elapsed time

sensor.reset() # Resets the sensor
sensor.set_contrast(3) # Sets the contrast to the highest level (min - 3, max 3)
sensor.set_gainceiling(16) # Sets the amplification of camera sensor signal

# HQVGA and GRAYSCALE are the best for face tracking.
sensor.set_framesize(sensor.HQVGA)
sensor.set_pixformat(sensor.GRAYSCALE)

# Load the built-in frontal face Haar Cascade
# By default this will use all stages, lower stages is faster but less accurate.
face_cascade = image.HaarCascade("frontalface", stages=25)
print(face_cascade) # Prints the Haar Cascade configuration

faceImage = image.Image("/face.pbm", copy_to_fb=False) # Loads a bitmap file from the flash storage
clock = time.clock() # Instantiates a clock object to calculate the FPS

while (True):
    clock.tick() # Advances the clock
    cameraImage = sensor.snapshot() # Takes a snapshot and saves it in memory

    # Find objects.
    # Note: Lower scale factor scales-down the image more and detects smaller objects.
    # Higher threshold results in a higher detection rate, with more false positives.
    boundingBoxes = cameraImage.find_features(face_cascade, threshold=1, scale_factor=1.5)

    # Draw objects
    for boundingBox in boundingBoxes:
        faceX = boundingBox[0]
        faceY = boundingBox[1]
        faceWidth = boundingBox[2]

        # Calculates the scale ratio to scale the bitmap image to match the bounding box
        scale_ratio = faceWidth / faceImage.width()
        # Draws the bitmap on top of the camera stream
        cameraImage.draw_image(faceImage, faceX, faceY, x_scale=scale_ratio, y_scale=scale_ratio)

    # Print FPS.
    # Note: The actual FPS is higher when not displaying a frame buffer preview in the IDE
    print(clock.fps())

```

26.6. Face Tracking

```

import sensor, time, image

# Reset sensor

```

```
sensor.reset()
sensor.set_contrast(3)
sensor.set_gainceiling(16)
sensor.set_framesize(sensor.QVGA)
sensor.set_windowing((320, 240))
sensor.set_pixformat(sensor.GRAYSCALE)

# Skip a few frames to allow the sensor settle down
sensor.skip_frames(time = 2000)

# Load Haar Cascade
# By default this will use all stages, lower stages is faster but less accurate.
face_cascade = image.HaarCascade("frontalface", stages=25)
print(face_cascade)

# First set of keypoints
kpts1 = None

# Find a face!
while (kpts1 == None):
    img = sensor.snapshot()
    img.draw_string(0, 0, "Looking for a face...")
    # Find faces
    objects = img.find_features(face_cascade, threshold=0.5, scale=1.25)
    if objects:
        # Expand the ROI by 31 pixels in every direction
        face = (objects[0][0]-31, objects[0][1]-31, objects[0][2]+31*2, objects[0][3]+31*2)
        # Extract keypoints using the detect face size as the ROI
        kpts1 = img.find_keypoints(threshold=10, scale_factor=1.1, max_keypoints=100, roi=face)
        # Draw a rectangle around the first face
        img.draw_rectangle(objects[0])

    # Draw keypoints
    print(kpts1)
    img.draw_keypoints(kpts1, size=24)
    img = sensor.snapshot()
    time.sleep_ms(2000)

    # FPS clock
    clock = time.clock()

while (True):
    clock.tick()
    img = sensor.snapshot()
    # Extract keypoints from the whole frame
    kpts2 = img.find_keypoints(threshold=10, scale_factor=1.1, max_keypoints=100, normalized=True)

    if (kpts2):
        # Match the first set of keypoints with the second one
        c=image.match_descriptor(kpts1, kpts2, threshold=85)
        match = c[6] # C[6] contains the number of matches.
        if (match>5):
            img.draw_rectangle(c[2:6])
            img.draw_cross(c[0], c[1], size=10)
```

```
print(kpts2, "matched:%d dt:%d"%(match, c[7]))  
  
# Draw FPS  
img.draw_string(0, 0, "FPS:%.2f"%(clock.fps()))
```

26.7. Face Detection using Edge Impulse and OpenMV

```
import sensor, image, time, os, tf  
  
sensor.reset()                      # Reset and initialize the sensor.  
sensor.set_pixformat(sensor.GRAYSCALE) # Set pixel format to RGB565 (or GRayscale)  
sensor.set_framesize(sensor.QVGA)      # Set frame size to QVGA (320x240)  
sensor.set_windowing((240, 240))       # Set 240x240 window.  
sensor.skip_frames(time=2000)          # Let the camera adjust.  
  
net = "trained.tflite"  
labels = [line.rstrip('\n') for line in open("labels.txt")]  
  
clock = time.clock()  
while(True):  
    clock.tick()  
  
    img = sensor.snapshot()  
  
    # default settings just do one detection... change them to search the image...  
    for obj in tf.classify(net, img, min_scale=1.0, scale_mul=0.8, x_overlap=0.5, y_overlap=0.5):  
        print("*****\nPredictions at [x=%d,y=%d,w=%d,h=%d]" % obj.rect())  
        img.draw_rectangle(obj.rect())  
    # This combines the labels and confidence values into a list of tuples  
    predictions_list = list(zip(labels, obj.output()))  
  
    for i in range(len(predictions_list)):  
        print("%s = %f" % (predictions_list[i][0], predictions_list[i][1]))  
  
    print(clock.fps(), "fps")  
  
.
```

26.7.1. Beispiel für serielles Blinken bei Dual-Core-Verarbeitung

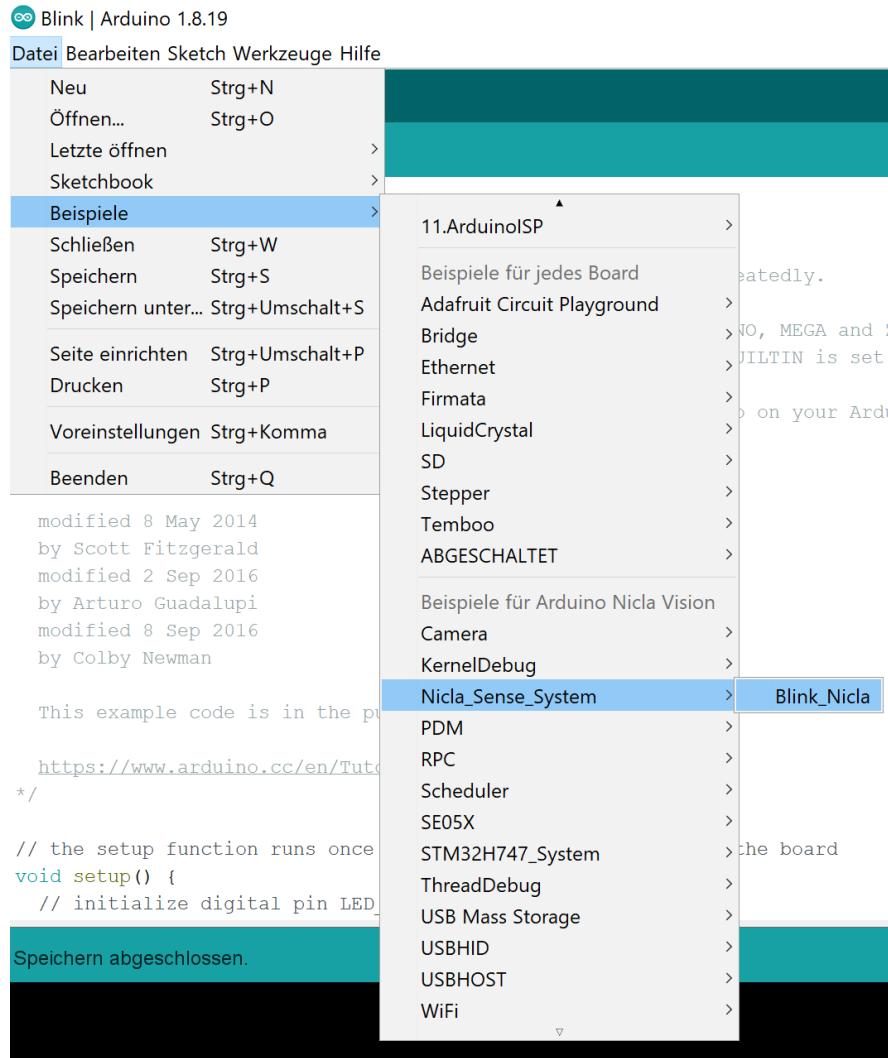


Figure 26.1.: Optionen der Menüleiste

Die Abbildung 26.2 stellt den seriellen Blink-Sketch dar. Dieser Sketch hilft bei der Demonstration der seriellen Datenübertragung, indem er die LED nach einer bestimmten Zeitspanne blinken lässt.

Hier wird mit dem Befehl `Serial.begin(115200)` die Datenübertragungsrate auf 115200 Bits pro Sekunde eingestellt. Die `LEDB` ist der Ausgang, der blau blinkt, und wir haben den Kern M7 angegeben. Wenn wir den Kern M4 verwenden möchten, müssen wir ihn zuerst mit dem Befehl `bootM4()` booten. Die Ausgangs-LED blinkt nach einer Verzögerung von 3 Sekunden blau, wie in der Skizze definiert.

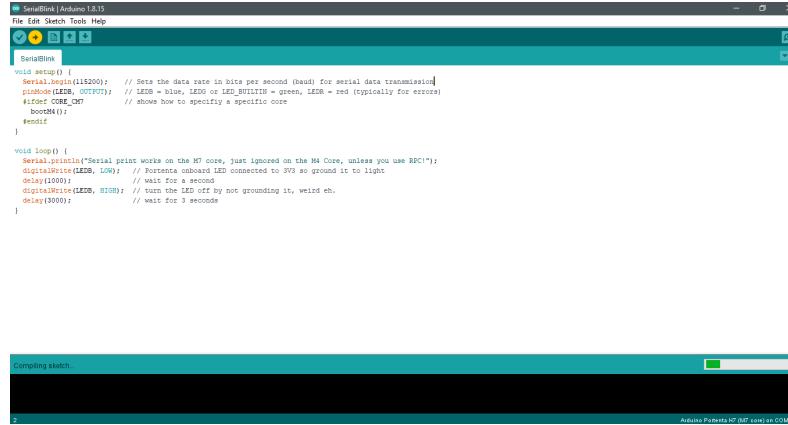


Figure 26.2.: Serieller Blink-Sketch I

Um den Kern M4 zu verwenden, müssen wir den Kern M7 von der Karte abwählen und den Kern M4 auswählen. Nach der Auswahl können wir unten den ausgewählten Kern zur Bestätigung sehen. Außerdem müssen wir einen weiteren Sketch schreiben und nun die LED-Farbe im Sketch auf rot ändern und eine Verzögerung von 4 Sekunden hinzufügen. 26.3

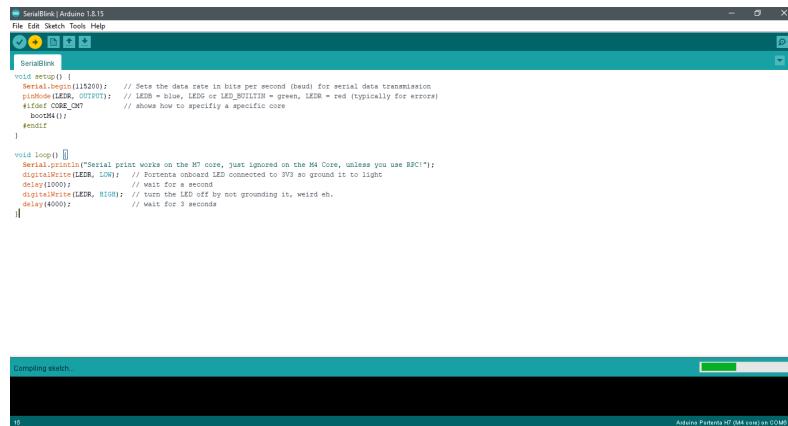


Figure 26.3.: Serieller Blink-Sketch II

Beachten Sie, dass in der Ausgabe sowohl die blaue als auch die grüne LED blinken und zu einem bestimmten Zeitpunkt fast zusammen blinken, was zu einer Lila Farbe führt. Dies zeigt das Dual-Core-Verarbeitungskonzept unter Verwendung einer LED. 26.4

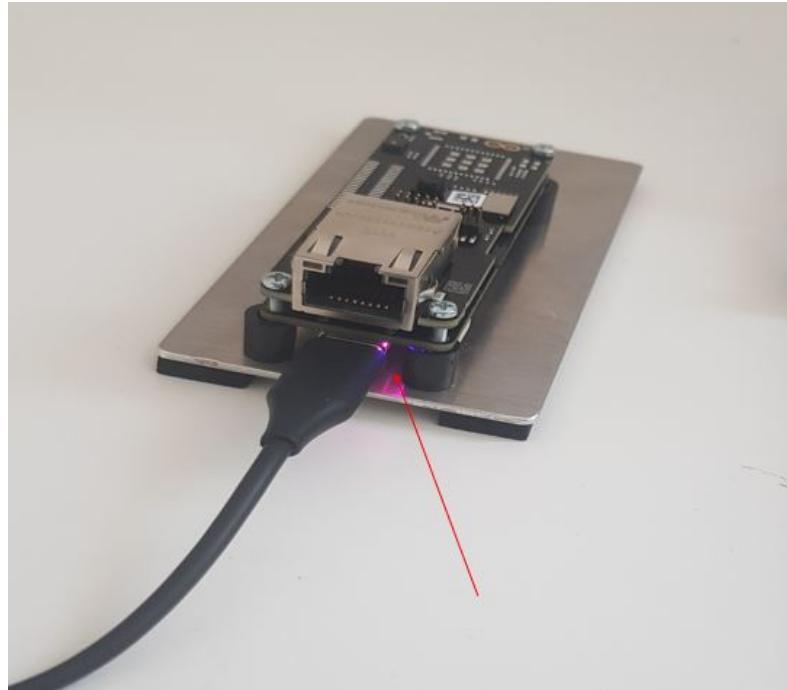


Figure 26.4.: Ausgang LED-Farbe

Aus der Abbildung ist ersichtlich, dass die Ausgangs-LED aufgrund der Überschneidung des gleichzeitigen Blinkens der grünen und der blauen LED in lila Farbe blinkt.

26.8. Arduino Portenta Vision Shield

The vision shield is a hardware add-on to the portenta h7 board in order to perform machine vision applications. It has a camera module with 324X324 pixel resolution which contain ultra-low power image sensor and is highly sensitive in detecting motion, gestures, and object identification. It also has a JTAG connector to perform debugging or getting special firmware for the board. In addition, it also has two on-board microphones for sound detection and analysis in real-time. It is available in two variants : one with Ethernet port for WiFi connectivity and the other with LoRa connectivity.

In order to perform machine vision applications, Arduino has partnered with OpenMV to provide free license for the OpenMV IDE. OpenMV enables us to develop low-cost MicroPython based machine vision applications. It also has MicroSD slot in order to save programs on it.[Ard21b]

Technical Specifications

- Camera: Himax HM-01B0 camera module
- Resolution: 320 x 320 active pixel resolution with support for QVGA
- Image sensor: Highly sensitive 3.6 μ BrightSenseTM pixel technology
- Microphones: 2x MP34DT06JTR
- Connection: LAN
- Interface: JTAG
- Dimensions (LxW): 66 x 25 mm

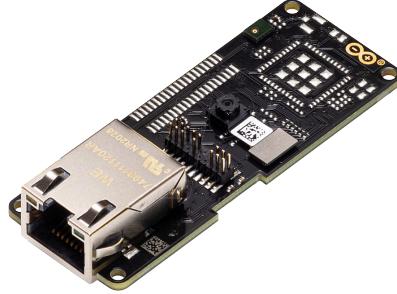


Figure 26.5.: Arduinos Vision Shield für den Portenta H7 Arduino Store

26.8.1. Camera Module

It has a very low power monochrome camera with a maximum of 60 fps and also allows the possibility to perform motion detection without interacting with the main processor. With its “Always on” functionality it switches on the main processor only when it detects motion and therefore enabling minimum power consumption. It captures images of size 324X324 pixels but this image is cropped down to OpenMV standard resolutions 320X240 which is QVGA(Quarter Video graphics Array). As the name suggests, QVGA is the quarter of the standard VGA which is 640X480 pixels.

26.8.2. Microphones

The microphones present in the vision shield are omnidirectional which means they can receive audio signals in all directions. These features can be used in applications such as speech recognition and predictive maintenance of machines based on the abnormal sounds made by them in an industry.

26.8.3. Micro SD card

The vision shield comes with a micro SD card slot to store images and videos captured by it or read configuration files.

26.8.4. Ethernet

The vision shield module with Ethernet connector allows connecting to 10/100 Tx base networks with the help of Ethernet PHY available on the Arduino Portenta H7 board. This enables it connect to the internet using RJ45 cable.

26.8.5. LoRa Module

The vision shield also comes with a LoRa(Long Range) module variant provided by the Murata CMWX1ZZABZ module which contains STM32L0 processor along with a Semtech SX1276 Radio. With the help of this module we can run computer vision applications wirelessly via LoRa to the cloud.

26.8.6. Power

The Arduino Portenta H7 supplies 3.3 Volts power via the output pin(3.3V) to the microSD card slot,dual microphones and LoRa module(ASX00026 only) via the high density conector. An onboard Low-Dropout(LDO) regulator supplies 2.8V output to the camera module.

https://www.reichelt.de/arduino-portenta-shield-vision-mit-lan-ard-shd-asx00021-p292402.html?CCOUNTRY=445&LANGUAGE=de&trstct=pos_3&nbc=1&r=1

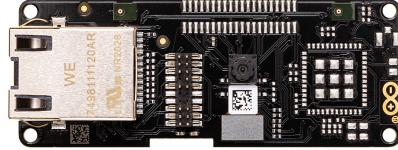


Figure 26.6.: Arduinos Vision Shield für den Portenta H7 Arduino Store

ARD SHD ASX00021

Portenta Vision Shield: eine produktionsreife Lösung für Embedded ML-Anwendungen in Kombination mit dem Arduino Portenta H7 Board

Das Vision Shield wird mit einem 324 x 324 Pixel großen Kameramodul geliefert, das einen Ultra Low Power-Bildsensor enthält, der für Always-on-Vision-Geräte und -Anwendungen entwickelt wurde. Die hochempfindlichen Bildsensoren können Gesten, Umgebungslicht, Näherungserkennung und Objektidentifikation erfassen.

- professionelle Bildverarbeitung
- gerichtete Audioerkennung
- Ethernet oder LoRa (je nach Shield)
- JTAG
- für Arduino Portenta

Arduino hat sich mit OpenMV zusammengetan, um eine kostenlose Lizenz für die OpenMV-IDE anzubieten, die mit MicroPython einen einfachen Einstieg in die Bildverarbeitung ermöglicht. Mit OpenMV können alle Fachleute, Forscher und Entwickler kostengünstige Python-basierte Kamera-Vision- und Audio-Anwendungen entwickeln. Mit den beiden in alle Richtungen integrierten Digitalmikrofonen können Sie Töne zu den Videos aufnehmen, welche auf einer MicroSD-Karte gespeichert werden können. Übertragen Sie Daten entweder über Ethernet- oder LoRa® ARD SHD ASX00021 (Shield mit LAN) ARD SHD ASX00026 (Shield mit LoRa)

Technische Daten • Kamera: Himax HM-01B0 Kameramodul • Auflösung: 320 x 320 aktive Pixelauflösung mit Unterstützung für QVGA • Bildsensor: Hochempfindliche 3,6- μ -BrightSenseTM-Pixeltechnologie • Mikrofone: 2x MP34DT06JTR • Verbindung: LAN • Schnittstelle: JTAG • Maße (LxB): 66 x 25 mm

Lieferumfang Portenta Shield mit LAN-Funktion

26.9. Accessing IMU Data on Nicla Vision

Learn how to access the data from the accelerometer and gyroscope that comes with the LSM6DSOXTR IMU module.

26.9.1. Overview

In this tutorial, we will learn how to access the gyroscope and accelerometer that is on the Nicla Vision board. For this, we will be using the Arduino_LSMDS63 library and the Arduino IDE. Printing the values in the serial monitor of the Arduino IDE.

26.9.2. Goals

The goals of this project are:

- Read accelerometer data.
- Read gyroscope data.
- Print the data in the Serial Monitor.

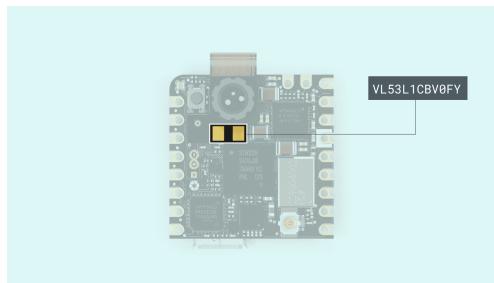


Figure 26.7.: Arduino Nicla Vision - Time of Flight sensor

26.9.3. Hardware & Software Needed

- Arduino IDE (online or offline).
- LSM6DSOX library
- Nicla Vision board

26.10. Proximity Detection with Arduino Nicla Vision

Learn how to use the proximity sensor to vary the speed of the LED's blink.

26.10.1. Overview

In this tutorial you will use the Nicla Vision to detect proximity, thanks to the Time of Flight (ToF) sensor VL53L1X.

This tutorial goes through how to create a sketch that will blink the built-in RGB LED and control the speed of its blink with the proximity values. It can be useful for future projects where there is a need to control the camera only when something is detected in front of the sensor.

The Arduino sketch shown is available inside the library [Arduino_Pro_Tutorials](#) by going to Examples > Nicla Vision > Proximity_Blink

26.10.2. Goals

The goals of this project are:

- Set up the needed libraries
- Learn how to interact with the proximity readings
- Change the RGB values of the LED

Required Hardware and Software

- Nicla Vision board
- VL53L1X library (Available in the Library Manager)

26.10.3. Instructions

Time of Flight Sensor

To make sure that the sketch works properly, the latest versions of the Arduino mbed Core and the VL53L1X library needs to be installed. Both can be found inside the Arduino IDE.

- The Arduino mbed Core can be found in the boards manager ...
- The VL53L1X library can be found in the Library manager,

If you are using version 1.6.2 or later of the Arduino software (IDE), you can use the Library Manager to install the VL53L1X library:

- In the Arduino IDE, open the "Sketch" menu, select "Include Library", then "Manage Libraries...".
- Search for "VL53L1X".
- Click the VL53L1X entry in the list, authored by Pololu.
- Click "Install". More detailed instructions on how to install a library can be found [here](#).

Include the Needed Libraries and Objects Declaration

First of all declare the sensor's class so you can access it later on in your sketch. We use variables to control the time elements in the sketch. This will make sure that the readings stay accurate over time.

Initialize the Proximity Sensor and the LED

Inside the setup you need to initialize and configure the proximity sensor. Also the RGB LED needs to be set as an output to make it light up and enable us to change its behavior.

Note: The LEDs are accessed in the same way as on the Portenta H7: LEDR, LEDG and LEDB

Note: Make sure you initialize Wire1, set the clock speed to 400kHz and set the bus pointer to Wire1, it won't work if you don't add these setting

Control the Speed of the Blink

The sketch is going to get the reading on every loop, store it and then the state of the LED will change, until the time is up and another proximity reading is taken.

Complete Sketch

API

26.10.4. Conclusion

In this tutorial we went through how to get readings from the ToF sensor. And how use these readings to change how the built-in LED behaves. At the end of the tutorial you can also find a reference list for the ToF library.

```

#include "VL53L1X.h"
VL53L1X proximity;

bool blinkState = false;
int reading = 0;
int timeStart = 0;
int blinkTime = 2000;

void setup() {
    Serial.begin(115200);
    Wire1.begin();
    Wire1.setClock(400000); // use 400 kHz I2C
    proximity.setBus(&Wire1);

    pinMode(LED_BUILTIN, OUTPUT);
    digitalWrite(LED_BUILTIN, blinkState);

    if (!proximity.init()) {
        Serial.println("Failed to detect and initialize sensor!");
        while (1);
    }

    proximity.setDistanceMode(VL53L1X::Long);
    proximity.setMeasurementTimingBudget(10000);
    proximity.startContinuous(10);
}

void loop() {
    reading = proximity.read();
    Serial.println(reading);

    if (millis() - timeStart >= reading) {
        digitalWrite(LED_BUILTIN, blinkState);
        timeStart = millis();

        blinkState = !blinkState;
    }
}

```

Listing 26.1.: Proximity-Sketch

Command	Details
<code>setAddress(newAddress)</code>	Change the I2C sensor's address (Mandatory to set it to Wire1)
<code>getAddress()</code>	Get the Sensor's I2C address
<code>init()</code>	Configures the sensor and needed data. Like the usual begin()
<code>setDistanceMode(mode)</code>	Set the distance mode (check the datasheet). Available modes
<code>getDistanceMode()</code>	Returns the mode that has been set. Available modes VL53L1X
<code>setMeasurementTimingBudget(uSeconds)</code>	Set the time to get the measure, greater the value, better precision
<code>getMeasurementTimingBudget()</code>	Get the measure timing value in micro seconds.
<code>startContinuous()</code>	Start the non stop readings, set the period inside the parameter
<code>stopContinuous()</code>	Stop the non stop measurements.
<code>read()</code>	Get the last reading from the continuous mode.
<code>readSingle()</code>	Trigger one reading and get its result.
<code>dataReady()</code>	Returns if the sensor has new data available.
<code>setTimeout(mSeconds)</code>	Configure the milliseconds the sensor will wait in case it is not ready
<code>getTimeout()</code>	Get the configured timeout value.
<code>timeoutOccurred()</code>	Returns true whenever the sensor had a timeout.

27. Gesichtserkennung mit Edge Impulse

In diesem Kapitel wird eine Modell zur Gesichtserkennung mit Hilfe der Plattform Edge IMPulse entwickelt. Ein weiteres Werkzeug, beispielsweise OpenMV IDE, wird nicht benötigt. Das Flussdiagramm für den Gesamtprozess ist in der folgenden Abbildung 27.1 dargestellt.

WS:Quelle

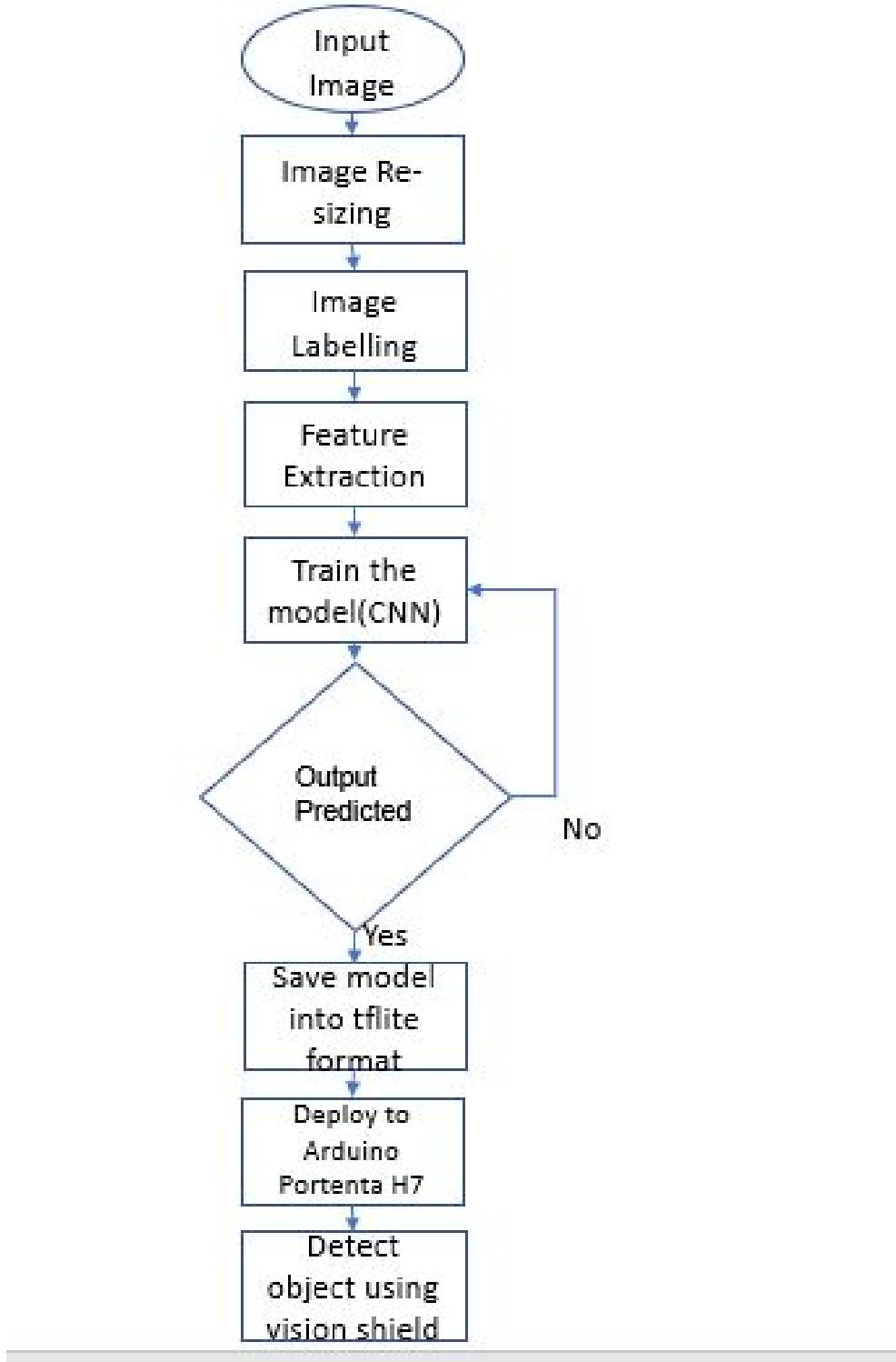


Figure 27.1.: Flussdiagramm des Gesamtprozesses

27.1. Datenbank

Zum Training der Modells wird der Datensatz UTKFace, siehe Kapitel 4.7, verwendet. Es können aber auch eigene Bilder hinzugefügt werden. Letztlich setzt sich der

verwendete Datensatz wie folgt zusammen:

- Manuell aufgenommene Bilder von der Kamera, Bilder aus dem UTKFace-Datensatz.
- Trainingssatz: Bestehend aus 388 Bildern
- Testsatz: bestehend aus 98 Bildern

Da unser Datensatz nun vorbereitet ist, können wir das Modell auf dem Plattform Edge Impulse trainieren. Zu diesem Zweck müssen wir ein Projekt auf der Edge Impulse-Plattform erstellen. Dies ist in Kapitel ?? beschrieben. Um Bilder, die sich auf dem Arduino Board befinden, hochzuladen, muss der Arduino zuerst mit der Plattform Edge Impulse verbunden werden. Dies wird im Abschnitt ?? beschrieben.

27.1.1. Kennzeichnung von Daten

Nachdem unsere Daten hochgeladen wurden, müssen wir sie nun beschriften. Dazu gehen wir auf die Option „Labelling Queue“ in der Datenerfassung und beginnen mit dem Zeichnen eines Begrenzungsrahmens um das Objekt und beschriften es. In unserer Datenbank haben wir zwei verschiedene Objekte, nämlich Flächen und Stifte. Wir zeichnen also manuell einen Begrenzungsrahmen um das Objekt und beschriften ihn für alle Bilder im Datensatz. 27.2

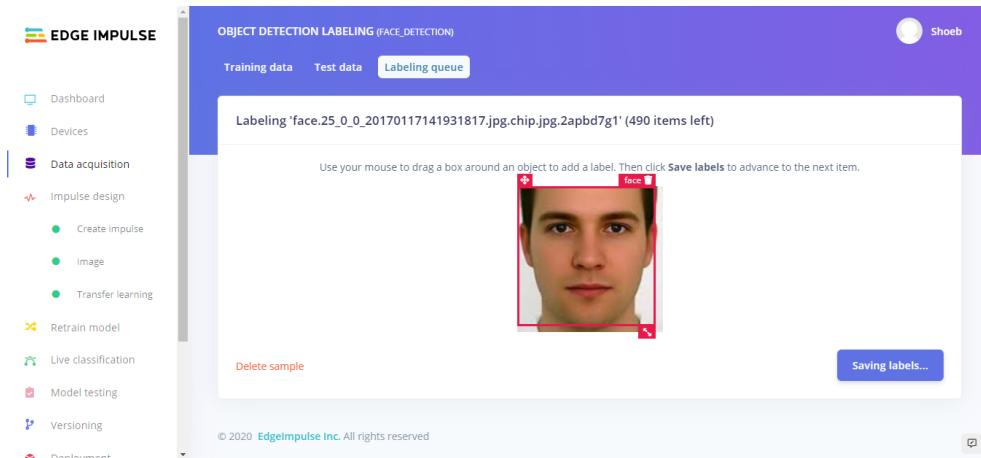


Figure 27.2.: Beschriftung eines Bildes

27.2. Umwandlung von Daten

27.2.1. Die Gestaltung des Impulses

Vorverarbeitung von Bildern:

Um den Impuls zu entwerfen, klicken Sie auf Impulsdesign und stellen Sie dann die Bildbreite und -höhe auf 320×320 ein, da das vortrainierte Objekterkennungsmodell nur mit Bildern dieser Größe funktioniert [Tea21b]. Der Bildvorverarbeitungsblock nimmt das Eingabebild auf und kann es optional in Graustufen umwandeln und die Daten dann in ein Merkmalsfeld umwandeln. Dann klicken wir auf den Lernblock und wählen Objekterkennung. Klicken Sie dann auf Impuls speichern, wie in der Abbildung 27.3 gezeigt.

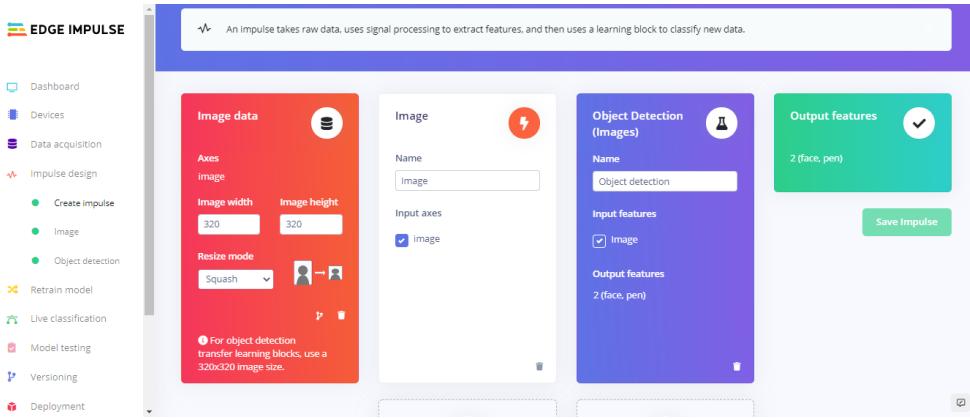


Figure 27.3.: Impulse schaffen

27.2.2. Konfigurieren des Bildverarbeitungsblocks

Um den Verarbeitungsblock zu konfigurieren, klicken Sie im Menü auf der linken Seite auf Bilder. Dies zeigt uns die Rohdaten oben auf dem Bildschirm und die Ergebnisse des Verarbeitungsschritts rechts. Wir können auch die Optionen verwenden, um zwischen den Modi „RGB“ und „Grayscale“ zu wechseln. Klicken Sie dann auf Parameter speichern, wie in Abbildung 27.4 gezeigt.

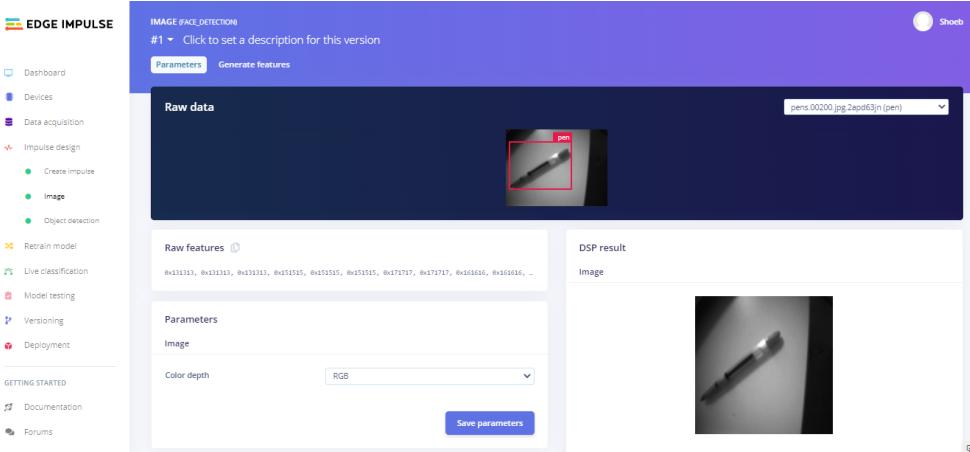


Figure 27.4.: Parameter speichern

Klicken Sie dann oben auf die Option „generate features“ und dann auf „generate features“. In der Abbildung 27.5 ist zu sehen, dass die aus dem Datensatz der Bilder, die aus Stiften und Gesichtern bestehen, generierten Merkmale leicht zu erkennen sind.

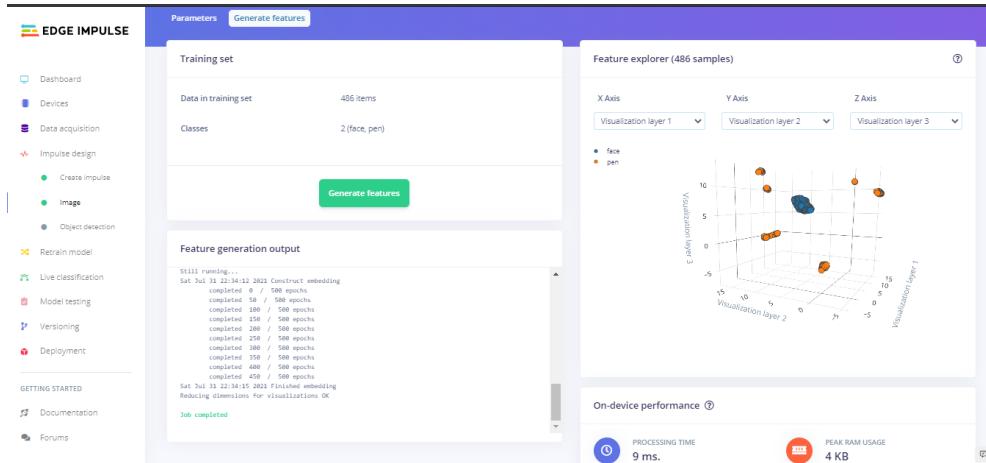


Figure 27.5.: Merkmale generieren

27.3. Data training

Konfigurieren des Transfer-Learning-Modells:

In diesem Fall verwenden wir den Transfer-Learning-Block, da es einfacher ist, ein bereits trainiertes Modell zu verwenden, indem nur die oberen Schichten eines neuronalen Netzes beibehalten werden, und wir das Modell dann in kurzer Zeit trainieren können. Um das Transfer-Learning-Modell zu konfigurieren, klicken Sie auf Objekterkennung. Hier können wir die Lernrate einstellen, in unserem Fall haben wir sie auf 0,015 und die Anzahl der Trainingszyklen auf 25 eingestellt. Dann wählen wir das Basismodell aus, in unserem Fall MobileNetV2, und klicken dann auf Training starten. Nachdem der Trainingsprozess abgeschlossen ist, können wir in der Abbildung 27.6 sehen, dass die Genauigkeit 87% beträgt.

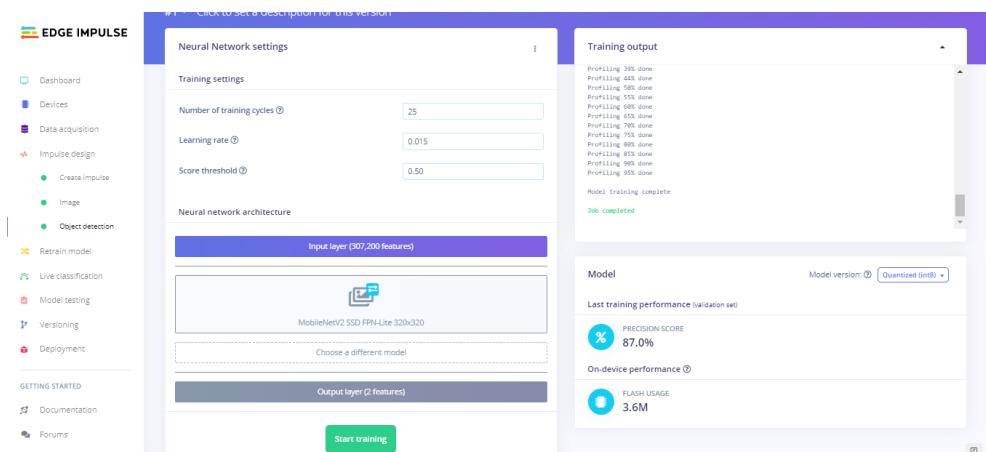


Figure 27.6.: Abgeschlossenes Training

27.4. Prüfung des Modells

Um das Modell zu testen, müssen wir auf die Option „model testing“ und dann auf „classify all“ klicken. Hier wurden die Bilder des Datensatzes automatisch in Trainings- und Testbilder unterteilt. In diesem Fall hat das Modell die Eingabebilder in Trainingsbilder mit 388 Bildern und Testbilder mit 98 Bildern unterteilt. Die

Testbilder werden dann verwendet, um zu überprüfen, wie gut das Modell funktioniert. Wie aus der Abbildung 27.7 hervorgeht, erreichte das Modell eine Genauigkeit von 87,1 %.

The screenshot shows the Edge Impulse web interface under the 'Model testing' section. On the left, a sidebar lists various options like Dashboard, Devices, Data acquisition, Impulse design, Retrain model, Live classification, Model testing (which is selected), Versioning, Deployment, Getting Started, Documentation, and Forums. The main area is titled 'MODEL TESTING FACE_DETECTION'. It has three tabs: 'Test data', 'Classify all', and 'Model testing output'. The 'Test data' tab shows a table of test samples, each with a sample name, expected outcome ('pen'), length, precision score (e.g., 100%), and result. The 'Model testing output' tab shows logs of the classification process, including 'Copying features from processing blocks...', 'Classifying data for Object detection...', and 'Job completed'. The 'Model testing results' tab displays an accuracy of 87.10% with a green circular icon.

Figure 27.7.: Prüfung des Modells

Wir können auch die Live-Klassifizierung durchführen und Objekte in Echtzeit mit dem Vision Shield erkennen. Klicken Sie im Menü auf die Option „live classification“ und verbinden Sie unser Portenta H7-Gerät mit dem Edge Impulse, indem Sie das Gerät auswählen. Sobald es angeschlossen ist, können wir den Kamerastream vom Vision Shield sehen. Hier haben wir das Modell getestet, indem wir uns auf das Gesicht konzentriert haben, und wie wir in der Abbildung 27.8 sehen können, gibt es die Ausgabe als Gesicht mit einem Wert von 0,89, was eine ziemlich gute Vorhersage ist.

The screenshot shows the Edge Impulse web interface under the 'Live classification' section. The sidebar is identical to Figure 27.7. The main area is titled 'Classification result'. It shows a summary table with a 'Name' field containing 'testing.jpg.2bqkdf8h', a 'Precision score' of 0.00%, and a 'face' category with a count of 1. To the right, there's a preview window titled 'RAW DATA' showing a portrait of a man with the file name 'testing.jpg.2bqkdf8h'. Below it, a 'Raw features' section lists binary data. Further down, another preview window titled 'Classification result' shows the same portrait with a red bounding box around the face and the text 'face 0.89'.

Figure 27.8.: Live-Klassifizierung mit Hilfe der Vision-Shield-Kamera

WS:Im Programm wird die Methode tf.classify verwendet. Details!!!

28. MicroPython & OpenMV

28.1. MicroPython

29. First Application: Object Detection

30. Training a Custom Machine Learning Model for Portenta H7

<https://www.arduino.cc/pro/tutorials/portenta-h7/vs-openmv-ml>

31. Use TensorFlow Lite with Portenta H7

Part IV.

Anhang

1	ARD PORTENTA H7 Arduino Portenta H7, STM32H747, ohne Header	www.reichelt.de	97,30 €
- ARD POR-TENTA H7			
1	Arduino Portenta Shield - Vision mit LAN	www.reichelt.de	49,75 €
- ARD SHD ASX00021			
1	USB 2.0 Kabel USB-C auf USB-C, 0,5 m	www.reichelt.de	17,20 €
- ST USB2CC50CM			
1	USB C Stecker auf DP Kabel, DP Mode, 4K60, 1 m, schwarz	www.reichelt.de	16,90 €
- SON X-UCC020-010			
1	Netzteil		€
1	Monitor		€

Stand: 18.05.2021

33. Datenblätter

33.1. Datenübersicht



Portenta H7 module

Family Overview

Arduino's series of high-performance industry-rated boards

Overview

Portenta H7 simultaneously runs high level code along with real time tasks

Program it with high-level languages and AI while performing low-latency operations on its customizable hardware

Its two asymmetric cores can simultaneously run high level code such as protocol stacks, machine learning or even interpreted languages like MicroPython or Javascript along with low-level real time tasks.

PH7's main processor is a "dual core unit" made of a Cortex® M7 running at 480 MHz and a Cortex® M4 running at 240 MHz. The two cores communicate via a "Remote Procedure Call" mechanism that allows calling functions on the other processor seamlessly.

Both processors share all the in-chip peripherals and can run:

- Arduino sketches on top of the mbedOS
- Native mbed applications
- Micropython / Javascript via an interpreter
- TensorFlow Lite

The onboard wireless module allows to simultaneously manage WiFi and Bluetooth connectivity. The WiFi interface can be operated as an Access Point, as a Station or as a dual mode simultaneous AP/STA and can handle up to 65 Mbps transfer rate. Bluetooth interface supports Bluetooth Classic and BLE.

The Portenta H7 has follows the Arduino MKR form factor, but is enhanced with the Portenta family 80 pin high-density connector. Learn more about the board's pinout by reading the board's pinout.

Applications

Use Portenta when performance is key

- High-end industrial machinery
- Laboratory equipment
- Computer vision
- PLCs
- Industry-ready user interfaces

- Robotics controller
- Mission-critical devices
- Dedicated stationary computer
- High-speed booting computation (ms)

A NEW STANDARD FOR PINOUTS

The Portenta family adds two 80 pin high density connectors at the bottom of the board. This ensures scalability for a wide range of applications by simply upgrading your Portenta board to the one suiting your needs.

Main Features

DUAL CORE - The PH7 has two processors in one, run parallel tasks in different languages

SELECT ENCRYPTION - Choose between two vendors for the crypto-chip that better suits your case

USB-C - Power your board, connect it to a display, or get it to work in OTG mode

MEMORY RANGE - Select the right amount and type of memory for your application

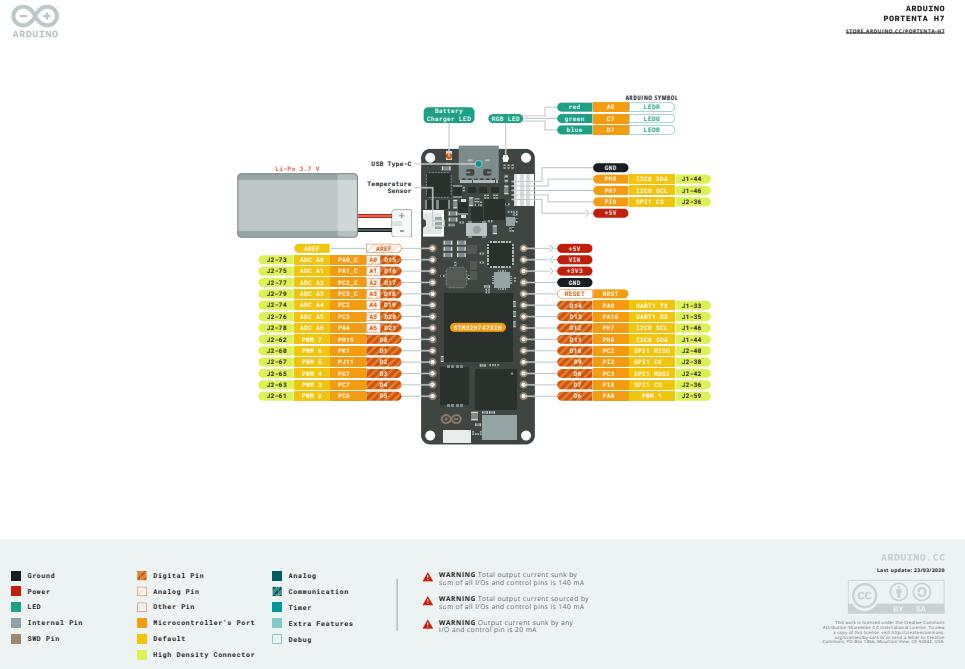
MULTIPLE OPTIONS ON ONE BOARD

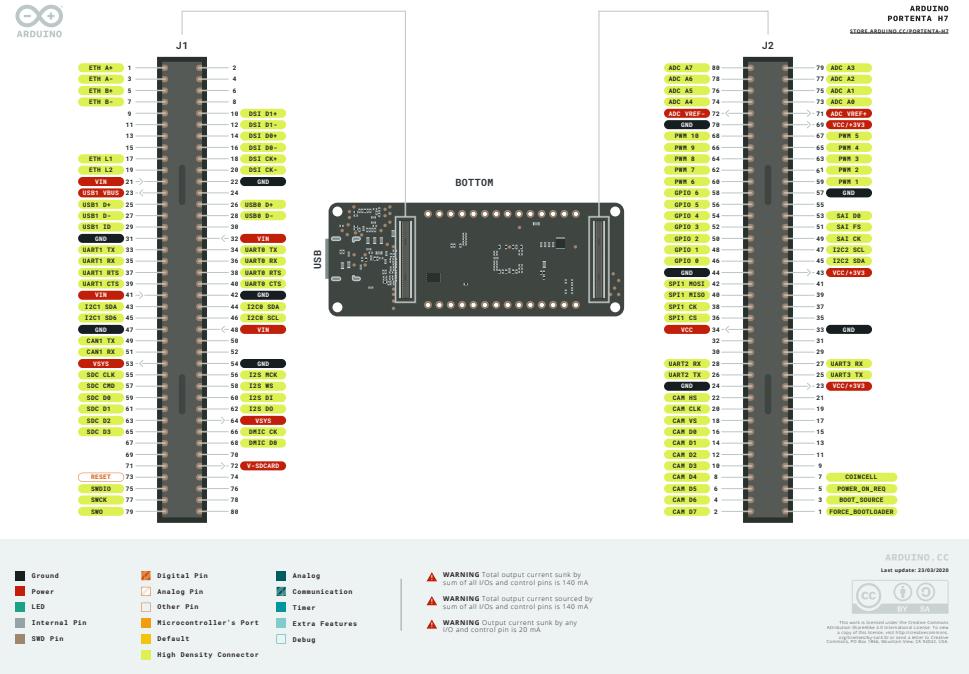
Order the default Portenta H7 using the codename 'Arduino Portenta H7-15EUNWAD'. The board comes with:

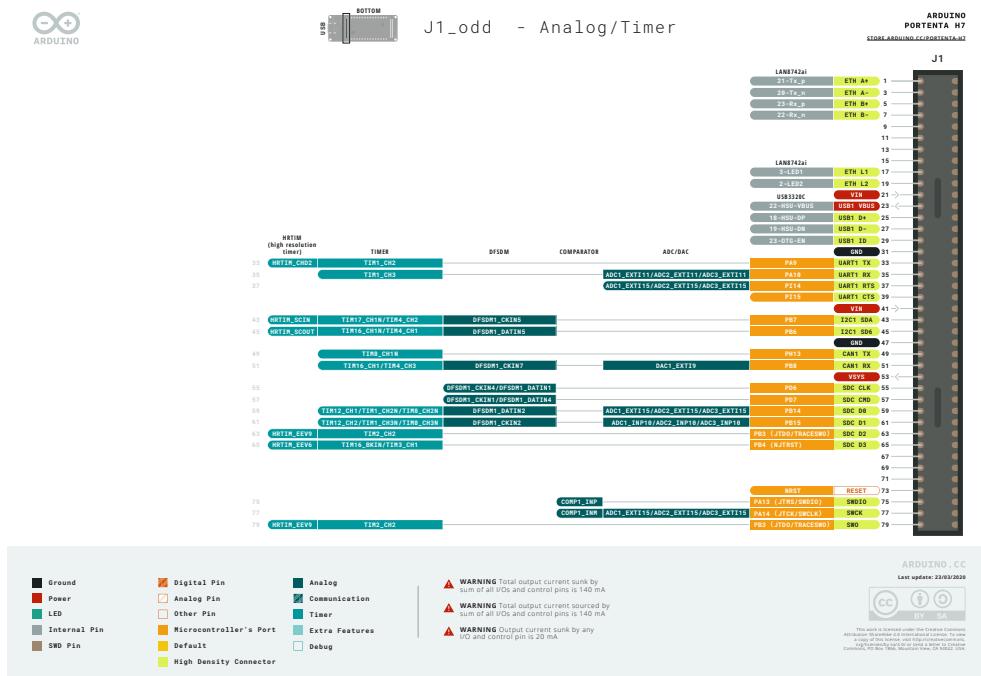
- STM32H747 dual-core processor with graphics engine
- 8MB SDRAM
- 16MB NOR Flash
- 10/100 Ethernet Phy
- USB HS
- NXP SE050C2 Crypto
- WiFi/BT Module
- Ceramic Antenna
- DisplayPort over USB-C

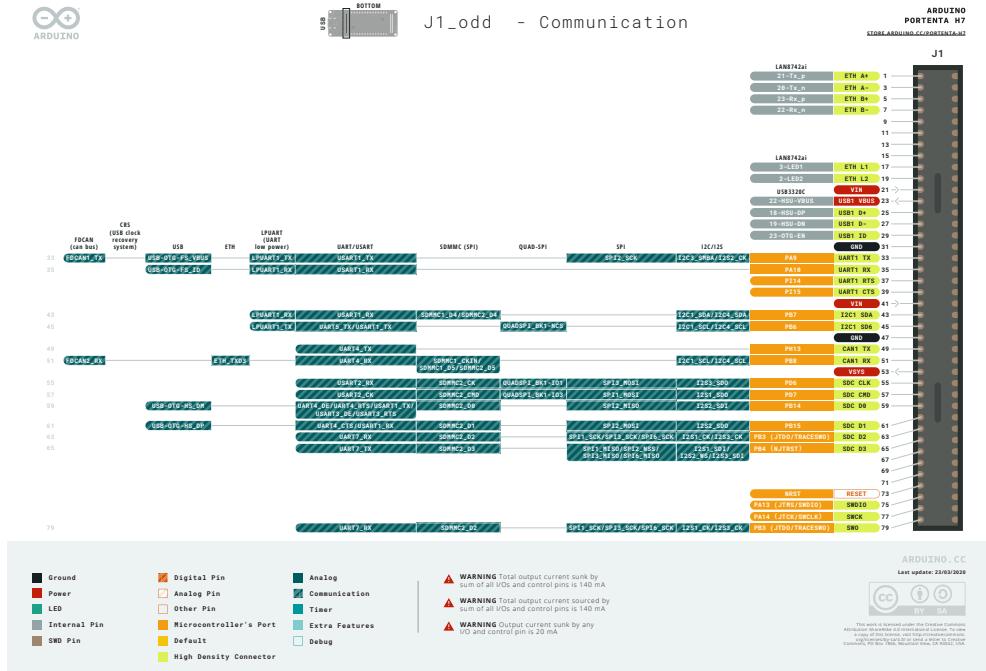
Tailor the hardware to your solution

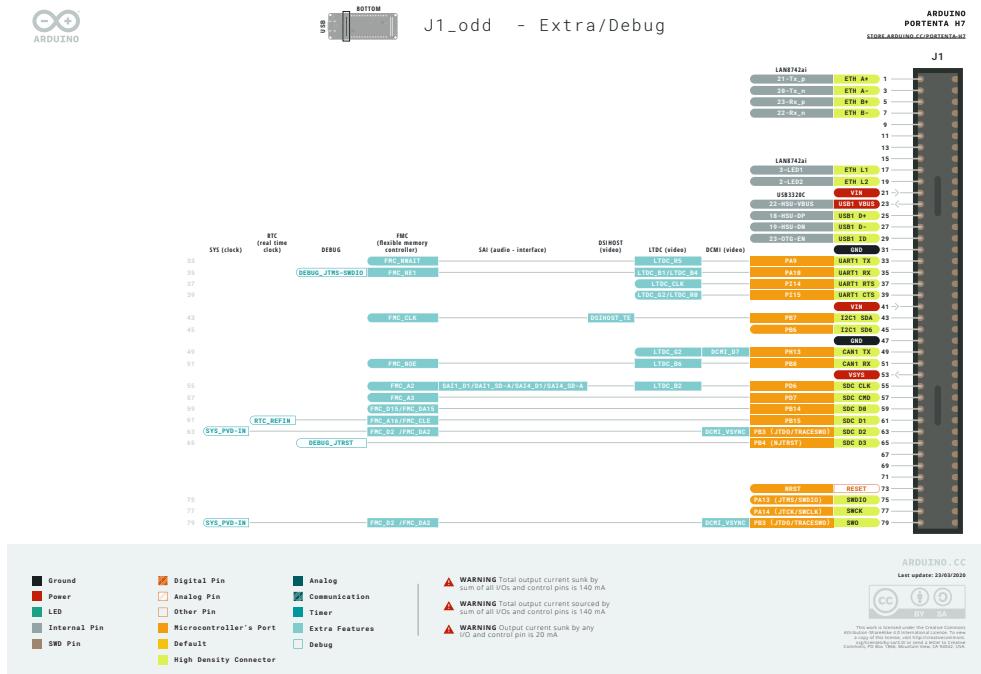
If you need more memory, Portenta H7 can host up to 64 MByte of SDRAM, and 128 MByte of QSPI Flash. Order it with an external UFL connector for adding a higher-gain antenna to the board. Decide between crypto-chips from Microchip® and NXP - the ATECC608A or the SE050C2 crypto chips.

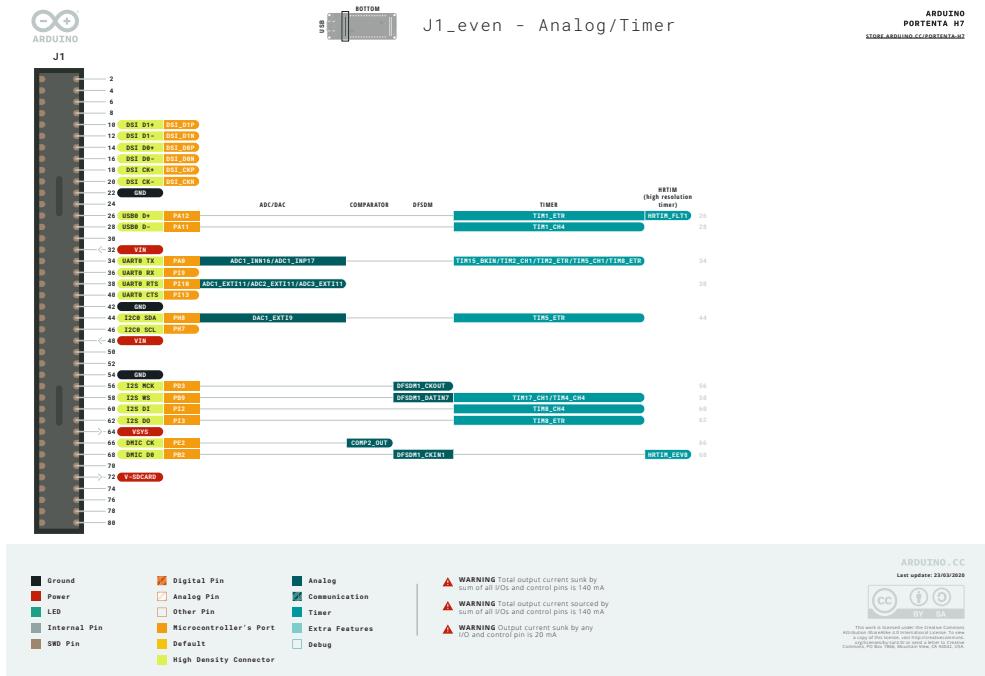


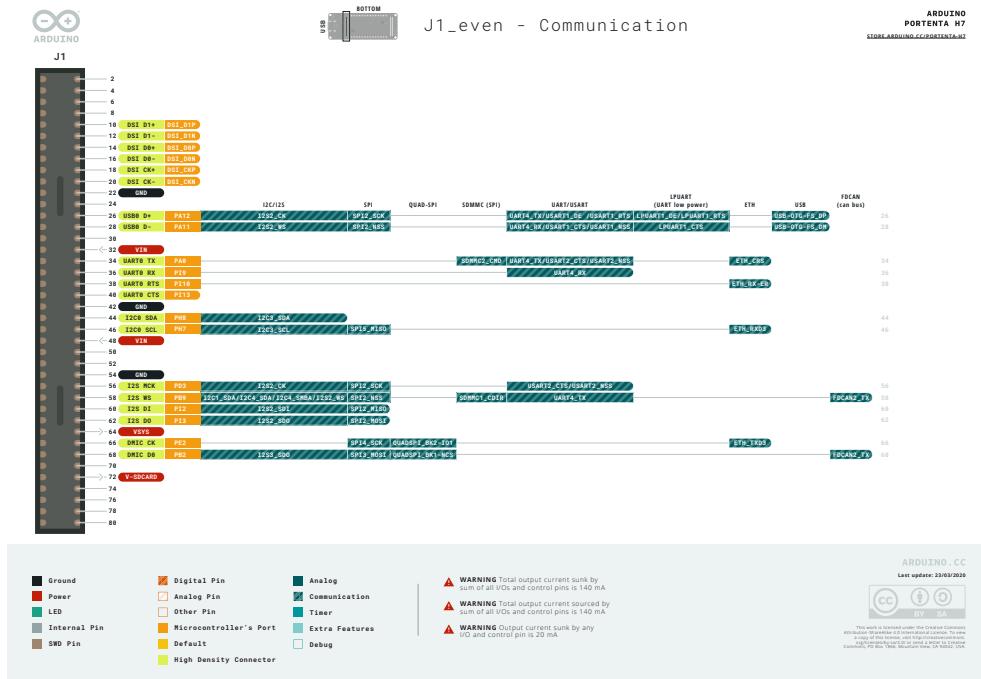


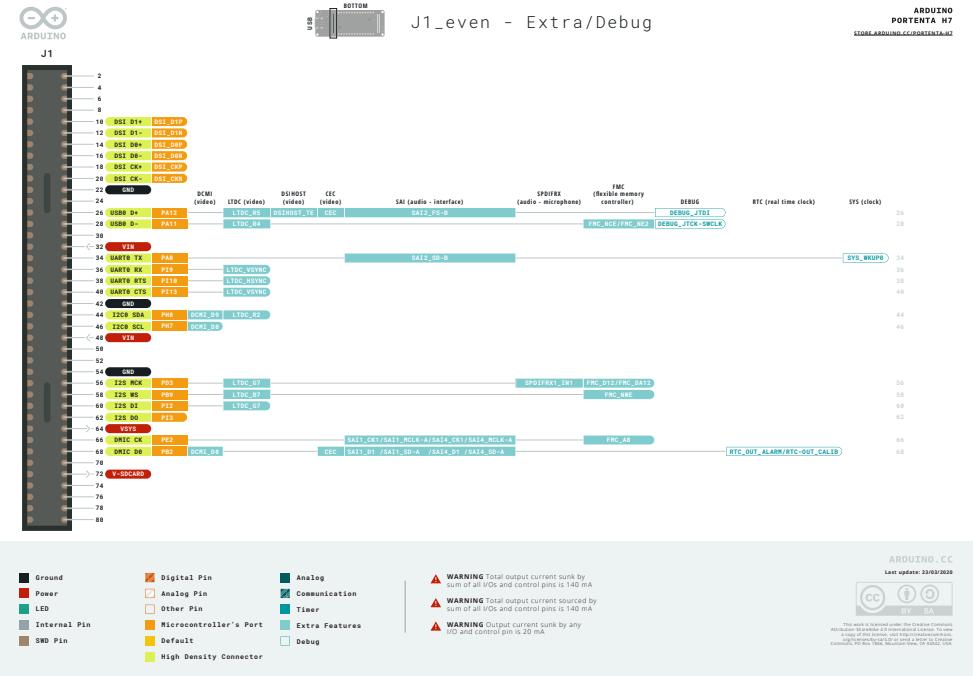


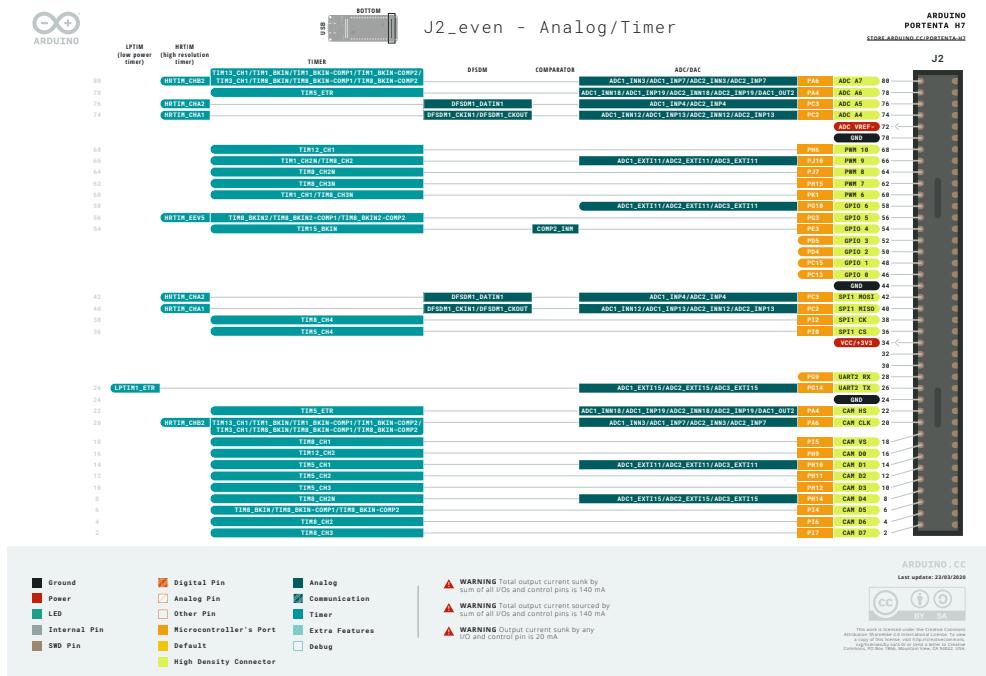


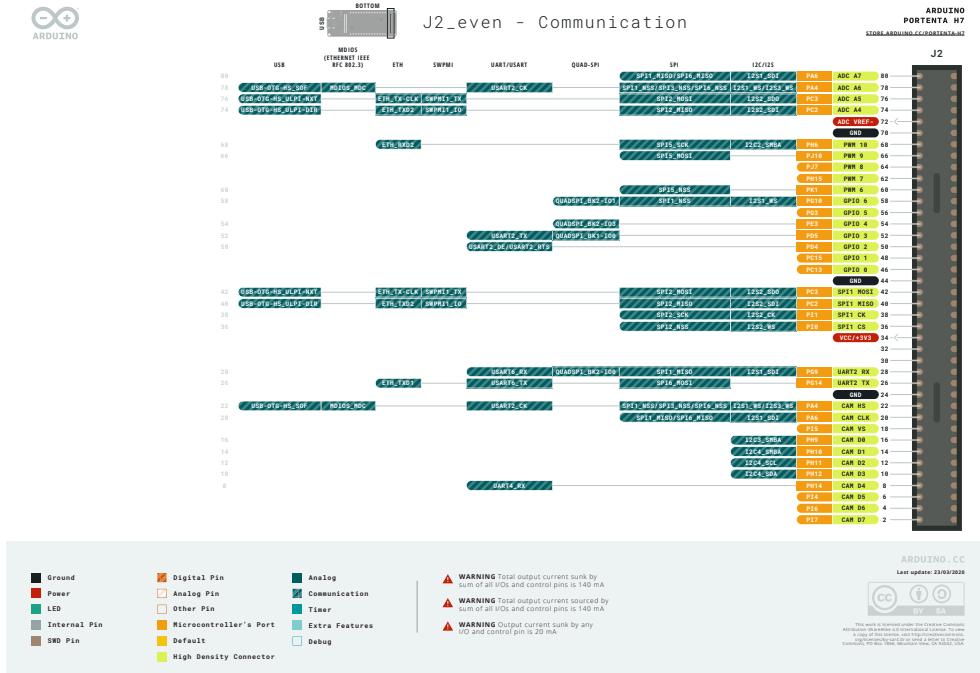


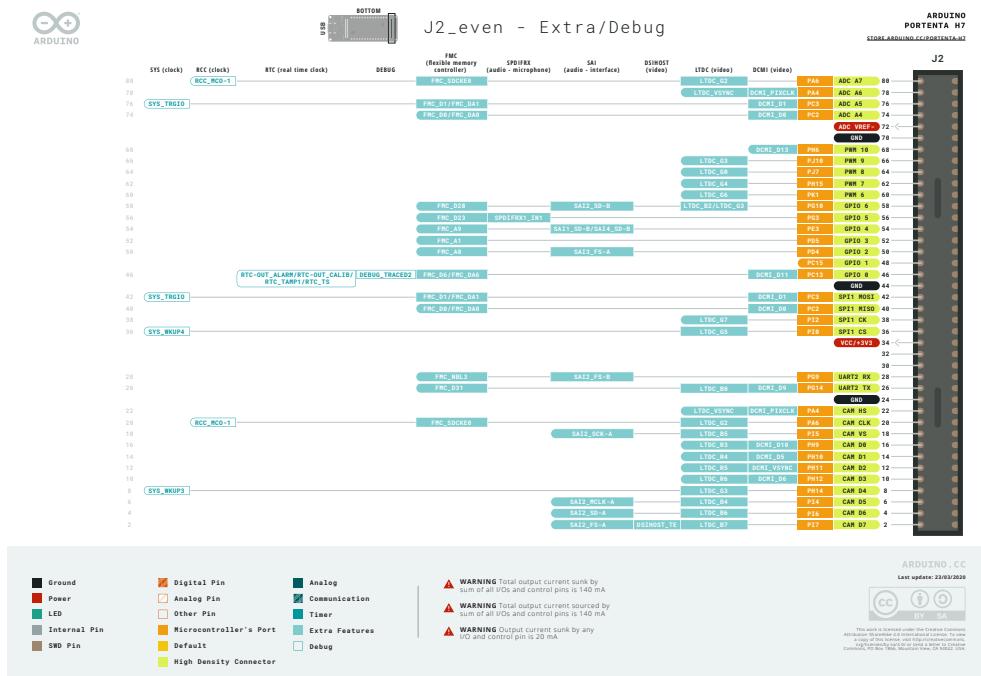


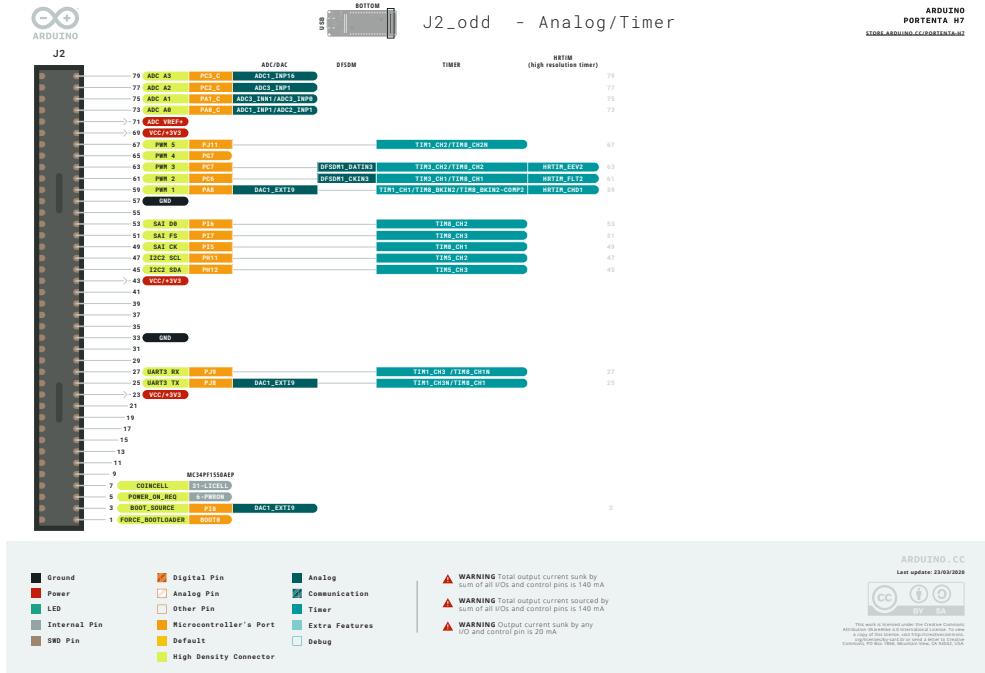


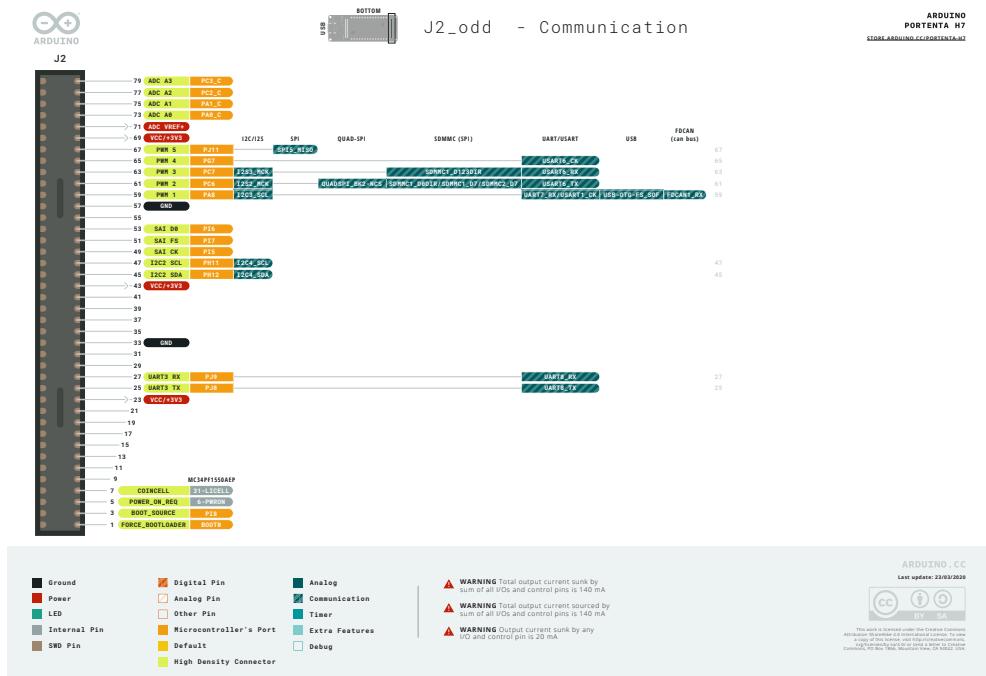


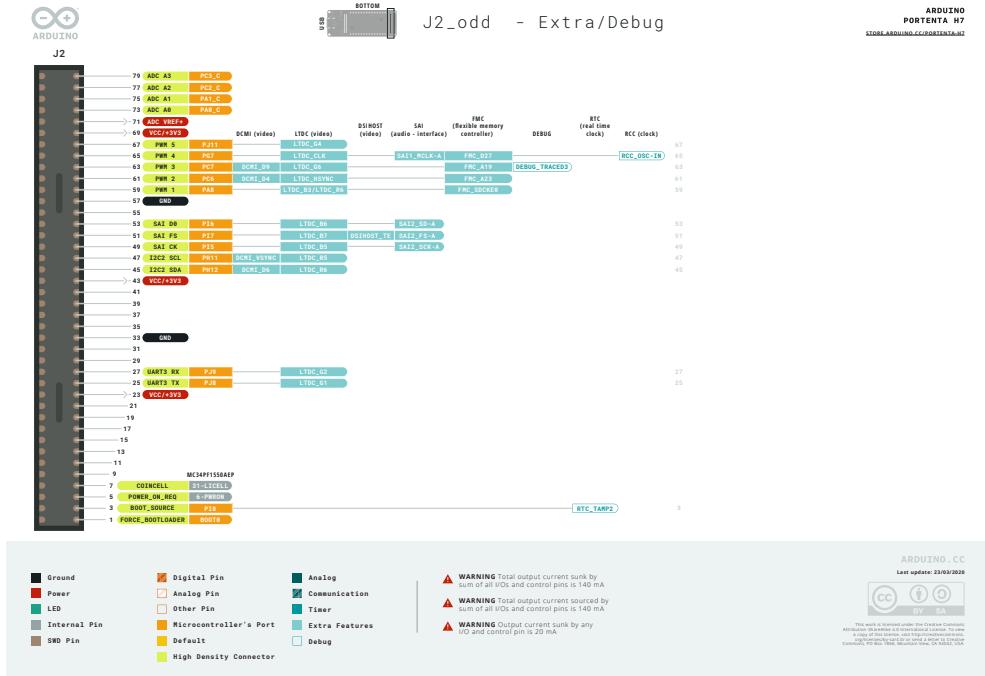


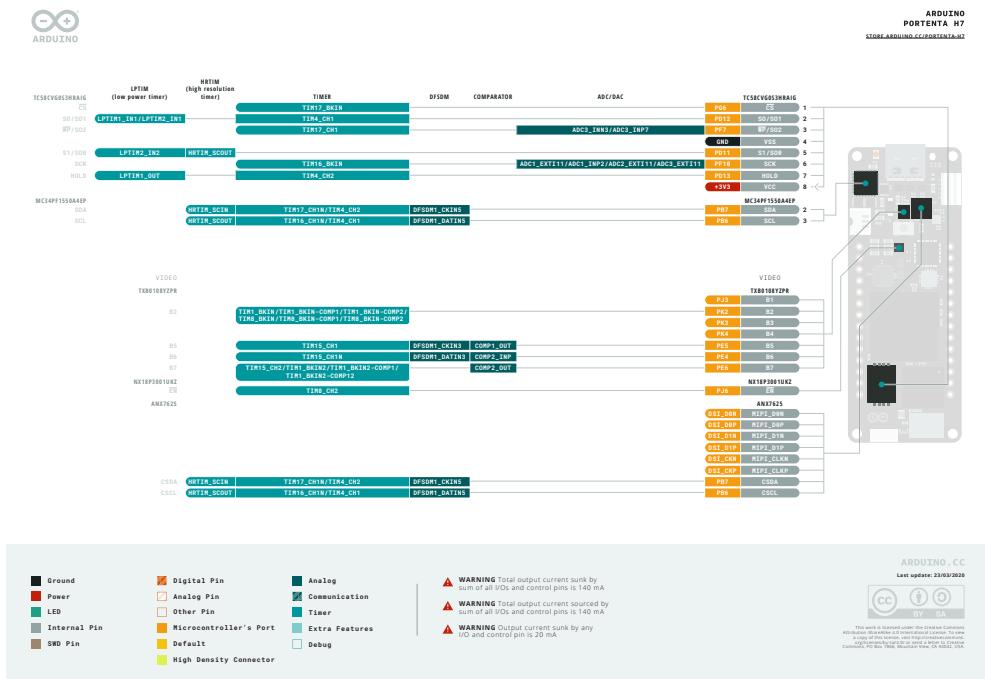


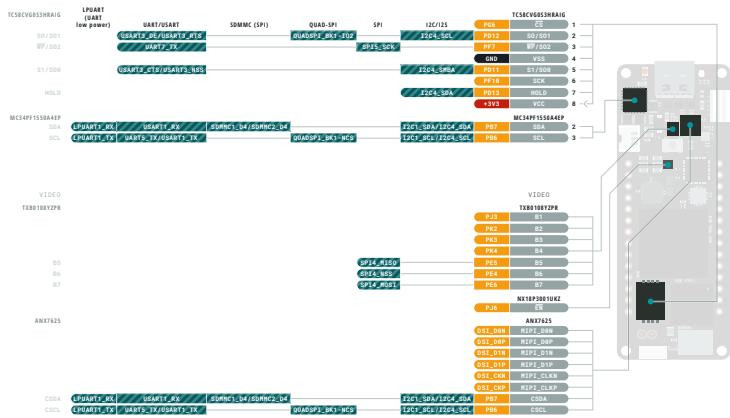












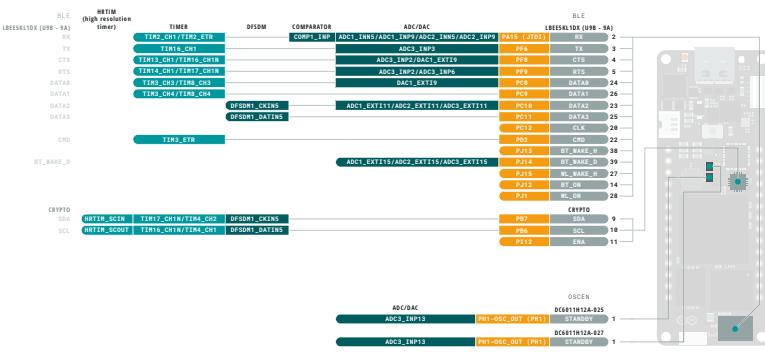
■ Ground
 ■ Power
 ■ LED
 ■ Internal Pin
 ■ SWD Pin

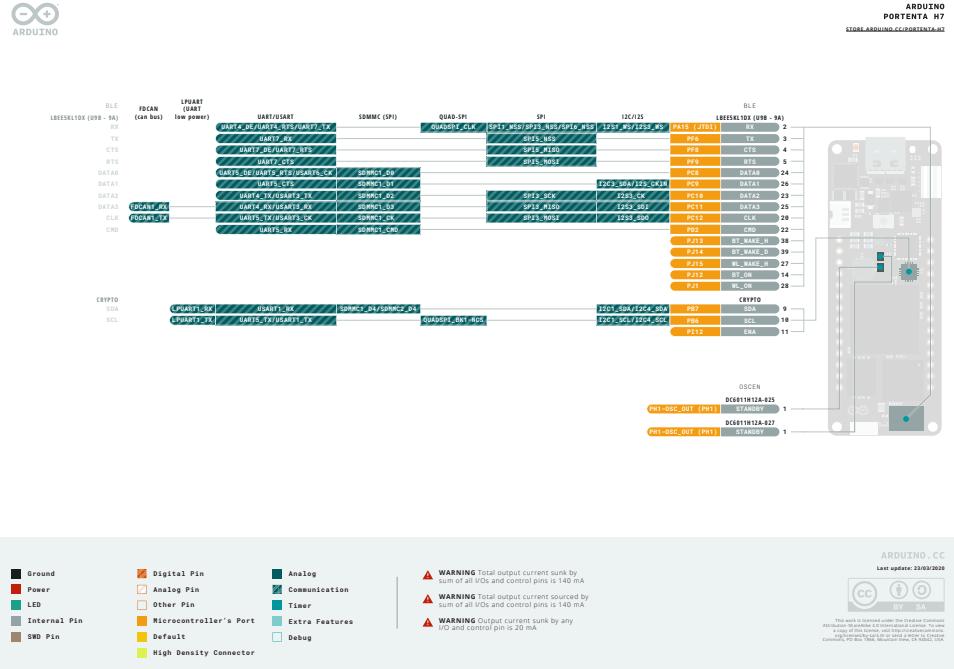
■ Digital Pin
 ■ Analog Pin
 ■ Other Pin
 ■ Microcontroller's Port
 ■ Default
 ■ High Density Connector

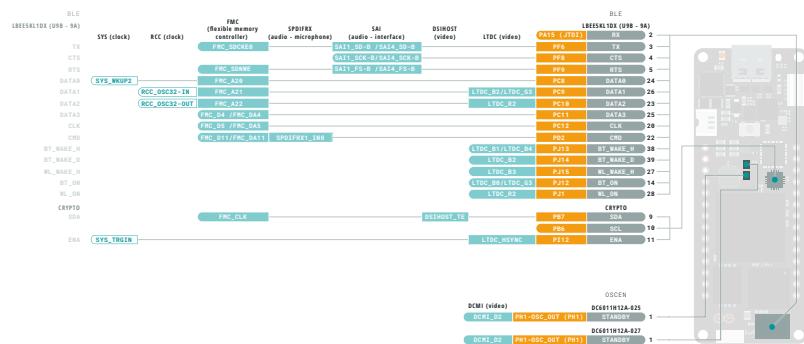
■ Analog
 ■ Communication
 ■ Timer
 ■ Extra Features
 ■ Debug

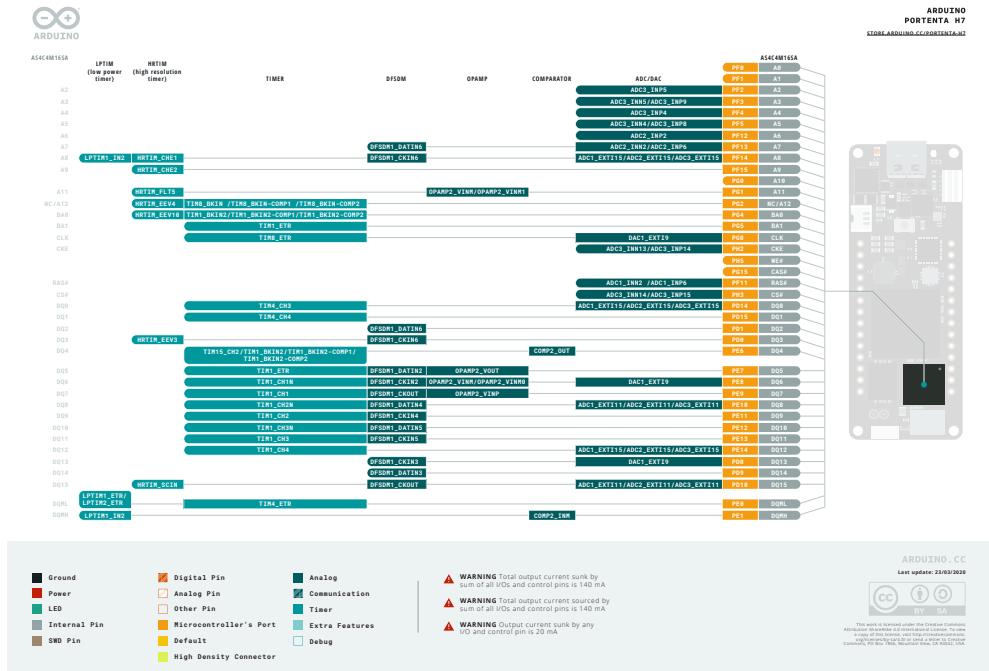
⚠ **WARNING** Total output current sunk by sum of all I/Os and control pins is 140 mA.
 ⚠ **WARNING** Total output current sourced by sum of all I/Os and control pins is 140 mA.
 ⚠ **WARNING** Output current sunk by I/O and control pin is 20 mA.



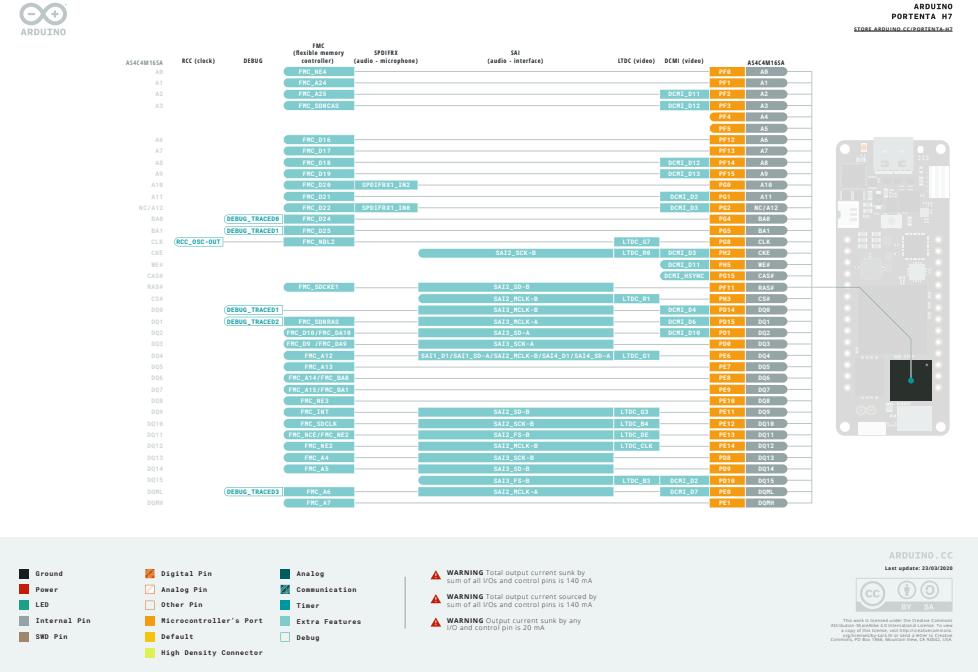


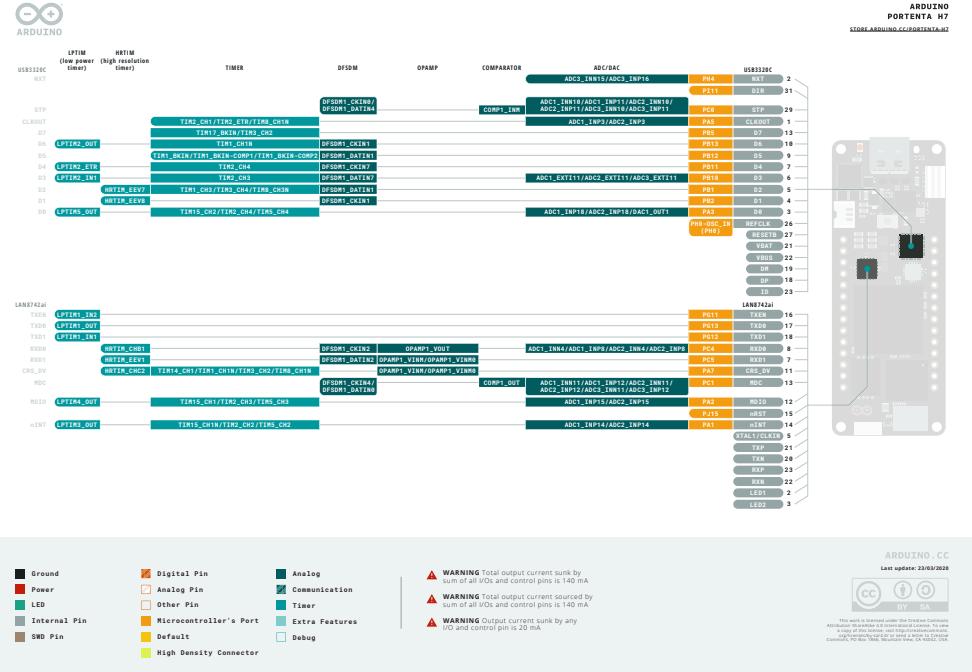


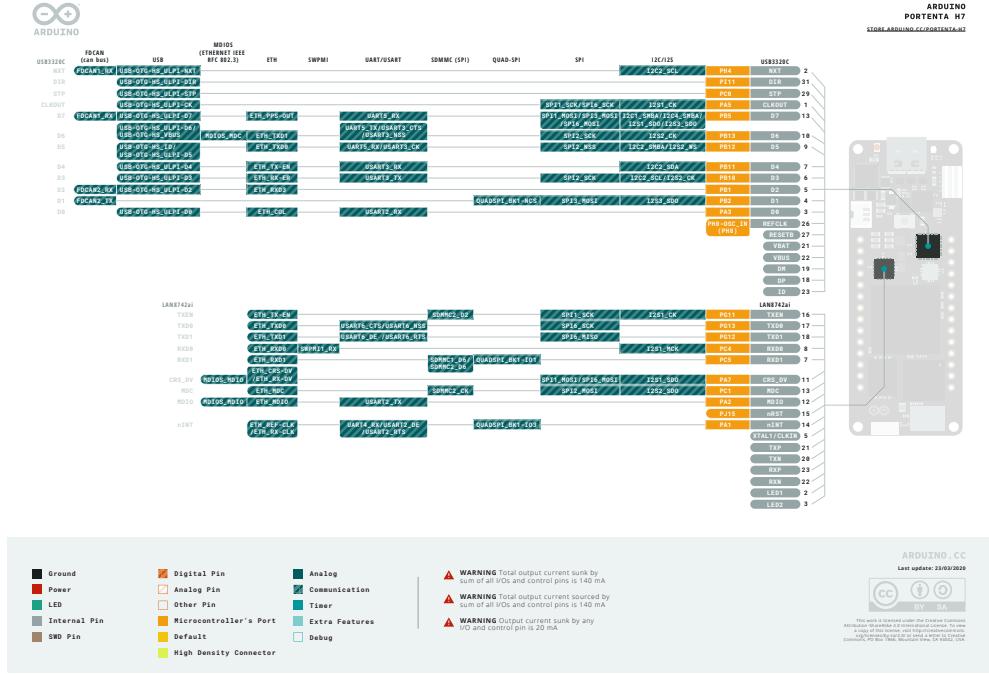


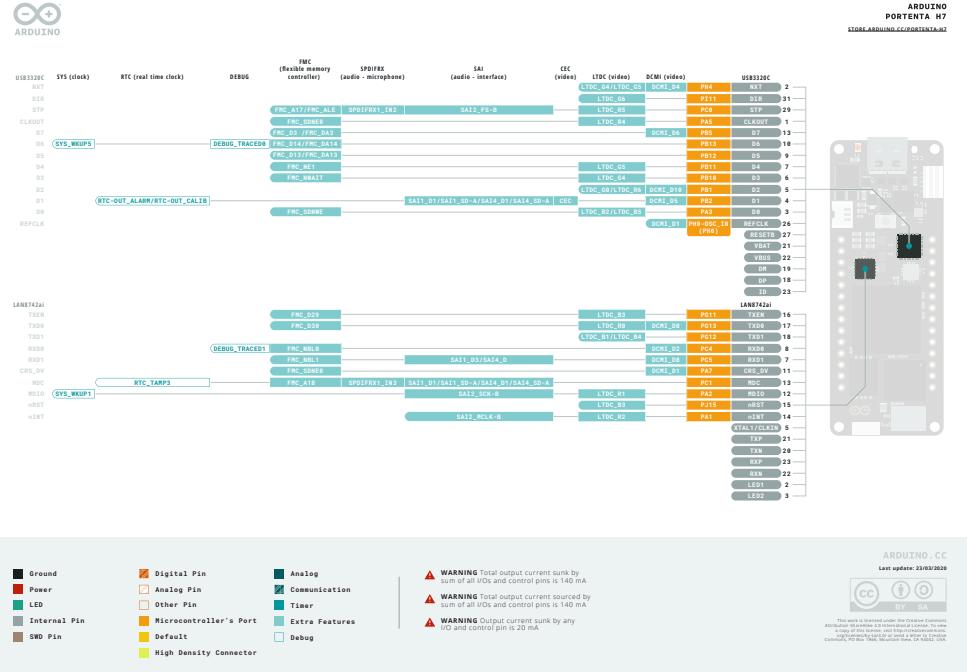


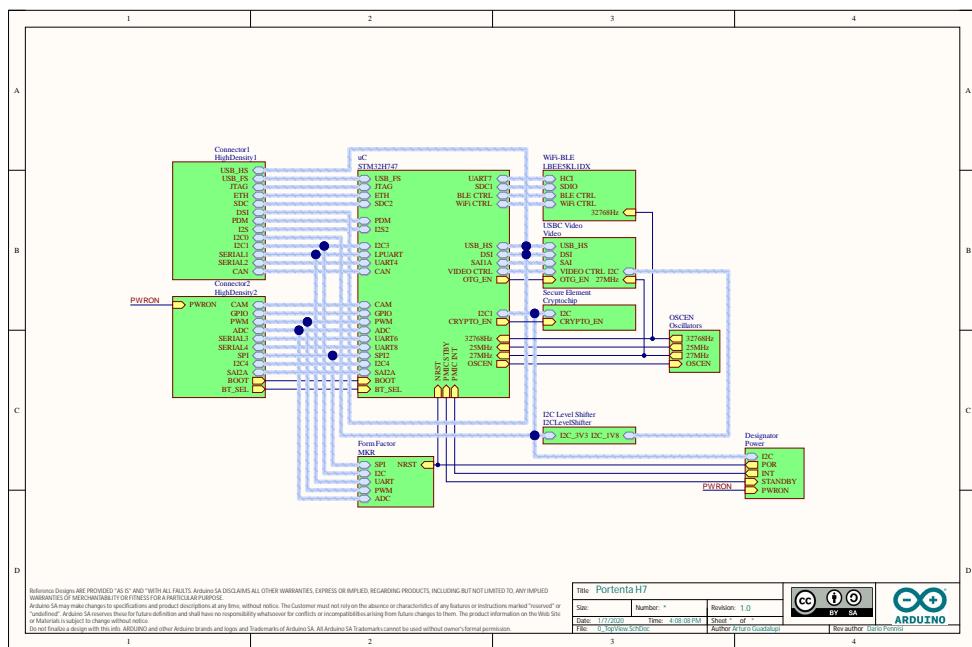


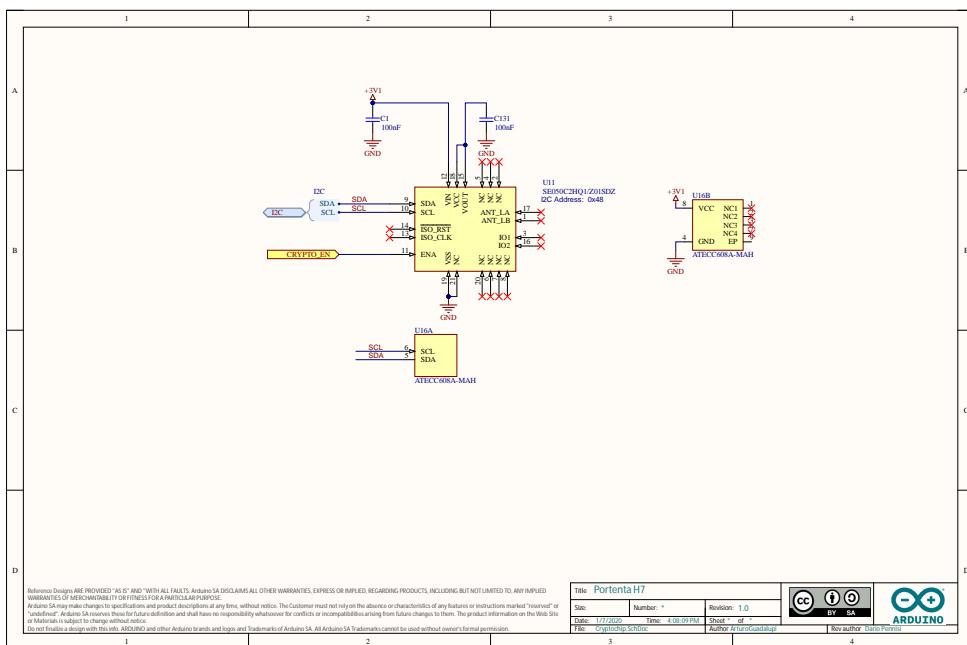


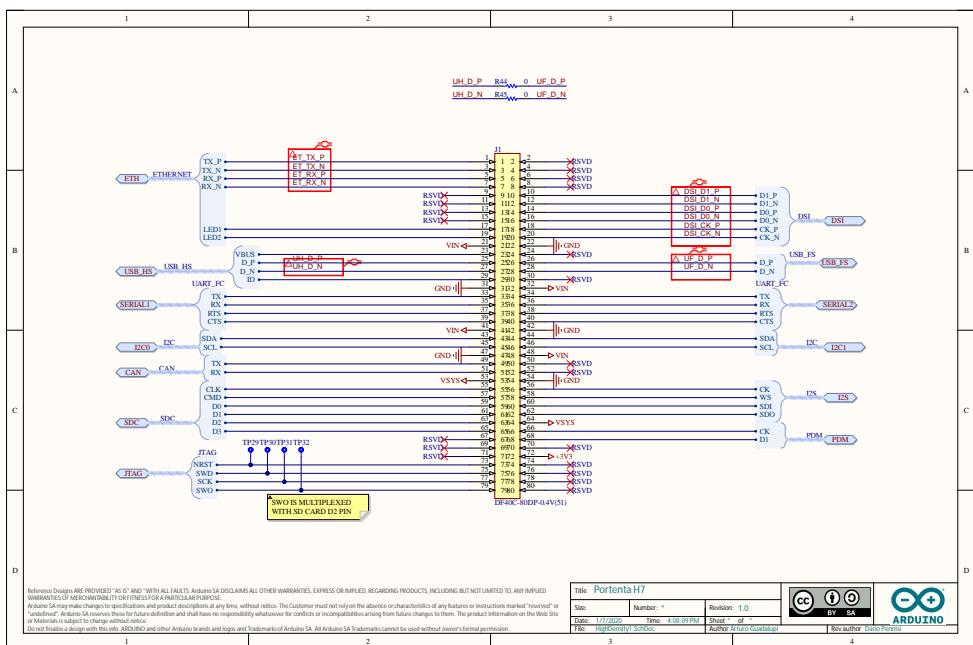


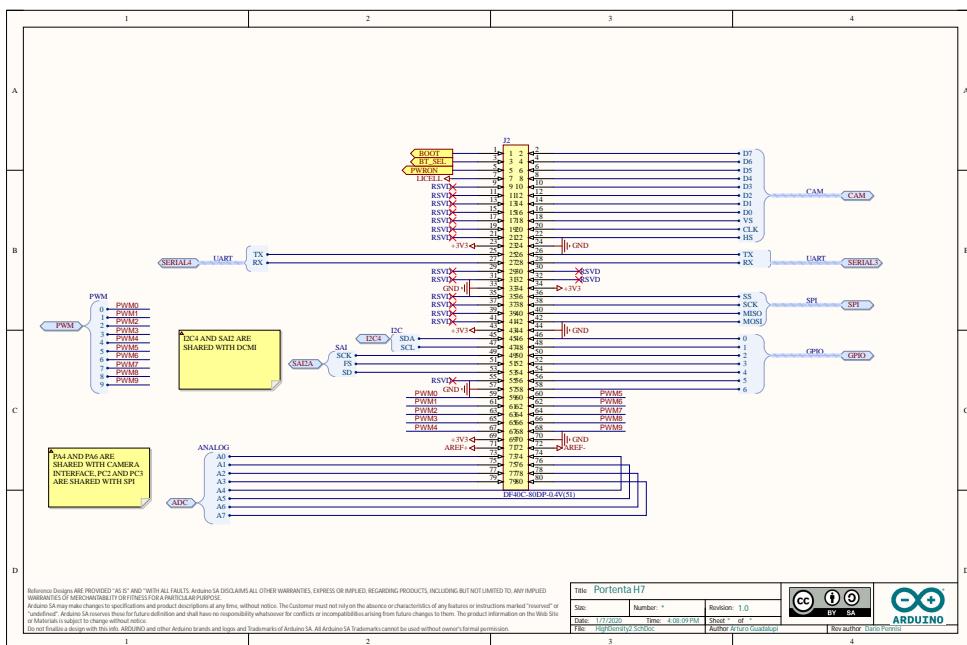


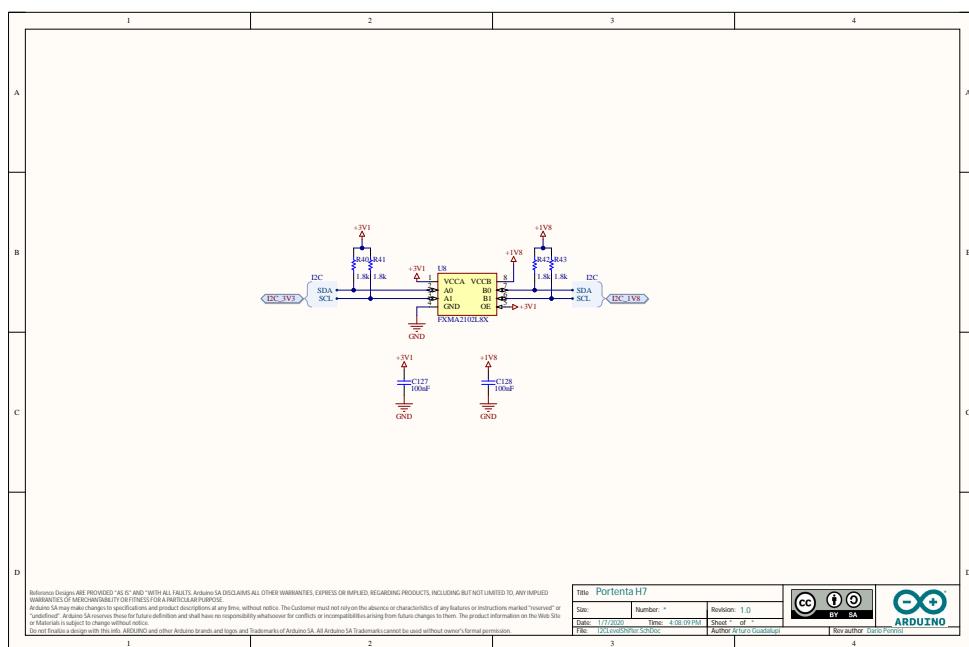


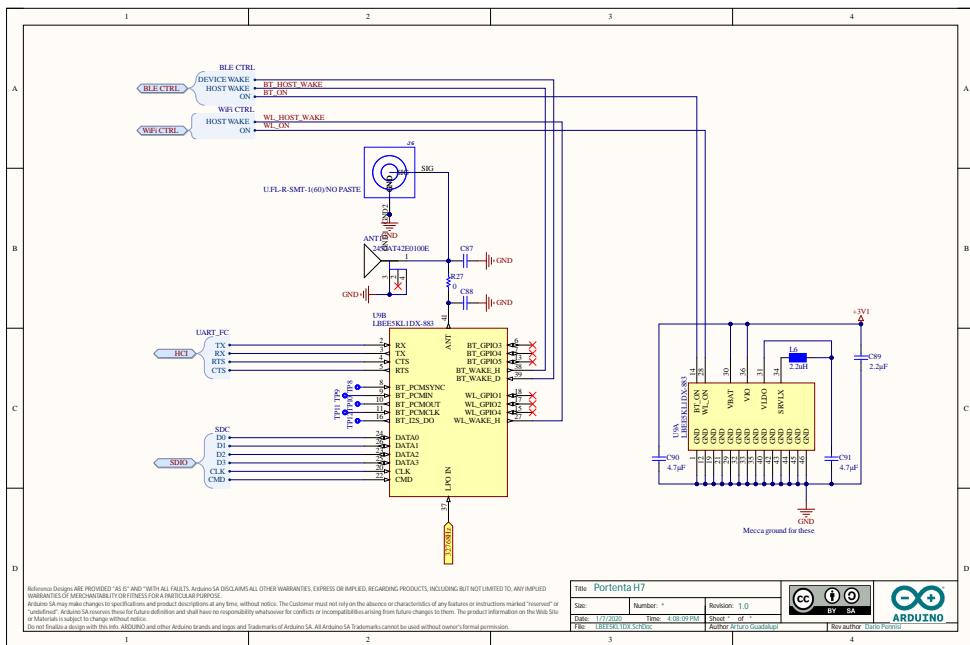




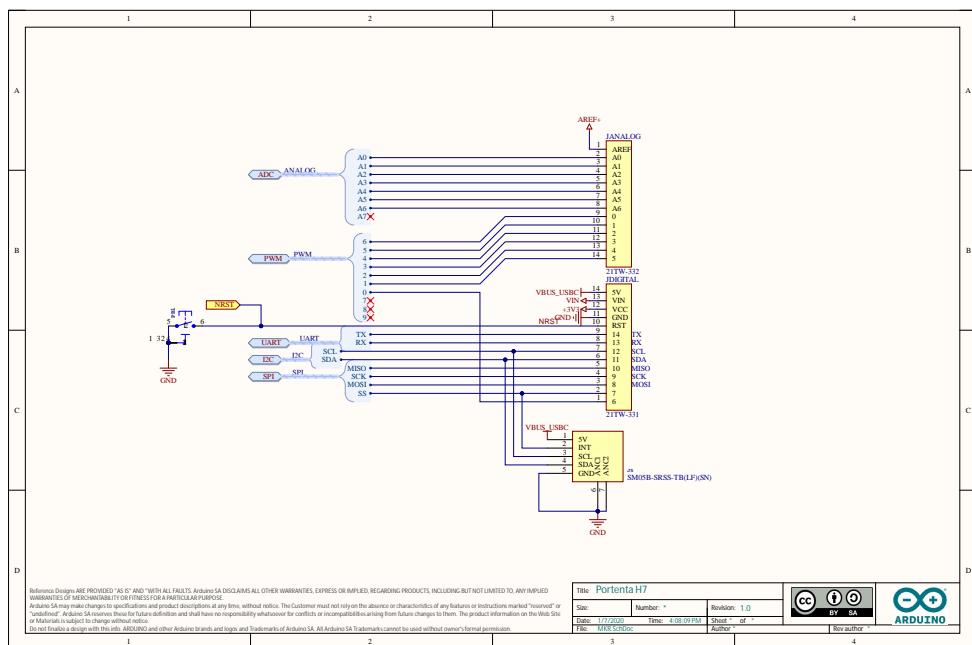


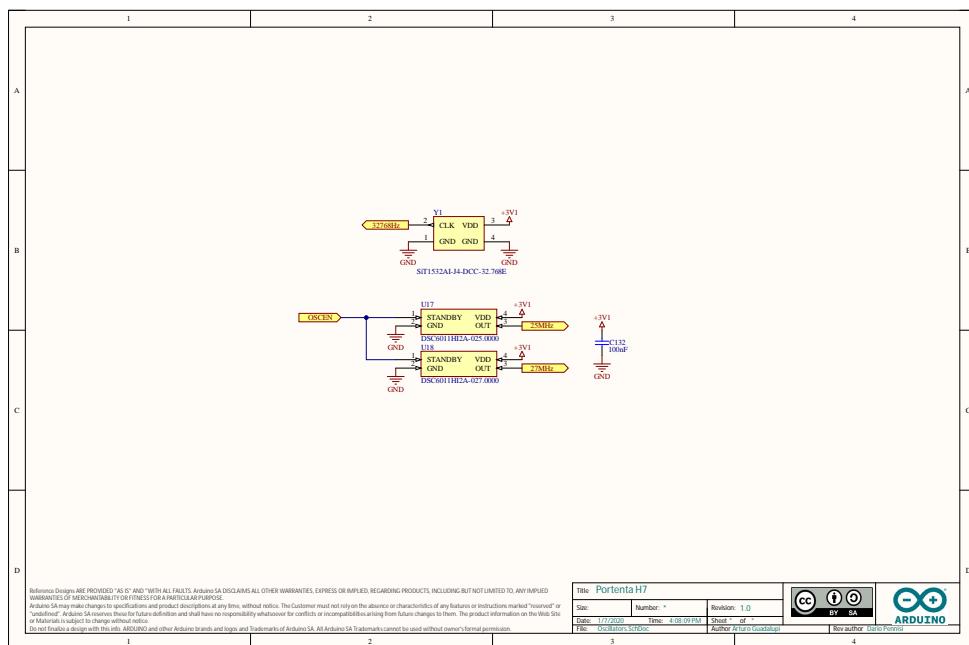


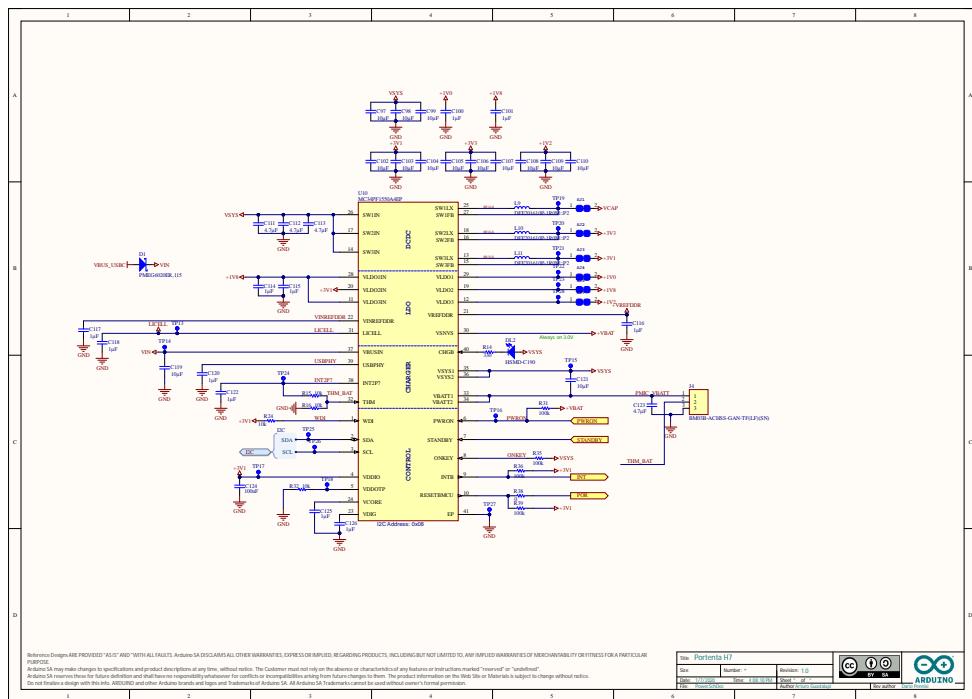


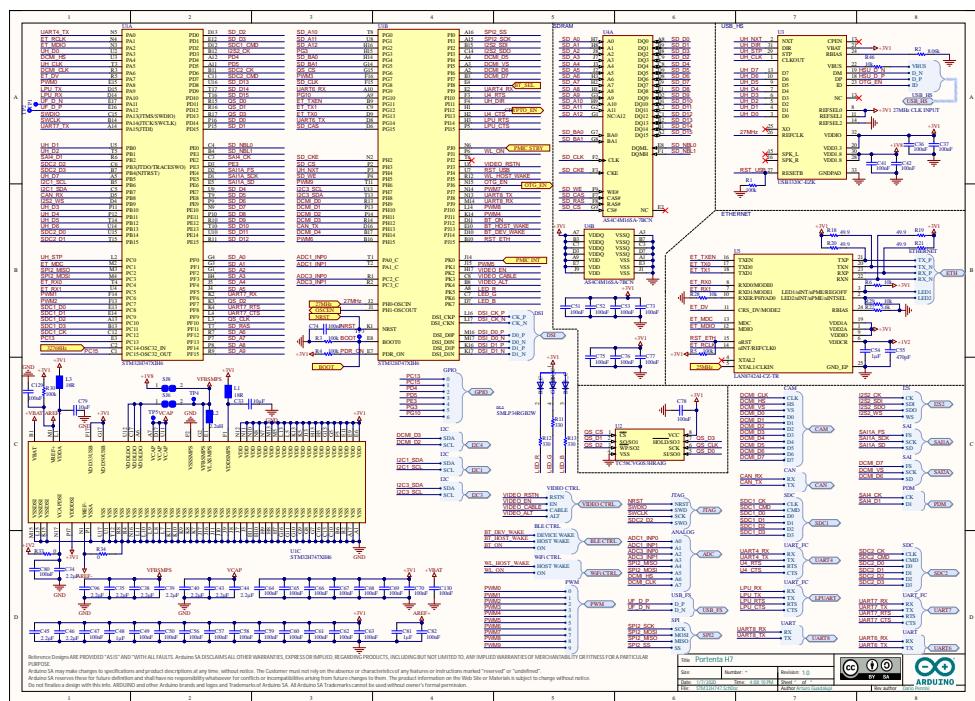


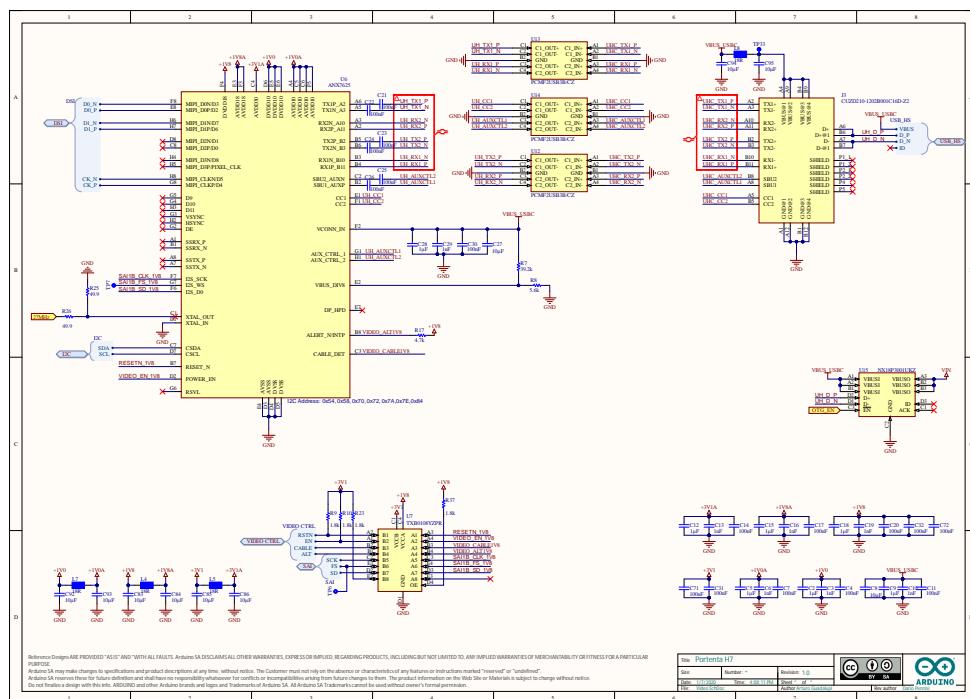
Title: Portenta H7		CC BY SA	ARDUINO
Ref.:	Number: 1.0	Revision: 1.0	
Date: 10/27/2020	Time: 4:08:09 PM	Author: Arturo Guida	Rev author: Mario Peroni
FIC: IEEE100X5762			











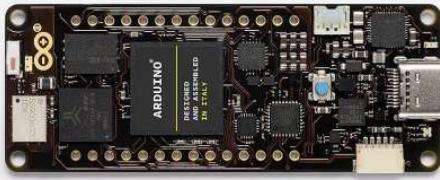
Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino SA DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Arduino SA may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Arduino SA reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes. The product information on the Web Site or Material is subject to change without notice. Do not feature a design with this info. Arduino and other Arduino brand and logo trademarks of Arduino SA. All Arduino SA trademarks cannot be used without written permission. Do not feature a design with this info. Arduino and other Arduino brand and logo trademarks of Arduino SA. All Arduino SA trademarks cannot be used without written permission.

Rev.	PartNumber	Number	Revision	I.D.
1.0	MAX98354	100-10100	1.0	ARDUINO



PORTENTA H7

Code: 7630049202252



WiFi | 32 bit | ARM | 100 mA | 3.3V | Large (~70) | LiPo

The Portenta H7 will be ready for shipment approximately at the end of March 2020.

Program it with high-level languages and AI while performing low-latency operations on its customizable hardware.

Portenta H7 simultaneously runs high level code along with real time tasks. The design includes two processors that can run tasks in parallel. For example, is possible to execute Arduino compiled code along with MicroPython one, and have both cores to communicate with one another. The Portenta functionality is two-fold, it can either be running like any other embedded microcontroller board, or as the main processor of an embedded computer. Use the [Portenta Carrier board](#) to transform your H7 into an eNUC computer and expose all of the H7 physical interfaces.

Portenta can easily run processes created with TensorFlow™ Lite, you could have one of the cores computing a computer vision algorithm on the fly, while the other could be making low level operationis like controlling a motor, or acting as a user interface.

Use Portenta when performance is key, among other cases, we envision it to be part of:

- High-end industrial machinery
- Laboratory equipment
- Computer vision
- PLCs
- Industry-ready user interfaces
- Robotics controller
- Mission-critical devices
- Dedicated stationary computer
- High-speed booting computation (ms)

Two Parallel Cores

H7's main processor is the dual core STM32H747 including a Cortex® M7 running at 480 MHz and a Cortex® M4 running at 240 MHz. The two cores communicate via a *Remote Procedure Call* mechanism that allows calling functions on the other processor seamlessly. Both processors share all the in-chip peripherals and can run:

- Arduino sketches on top of the Arm® Mbed™ OS
- Native Mbed™ applications
- MicroPython / JavaScript via an interpreter
- TensorFlow™ Lite

Graphics Accelerator

Probably one of the most exciting features of the Portenta H7 is the possibility of connecting an external monitor to build your own dedicated embedded computer with a user interface. This is possible thanks to the STM32H747 processor's on-chip GPU, the Chrom-ART Accelerator™. Besides the GPU, the chip includes a dedicated JPEG encoder and decoder.

A new standard for pinouts

The Portenta family adds **two 80 pin high density connectors** at the bottom of the board. This ensures scalability for a wide range of applications by simply upgrading your Portenta board to the one suiting your needs.

On-board Connectivity

The onboard wireless module allows to simultaneously manage WiFi and Bluetooth® connectivity. The WiFi interface can be operated as an Access Point, as a Station or as a dual mode simultaneous AP/STA and can handle up to 65 Mbps transfer rate. Bluetooth® interface supports Bluetooth Classic and BLE. It is also possible to expose a series of different wired interfaces like UART, SPI, Ethernet, or I2C, both through some of the MKR styled connectors, or through the new Arduino industrial 80 pin connector pair.

USB-C Multipurpose Connector

The board's programming connector is a USB-C port that can also be used to power the board, as a USB Hub, to connect a DisplayPort monitor, or to deliver power to OTG connected devices.

Multiple options in one board

Order the default Arduino Portenta H7 (codename H7-15EUNWAD) that comes with:

- STM32H747 dual-core processor with graphics engine
- 8MB SDRAM
- 16MB NOR Flash
- 10/100 Ethernet Phy
- USB HS
- NXP SE050C2 Crypto
- WiFi/BT Module
- Ceramic Antenna
- DisplayPort over USB-C

If you need more memory, Portenta H7 can host up to 64 MByte of SDRAM, and 128 MByte of QSPI Flash. Order it with an external UFL connector for adding a higher-gain antenna to the board. Decide between crypto-chips from Microchip® and NXP. The board is highly customizable in volumes, ask our sales representatives for options.

The basic configurations you can consider to get the board to accommodate to your needs and budget are:

Option	Description	Option Codes
SDRAM	external SDRAM memory	0 - None 1 - 8 MByte 2 - 16 MByte 3 - 32 MByte 4 - 64 MByte
FLASH	external QSPI Flash Memory	0 - None 1 - 2 MByte (NOR) 5 - 16 MByte (NOR) 8 - 128 MByte (NAND)
Ethernet	10/100 Ethernet PHY	0 - None E - Fitted
HS USB	High Speed USB PHY	0 - None U - Fitted
Crypto	Crypto Chip	0 - None M - ATECC608A N - SE050C2
Wireless	Wireless Module	0 - None W - Fitted
Antenna	Antenna option	0 - None A - on board ceramic antenna C - UFL connector
Video	Displayport output over USB-C	0 - None D - Fitted

Need Help?

Check the Arduino Forum for questions about the [Arduino Language](#), or how to make your own [Projects with Arduino](#). Need any help with your board please get in touch with the official Arduino User Support as explained in our [Contact Us](#) page.

Warranty

You can find [here](#) your board warranty information.

<https://www.arduino.cc/en/Main/warranty>

The Arduino Portenta H7 is based on the STM32H747 microcontroller, XI series.

Microcontroller	STM32H747XI dual Cortex®-M7+M4 32bit low power ARM MCU (datasheet)
Radio module	Murata 1DX dual WiFi 802.11b/g/n 65 Mbps and Bluetooth 5.1 BR/EDR/LE (datasheet)
Secure Element (default)	NXP SE0502 (datasheet)
Board Power Supply (USB/VIN)	5V
Supported Battery	Li-Po Single Cell, 3.7V, 700mAh Minimum (integrated charger)
Circuit Operating Voltage	3.3V
Current Consumption	2.95 µA in Standby mode (Backup SRAM OFF, RTC/LSE ON)
Display Connector	MIPI DSI host & MIPI D-PHY to interface with low-pin count large display
GPU	Chrom-ART graphical hardware Accelerator™

Timers	22x timers and watchdogs
UART	4x ports (2 with flow control)
Ethernet PHY	10 / 100 Mbps (through expansion port only)
SD Card	Interface for SD Card connector (through expansion port only)
Operational Temperature	-40 °C to +85 °C (excl. Wireless module) / -10 °C to +55 °C (incl. Wireless module)
MKR Headers	Use any of the existing industrial MKR shields on it
High-density Connectors	Two 80 pin connectors will expose all of the board's peripherals to other devices
Camera Interface	8-bit, up to 80 MHz
ADC	3x ADCs with 16-bit max. resolution (up to 36 channels, up to 3.6 MSPS)
DAC	2x 12-bit DAC (1 MHz)
USB-C	Host / Device, DisplayPort out, High / Full Speed, Power delivery

OSH: Schematics

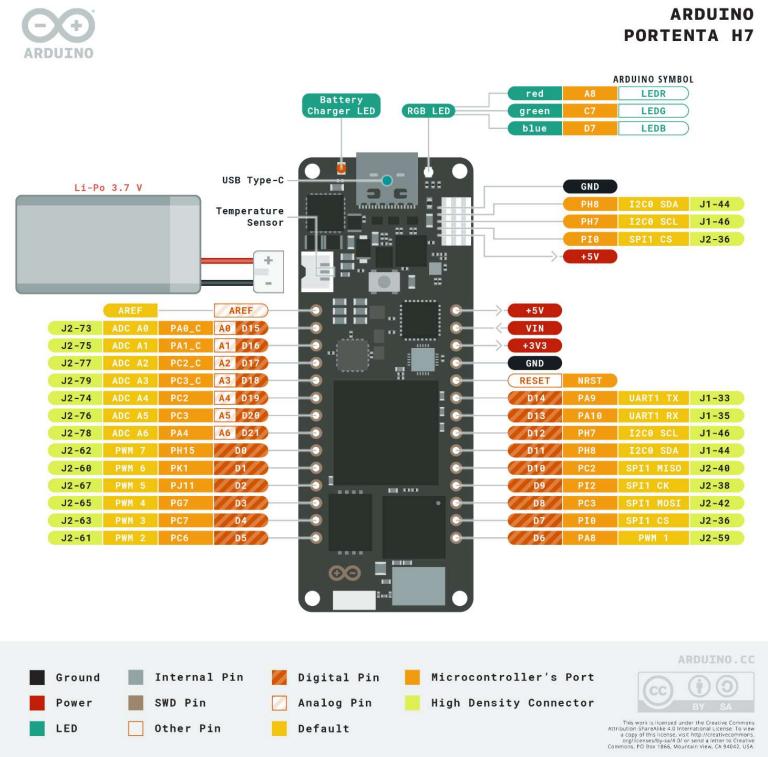
The Portenta H7 is open-source hardware! You can study how the board works using the following files:

[SCHEMATICS IN .PDF](#)

<https://content.arduino.cc/assets/Arduino-PortentaH7-schematic-V1.0.pdf>

Pinout Diagram

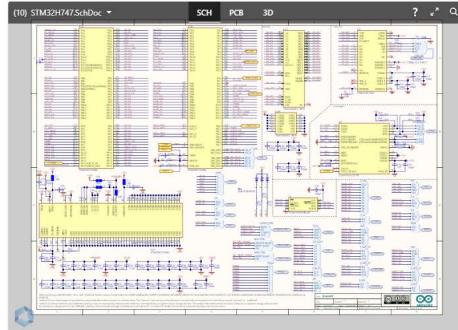
The Portenta H7 follows the Arduino MKR form factor, but enhanced with the Portenta family 80 pin high-density connector. Learn more about the board's pinout by reading the board's pinout documentation.



Download the full pinout diagram as PDF [here](#).

https://content.arduino.cc/assets/Pinout-PortentaH7_latest.pdf

Interactive Board Viewer

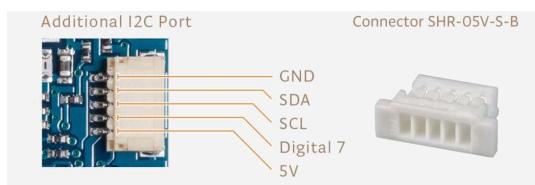


Additional I2C Port

The Portenta H7 has an additional connector meant as an extension of the I2C bus. It's a small form factor 5-pin connector with 1.0 mm pitch. The mechanical details of the connector can be found in the [connector's datasheet](#).

https://content.arduino.cc/assets/ESLOV_connector.pdf

The I2C port, also referred to as the Eslov self-identification port within Arduino, comes with: SDA, SCL, GND, +5V, and an extra digital pin meant to send an alarm to the otherwise plain I2C devices connected to it. The pinout is shown in the following image:



If you are interested in designing your own modules for Arduino boards with this expansion port, the connector we suggest using is code: SHR-05V-S-B, also in the picture.



<https://store.arduino.cc/usa/portenta-h7/3-13-20>

Portenta H7

A new board designed for high performance, meet Portenta H7

Program it with high-level languages and AI while performing low-latency operations on its customizable hardware

TUTORIAL | PYTHON | JAVASCRIPT | DISPLAYPORT | WiFi

TWO PARALLEL CORES

Portenta H7 simultaneously runs high level code along with real time tasks

H7's main processor is the dual core **ST32MP1F4** featuring a Cortex® M7 running at 480 MHz and a Cortex® M4 running at 240 MHz. The two cores communicate via a Remote Procedure Call mechanism that allows calling functions on the other processor seamlessly.

Both processors share all the in-chip peripherals and can run:

- Arduino sketches on top of the Arm® Mbed™ OS
- Native MicroPython
- MicroPython / JavaScript via an interpreter
- TensorFlow™ Lite

The onboard wireless module allows to simultaneously manage WiFi and Bluetooth® connectivity. The WiFi interface can be operated as an Access Point, as a Station or as a dual mode simultaneous AP/STA and can handle up to 65 Mbps transfer rate. Bluetooth® interface supports Bluetooth Classic and BLE.

The Portenta H7 follows the Arduino MKR form factor, but enhanced with the Portenta family 80 pin high-density connector. Learn more about the board's pinout by reading the board's pinout documentation.

USE PORTENTA WHEN PERFORMANCE IS KEY

- High-end industrial machinery
- Laboratory equipment
- Computer vision
- PLCs
- Industry-ready user interfaces
- Robotics controller
- Mission-critical devices
- Dedicated stationary computer
- High-speed booting computation (mcu)

A NEW STANDARD FOR PINOUTS

The Portenta family adds two 80 pin high density connectors at the bottom of the board. This ensures scalability for a wide range of applications by simply upgrading your Portenta board to the one suiting your needs.

DUAL CORE
Two processors in one, run parallel tasks

CUSTOMIZATION
The board is highly customizable in volumes

DEDICATED COMPUTER
Power your board, connect it to a display, implement WiFi

AT ON THE EDGE
So powerful it can run AI tasks machine

Multiple options in one board

By default the Arduino Portenta H7 comes with:

- ST32MP1F4 dual core processor with graphics engine
- 16MB SDRAM
- 16MB NOR Flash
- 10/100 Ethernet Phy
- USB HS
- NOR ST505C2 Crypto
- WiFi/BT Module
- Ceramic Antenna
- DisplayPort over USB-C

Tailor the hardware to your needs

If you need more memory, Portenta H7 can host up to 64 Mbytes of SDRAM and 128 Mbytes of QSPI Flash. Order it with an external WiFi connector for adding a higher gain antenna to the board. Decide between crypto-chips from Microchip® and NXP®

PORTENTA H7 TECH SPECS	
MAIN PROCESSOR	STM32H747XI dual Cortex®-M7+M4 32bit low power Arm® MCU
SDRAM	8-64 MByte option
QSPI FLASH	2-128 MByte option
ETHERNET	10/100 Phy option
WIRELESS	BT5.0 + WiFi 802.11 b/g/n 65Mbps option
CRYPTO CHIP	ECC608 or SE050C2 (Common Criteria EAL 6+) option
DISPLAY CONNECTOR	MIPI DSI host & MIPI D-PHY to interface with low-pin count large displays
GPU	Chrom-ART graphical hardware Accelerator™
TIMERS	22x timers and watchdogs
UART	4x ports (2 with flow control)
SD CARD	Interface for SD Card connector (through expansion port only)
OPERATIONAL TEMPERATURE	-40 °C to +85 °C (excl. Wireless module) / -10 °C to +55 °C (incl. Wireless module)
POWER	Through USB-C connector or LiPo battery (integrated charger)
CURRENT CONSUMPTION	2.95 µA in Standby mode (Backup SRAM OFF, RTC/LSE ON)
USB-C	Host / Device, DisplayPort out, High / Full Speed, Power delivery
MKR HEADERS	Use any of the existing industrial MKR shields on it
HIGH DENSITY CONNECTORS	Two 80 pin connectors will expose all of the board's peripherals to other devices
ESLOV CONNECTOR	Arduino's open connector standard for self-identifiable hardware
CAMERA INTERFACE	8-bit, up to 80 MHz
ADC	3x ADCs with 16-bit max. resolution (up to 36 channels, up to 3.6 MSPS)
DAC	2x 12-bit DAC (1 MHz)

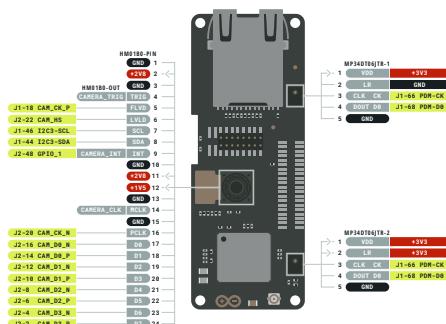
BACK TO TOP ^

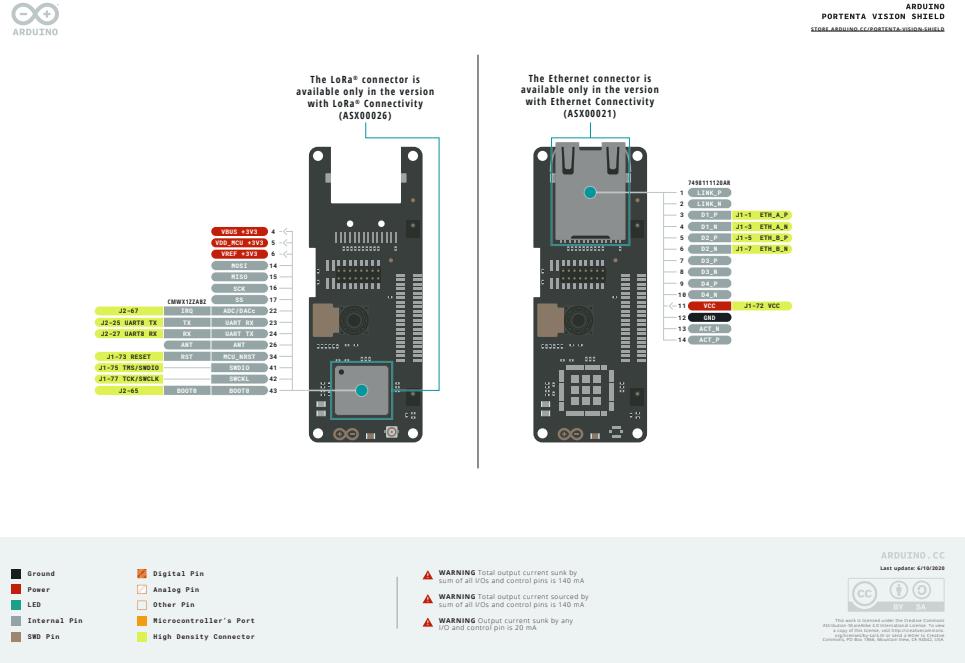
<https://www.arduino.cc/pro/hardware/product/portenta-h7>

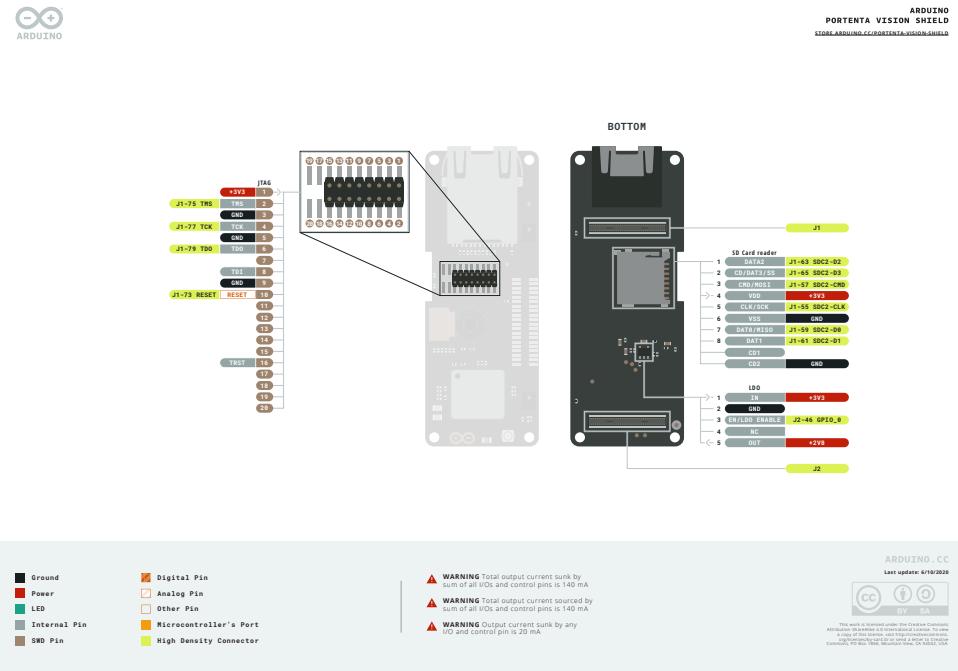
34. Datenblätter Arduino Vision Shield



ARDUINO
PORTENTA VISION SHIELD
[STORE ARDUINO CC/PORTENTA-VISION-SHIELD](https://store.arduino.cc/portenta-vision-shield)









Arduino® Vision Shield

Product Datasheet
SKU (Ethernet): ASX00021
SKU (LoRa): ASX00026

Description:

The Arduino Portenta Vision Shield is an addon board providing machine vision capabilities and additional connectivity to the Portenta family of Arduino boards, designed to meet the needs of industrial automation. The Portenta Vision Shield connects via a high density connector to the Portenta H7 with minimal hardware and software setup.



Target areas:
Industry, surveillance

Features

Note: This board needs the Arduino Portenta H7 to function

- **Himax HM-01B0 camera module**
 - Ultra Low Power Image Sensor designed for Always-on vision devices and applications
 - High sensitivity 3.6 μ BrightSense™ pixel technology
 - window, vertical flip and horizontal mirror readout
 - Programmable black level calibration target, frame size, frame rate, exposure, analog gain (up to 8x) and digital gain (up to 4x)
 - Automatic exposure and gain control loop with support for 50Hz / 60Hz flicker avoidance
 - Motion Detection circuit with programmable ROI and detection threshold with digital output to serve as an interrupt
- **Supported resolutions**
 - QQVGA (160x120) at 15, 30, 60 and 120 FPS
 - QVGA (320x240) at 15, 30 and 60 FPS
- **Power**
 - <1.1mW QQVGA resolution at 30FPS,
 - <2mW QVGA resolution at 30FPS
- **2x MP34DT06jTR MEMS PDM Digital Microphone**
 - AOP = 122.5 dB SPL
 - 64 dB signal-to-noise ratio
 - Omnidirectional sensitivity
 - -26 dBFS ± 1 dB sensitivity
- **MIPI 20 pin compatible JTAG Connector**
- **Memory**
 - Micro SD Card Slot



Contents

1. The board	4
Application examples	4
Related Products	4
2. Ratings	4
2.1 Absolute Maximum	4
2.2 Thermal	4
3. Functional Overview	5
3.1 Board Topology	5
3.2 Camera module	6
3.3 Digital Microphones	6
3.3 Micro SD Card Slot	6
3.4 Ethernet (ASX00021 only)	6
3.5 LoRa module (ASX00026 only)	6
3.6 Power	7
4. Board Operation	7
4.1 Getting Started – IDE	7
4.2 Getting Started – Arduino Web Editor (Create)	7
4.3 Getting Started – Arduino IoT Cloud	7
4.4 Getting Started – OpenMV	7
4.5 Online resources	8
4.6 Board Recovery	8
5. Connector Pinouts	9
5.1 JTAG	9
5.2 High Density Connector	10
6. Mechanical Information	11
6.1 Board Outline	11
6.2 Mounting holes	11



6.3 Connector and component positions	12
6.4 Mounting Instructions	13
7. Certifications	14
7.1 Declaration of Conformity CE/RED DoC (EU)	14
7.2 Declaration of Conformity to EU RoHS & REACH 191 11/26/2018	14
7.3 Conflict Minerals Declaration	14
8. Reference Documentation	16
9. Revision History	17



1. The board

The included HM-01B0 camera module has been pre-configured to work with the OpenMV libraries provided by Arduino. Based on the specific application requirements, the Portenta Vision Shield is available in two configurations with either Ethernet or LoRa connectivity. Ethernet is designed for integration of the Portenta into wired networks and providing high bandwidth. In situations requiring long range operation at low bandwidth, LoRa connectivity is the way to go. The multicore processor of the Portenta H7 makes embedded vision possible by minimising the data bandwidth required.

Note: The Portenta Vision Shield is available in two SKU, Ethernet (ASX00021) and LoRa (ASX00026)

1.1 Application examples

Thanks to the low power consumption of the Vision Shield, it is well suited for bringing machine learning to a wide range of Industry 4.0 and IoT applications.

Industrial production: The included HM-01B0 camera along with the OpenMV libraries allows for quality control of items within a manufacturing or packaging plant. The small footprint, low power consumption and LoRa/Ethernet connectivity allows for the module to be deployed essentially anywhere so that defects are identified quickly and removed from the production environment.

Predictive maintenance: The combination of machine vision and machine learning capabilities of the Vision Shield and the Portenta H7 opens up possibilities for predictive maintenance based on subtle differences in the visual representation of machinery. These capabilities are further enhanced with the two MP34DT05 MEMS microphones included in the Vision Shield.

Surveillance: The Vision Shield is able to provide surveillance capabilities in areas with low WiFi penetration (e.g. warehouse) and large areas (e.g. shopping centres). The OpenMV libraries enable the Vision Shield to identify objects and alert the operator via LoRa while saving a snapshot on the microSD storage slot.

1.2 Related Products

The Vision Shield is developed as an add-on shield that requires the Portenta H7.

2. Ratings

2.1 Absolute Maximum

Symbol	Description	Min	Typ	Max	Unit
VIN _{Max}	Input voltage from HD Connectors	-0.3	-	3.3	V
P _{Max}	Maximum Power Consumption	-	-	TBC	mW

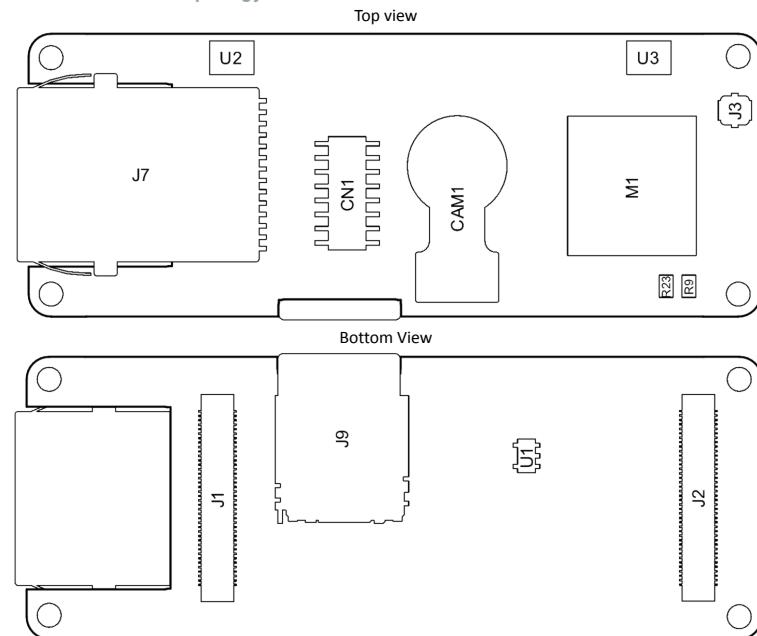
2.2 Thermal

Symbol	Description	Min	Typ	Max	Unit
T _{ST}	Storage Temperature	-30		85	°C
T _{OP}	Operating Temperature	-20		85	°C



3. Functional Overview

3.1 Board Topology



Ref.	Description	Ref.	Description
U1	Voltage Regulator	J3	LoRa Radio Antenna U.FL Connector (ASX00026 Only)
U2,U3	ST MP34DT06JTR Digital Microphone	J7	Ethernet Connector (ASX00021 Only)
M1	Murata CMWX1ZZABZ LoRa Module (ASX00026 Only)	J9	Micro SD Card Connector
J1,J2	High Density Connectors	CN1	JTAG Connector



3.2 Camera module

The Himax HM-01B0 Module is a very low power camera with 324x324 resolution and a maximum of 60FPS depending on the operating mode. Video data is transferred over a configurable 8-bit interconnect with support for frame and line synchronisation. The module delivered with the Vision Shield is the monochrome version. Configuration is achieved via a I2C connection with the Portenta H7.

HM-01B0 offers very low power image acquisition and provides the possibility to perform motion detection without main processor interaction. "Always-on" operation provides the ability to turn on the main processor when movement is detected with minimal power consumption.

3.3 Digital Microphones

The dual MP34DT05 digital MEMS microphones are omnidirectional and operate via a capacitive sensing element with a high (64 dB) signal to noise ratio. The microphones have been configured to provide separate left and right audio over a single PDM stream.

The sensing element, capable of detecting acoustic waves, is manufactured using a specialized silicon micromachining process dedicated to produce audio sensors

3.3 Micro SD Card Slot

A micro SD card slot is available under the Vision Shield board. Available libraries allow reading and writing to FAT16/32 formatted cards.

3.4 Ethernet (ASX00021 only)

Ethernet connector allows connecting to 10/100 Base TX networks using the Ethernet PHY available on the Portenta board.

3.5 LoRa module (ASX00026 only)

LoRa connectivity is provided by the Murata CMWX1ZZABZ module. This module contains a STM32L0 processor along with a Semtech SX1276 Radio. The processor is running on Arduino open source firmware based on Semtech code.



3.6 Power

The Portenta H7 supplies 3.3V power to the LoRa module (ASX00026 only), microSD slot and dual microphones via the 3.3V output via the high density connector. An onboard LDO regulator supplies a 2.8V output (300mA) for the camera module.

4. Board Operation

4.1 Getting Started – IDE

If you want to program your Arduino board while offline you need to install the Arduino Desktop IDE [1] To connect the board to your computer, you'll need a USB cable. This also provides power to the board, as indicated by the LED.

4.2 Getting Started – Arduino Web Editor (Create)

All Arduino and Genuino boards, including this one, work out-of-the-box on the Arduino Web Editor [2], by just installing a simple plugin.

The Arduino Web Editor is hosted online, therefore it will always be up-to-date with the latest features and support for all boards. Follow [3] to start coding on the browser and upload your sketches onto your board.

4.3 Getting Started – Arduino IoT Cloud

All Arduino IoT enabled products are supported on Arduino IoT Cloud which allows you to Log, graph and analyze sensor data, trigger events, and automate your home or business.

4.4 Getting Started – OpenMV

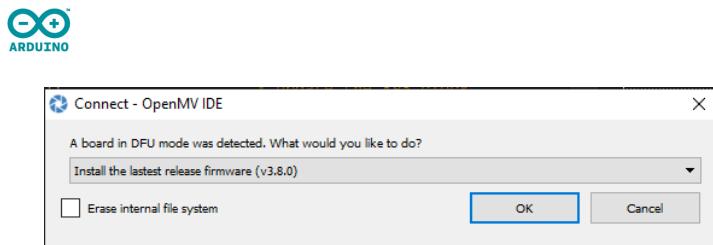
NOTE!

It is highly recommended that you ensure you have the latest bootloader on your Portenta H7 before loading OpenMV firmware.

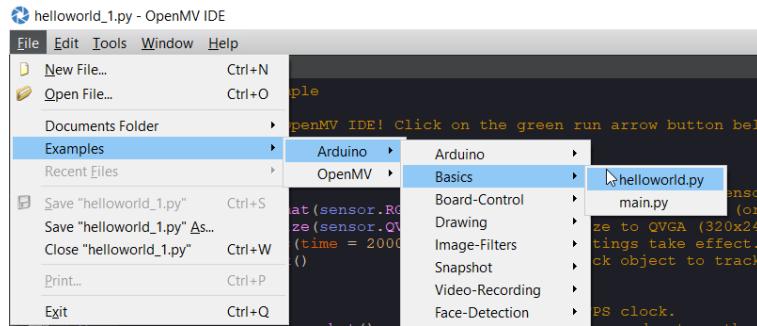
Arduino Vision Shield and Portenta H7 are supported under OpenMV. In order to easily use OpenMV download the latest OpenMV IDE [5] set up Portenta H7 in boot mode by double tapping reset and connect via the connection button.



Once connected you will receive a message like the following:



Click on "OK" and the latest OpenMV firmware will be automatically loaded. To open the "Hello World" example, under the *File* menu select *Examples* -> *Arduino* -> *Basics* and click on *helloworld.py*.



Click on the green square underneath the connection button to run.



4.5 Online resources

Now that you have gone through the basics of what you can do with the board you can explore the endless possibilities it provides by checking exciting projects on ProjectHub [6], the Arduino Library Reference [7] and the online store [8] where you will be able to complement your board with sensors, actuators and more.

4.6 Board Recovery

All Arduino boards have a built-in bootloader which allows flashing the board via USB. In case a sketch locks up the processor and the board is not reachable anymore via USB it is possible to enter bootloader mode by double-tapping the reset button right after power up.



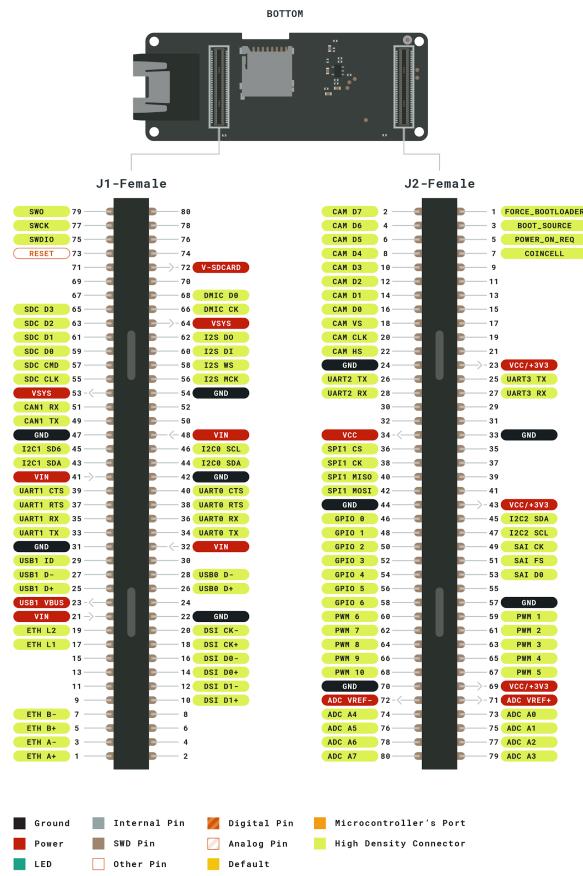
5. Connector Pinouts

5.1 JTAG

Pin	Function	Type	Description
1	VDDIO	Power	Positive Reference voltage for debug interface
2	SWD	I/O	Single Wire Debug Data
3,5,9	GND	Power	Negative reference voltage for debug interface
4	SCK	Output	Single Wire Debug Clock
6	SWO	I/O	Single Wire Debug Trace
10	RESET	Input	CPU Reset
7,11,12,13, 14,15,17,1 8,19,20	NC	Not Connected	



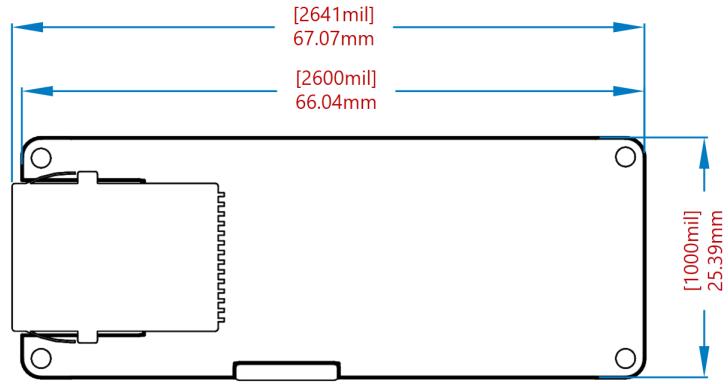
5.2 High Density Connector



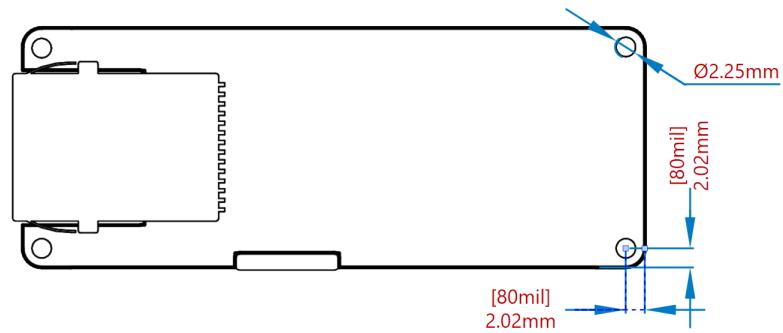


6. Mechanical Information

6.1 Board Outline

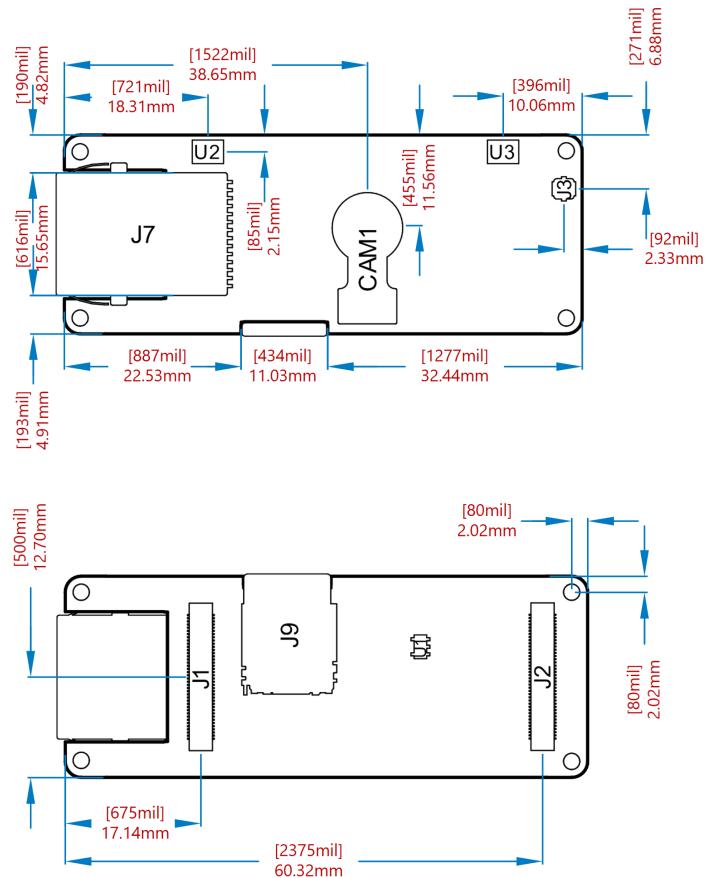


6.2 Mounting holes



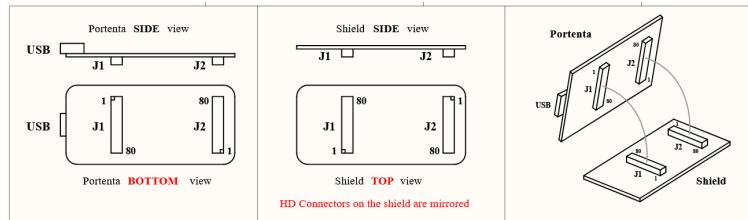


6.3 Connector and component positions





6.4 Mounting Instructions





7. Certifications

7.1 Declaration of Conformity CE/RED DoC (EU)

We declare under our sole responsibility that the products above are in conformity with the essential requirements of the following EU Directives and therefore qualify for free movement within markets comprising the European Union (EU) and European Economic Area (EEA).

ROHS 2 Directive 2011/65/EU

Conforms to: EN50581:2012

Directive 2014/35/EU. (LVD)

Conforms to: EN 60950-1:2006/A11:2009/A1:2010/A12:2011/AC:2011

Directive 2004/40/EC & 2008/46/EC & 2013/35/EU, EMF

Conforms to: EN 62311:2008

7.2 Declaration of Conformity to EU RoHS & REACH 191 11/26/2018

Arduino boards are in compliance with Directive 2011/65/EU of the European Parliament and Directive 2015/863/EU of the Council of 4 June 2015 on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

Substance	Maximum Limit (ppm)
Lead (Pb)	1000
Cadmium (Cd)	100
Mercury (Hg)	1000
Hexavalent Chromium (Cr6+)	1000
Poly Brominated Biphenyls (PBB)	1000
Poly Brominated Diphenyl ethers (PBDE)	1000
Bis(2-Ethylhexyl) phthalate (DEHP)	1000
Benzyl butyl phthalate (BBP)	1000
Dibutyl phthalate (DBP)	1000
Diisobutyl phthalate (DIBP)	1000

Exemptions : No exemptions are claimed.

Arduino Boards are fully compliant with the related requirements of European Union Regulation (EC) 1907 /2006 concerning the Registration, Evaluation, Authorization and Restriction of Chemicals (REACH). We declare none of the SVHCs (<https://echa.europa.eu/web/guest/candidate-list-table>), the Candidate List of Substances of Very High Concern for authorization currently released by ECHA, is present in all products (and also package) in quantities totaling in a concentration equal or above 0.1%. To the best of our knowledge, we also declare that our products do not contain any of the substances listed on the "Authorization List" (Annex XIV of the REACH regulations) and Substances of Very High Concern (SVHC) in any significant amounts as specified by the Annex XVII of Candidate list published by ECHA (European Chemical Agency) 1907 /2006/EC.

7.3 Conflict Minerals Declaration

As a global supplier of electronic and electrical components, Arduino is aware of our obligations with regards to laws and regulations regarding Conflict Minerals, specifically the Dodd-Frank Wall Street Reform and Consumer Protection Act, Section 1502. Arduino does not directly source or process conflict minerals such as Tin, Tantalum, Tungsten, or Gold. Conflict minerals are contained in our products in the form of solder, or as a



component in metal alloys. As part of our reasonable due diligence Arduino has contacted component suppliers within our supply chain to verify their continued compliance with the regulations. Based on the information received thus far we declare that our products contain Conflict Minerals sourced from conflict-free areas.



8. Reference Documentation

Ref	Link
1. Arduino IDE (Desktop)	https://www.arduino.cc/en/Main/Software
2. Arduino IDE (Cloud)	https://create.arduino.cc/editor
3. Cloud IDE Getting Started	https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-arduino-web-editor-4b3e4a
4. Forum	http://forum.arduino.cc/
5. OpenMV IDE	https://openmv.io/pages/download
6. ProjectHub	https://create.arduino.cc/projecthub?by=part&part_id=11332&sort=trending
7. Library Reference	https://www.arduino.cc/reference/en/
8. Arduino Store	https://store.arduino.cc/



9. Revision History

Date	Revision	Changes
03/03/2021	1	First Release

Literaturverzeichnis

- [ABF11] L. Akremi, N. Baur, and S. Fromm, eds. *Datenanalyse mit SPSS für Fortgeschrittene 1 – Datenaufbereitung und uni- und bivariate Statistik*. Vol. 3. Springer Fachmedien, 2011.
- [AIT20] L. Artificial Intelligence Techniques, ed. *OpenNN*. 2020. URL: <https://www.opennn.net/>.
- [AR+16] R. Al-Rfou et al. “Theano: A Python framework for fast computation of mathematical expressions”. In: *arXiv e-prints* abs/1605.02688 (May 2016). [pdf], [Link]. URL: <http://arxiv.org/abs/1605.02688>.
- [Aba+16] M. Abadi et al. “TensorFlow: A System for Large-Scale Machine Learning”. In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, Nov. 2016, pp. 265–283. ISBN: 978-1-931971-33-1. URL: <https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi>.
- [Aga18a] A. F. M. Agarap. “On Breast Cancer Detection: An Application of Machine Learning Algorithms on the Wisconsin Diagnostic Dataset”. In: *Proceedings of the 2nd International Conference on Machine Learning and Soft Computing*. ICMLSC ’18. Link, Link. Phu Quoc Island, Viet Nam: Association for Computing Machinery, 2018, pp. 5–9. ISBN: 9781450363365. DOI: [10.1145/3184066.3184080](https://doi.org/10.1145/3184066.3184080). URL: <https://doi.org/10.1145/3184066.3184080>.
- [Aga18b] A. F. Agarap. “Deep Learning using Rectified Linear Units (ReLU)”. In: (Mar. 22, 2018). DOI: [10.48550/ARXIV.1803.08375](https://doi.org/10.48550/ARXIV.1803.08375). arXiv: [1803.08375](https://arxiv.org/abs/1803.08375) [cs.NE]. URL: <https://arxiv.org/pdf/1803.08375v2.pdf>.
- [Ala08] R. Alake. *Implementing AlexNet CNN Architecture Using TensorFlow 2.0+ and Keras*. 14.08.2020. URL: <https://towardsdatascience.com/implementing-alexnet-cnn-architecture-using-tensorflow-2-0-and-keras-2113e090ad98>.
- [Alo+18] M. Z. Alom et al. *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches*. Tech. rep. [pdf]. 2018. arXiv: [1803.01164](https://arxiv.org/abs/1803.01164) [cs.CV].
- [And35] E. Anderson. “The irises of the Gaspé Peninsula”. In: *Bulletin of the American Iris Society* 59 (1935), pp. 2–5.
- [Ard] Arduino CLI 0.33. 2018. URL: <https://arduino.github.io/arduino-cli/0.33/commands/arduino-cli/> (visited on 07/24/2023).
- [Ard20] Arduino, ed. *Arduino CLI: An Introduction*. 2020. URL: <https://blog.arduino.cc/2020/03/13/arduino-cli-an-introduction>.
- [Ard21a] Arduino, ed. *Arduino Portenta H7*. 2021. URL: <https://www.arduino.cc/pro/hardware/product/portenta-h7>.
- [Ard21b] Arduino, ed. *Arduino Portenta Vision Shield*. 2021. URL: <https://store.arduino.cc/portenta-vision-shield>.
- [Ard22a] Arduino, ed. *Arduino CLI (Command Line Interface) Application*. 2022. URL: <https://github.com/arduino/arduino-cli>.

-
- [Ard22b] Arduino, ed. *Arduino CLI*. 2022. URL: <https://arduino.github.io/arduino-cli/0.22/>.
- [Ard22c] Arduino, ed. *arduino-cli*. 2022. URL: <https://www.arduino.cc/pro/cli>.
- [Aru18] Arunava. *Convolutional Neural Network – Towards Data Science*. 2018. URL: <https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05> (visited on 11/08/2020).
- [BA94] R. J. Brachman and T. Anand. “The Process of Knowledge Discovery in Databases: A First Sketch”. In: *AAAI Technical Report WS-94-03* (1994).
- [BS18] N. Brown and T. Sandholm. “Superhuman AI for heads-up no-limit poker: Libratus beats top professionals”. In: *Science* 359.6374 (2018), pp. 418–424. ISSN: 0036-8075. DOI: 10.1126/science.aoa1733. eprint: <https://science.sciencemag.org/content/359/6374/418.full.pdf>. URL: <https://science.sciencemag.org/content/359/6374/418>.
- [BS19] P. Buxmann and H. Schmidt. *Künstliche Intelligenz*. Springer, 2019.
- [BWL20] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. “YOLOv4: Optimal Speed and Accuracy of Object Detection”. In: *arXiv e-prints*, arXiv:2004.10934v1 (2020). [pdf], arXiv: 2004.10934 [cs.CV].
- [Bal19] Balakreshnan. *Inferencing Model - Jetson Nano Without Docker container*. Eingesehen am 20.04.2020 [online]. 2019. URL: <https://github.com/balakreshnan/WorkplaceSafety/blob/master/InferencinginJetsonNano.md#inferencing-model---jetson-nano-without-docker-container>.
- [Bec18a] F. Beck. *Neuronale Netze – Eine Einführung – Aktivität*. 2018. URL: <http://www.neuronalesnetz.de/aktivitaet.html>.
- [Bec18b] R. Becker. *Machine / Deep Learning – Wie lernen künstliche neuronale Netze?* Eingesehen am 27.01.2020 [online]. 2018. URL: <https://jaai.de/machine-deep-learning-529/>.
- [Bec18c] R. Becker. *Transfer Learning – So können neuronale Netze voneinander lernen*. Eingesehen am 30.01.2020 [online]. 2018. URL: <https://jaai.de/transfer-learning-1739/>.
- [Bri15] D. Britz. *Understanding Convolutional Neural Networks for NLP*. 2015. URL: <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>.
- [Bro+17] T. B. Brown et al. “Adversarial Patch”. In: *Neural Information Processing Systems - Machine Learning and Computer Security Workshop*. [pdf]. 2017. arXiv: 1712.09665 [cs.CV].
- [Cha+00] P. Chapman et al. “CRISP-DM 1.0: Step-by-step data mining guide”. In: (2000). URL: <https://www.semanticscholar.org/paper/CRISP-DM-1.0%3A-Step-by-step-data-mining-guide-Chapman-Clinton-54bad20bbc7938991bf34f86dde0babfdbd2d5a72>.
- [Cha17] M. Chablani. “YOLO — You only look once, real time object detection explained”. In: *Towards Data Science* (2017). URL: <https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006>.
- [Cho18] F. Chollet. *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG, 2018.
- [Cho+19] A. Chowdhery et al. *Visual Wake Words Dataset*. [pdf], [github]. 2019. DOI: 10.48550/ARXIV.1906.05721. arXiv: 1906.05721 [cs.CV]. URL: <https://github.com/Mxbonn/visualwakewords>.

- [Cor20] N. Corporation, ed. *CUDA*. 2020. URL: <https://developer.nvidia.com/cuda-zone>.
- [DK16] S. Dodge and L. Karam. *Understanding How Image Quality Affects Deep Neural Networks*. 2016. URL: <https://arxiv.org/pdf/1604.04004.pdf>.
- [DV16] V. Dumoulin and F. Visin. “A guide to convolution arithmetic for deep learning”. In: (Mar. 23, 2016). arXiv: 1603 . 07285 [stat.ML]. URL: <https://arxiv.org/pdf/1603.07285v2.pdf>.
- [Den+09] J. Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848).
- [Den12] Deng, L. “The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]”. In: *IEEE Signal Processing Magazine* 29.6 (2012). [pdf], Link, pp. 141–142.
- [Dor13] C. F. Dormann. *Datenanalyse mit SPSS für Fortgeschrittene 1 – Datenaufbereitung und uni- und bivariate Statistik*. Vol. 2. Springer-Verlag, 2013.
- [Dör18] S. Dörn. “Neuronale Netze”. In: *Programmieren für Ingenieure und Naturwissenschaftler*. Springer, 2018, pp. 89–148.
- [Düs00] R. Düsing. “Knowledge Discovery in Databases”. In: *Wirtschaftsinformatik* 42.1 (2000), pp. 74–75. ISSN: 1861-8936. DOI: [10.1007/BF03250720](https://doi.org/10.1007/BF03250720).
- [Ecl20] Eclipse, ed. *Deep Learning for Java*. 2020. URL: <https://deeplearning4j.org/>.
- [Ert16] W. Ertel. *Grundkurs Künstliche Intelligenz*. Wiesbaden: Springer Fachmedien Wiesbaden, 2016. ISBN: 978-3-658-13548-5. DOI: [10.1007/978-3-658-13549-2](https://doi.org/10.1007/978-3-658-13549-2).
- [FAD18] M. Fezari and A. Al Dahoud. “Integrated Development Environment “IDE” For Arduino”. In: (Oct. 2018).
- [FPSS96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. “From Data Mining to Knowledge Discovery in Databases”. In: 17.3 (1996). [pdf], p. 37. DOI: [10.1609/aimag.v17i3.1230](https://ojs.aaai.org/index.php/aimagazine/article/view/1230). URL: <https://www.aaai.org/ojs/index.php/aimagazine/article/view/1230>.
- [FVP98] J. Funke and B. Vaterrodt-Pi"unnecke. *Was ist Intelligenz?* C.H. Beck Wissen, 1998.
- [Fac20] Facebook, ed. *PyTorch*. 2020. URL: <https://pytorch.org/>.
- [Fer12] D. A. Ferrucci. “Introduction to “This is Watson””. In: *IBM Journal of Research and Development* 56.3.4 (2012), 1:1–1:15. DOI: [10.1147/JRD.2012.2184356](https://doi.org/10.1147/JRD.2012.2184356).
- [Fis] R. A. Fisher. “The use of Multiple Measurements in Taxonomic Problems”. In: () .
- [Fis36a] R. A. Fisher. *Fisher's Iris Dataset*. [pdf]. 1936. URL: <https://archive.ics.uci.edu/ml/datasets/iris>.
- [Fis36b] R. A. Fisher. *Fisher's Iris Dataset*. [pdf]. 1936. DOI: [10.24432/C56C76](https://doi.org/10.24432/C56C76). URL: <https://archive.ics.uci.edu/ml/datasets/iris>.
- [Fou20a] P. S. Foundation, ed. *glob — Unix style pathname pattern expansion*. Eingesehen am 15.12.2019 [online]. 2020. URL: <https://docs.python.org/3/library/glob.html>.
- [Fou20b] P. S. Foundation, ed. *numpy*. Eingesehen am 15.12.2019 [online]. 2020. URL: <https://numpy.org/>.

-
- [Fou20c] P. S. Foundation, ed. *python-pickle-module-save-objects-serialization*. Eingeschen am 17.12.2019 [online]. 2020. URL: <https://pythonprogramming.net/python-pickle-module-save-objects-serialization/>.
- [Fou20d] P. S. Foundation, ed. *random*. Eingeschen am 16.12.2019 [online]. 2020. URL: <https://docs.python.org/3/library/random.html>.
- [Fou21a] P. S. Foundation, ed. *The Python Package Index*. 2021. URL: <https://pypi.org>.
- [Fou21b] P. S. Foundation, ed. *pycocotools*. 2021. URL: <https://pypi.org/project/pycocotools>.
- [GC06] P. Gluchowski and P. Charmoni. *Analytische Informationssysteme: Business Intelligence-Technologien und -Anwendungen*. Vol. 3. 2006.
- [GMG16] P. Gysel, M. Motamedi, and S. Ghiasi. “Hardware-oriented approximation of convolutional neural networks”. In: *arXiv preprint arXiv:1604.03168* (2016).
- [GSS14] I. J. Goodfellow, J. Shlens, and C. Szegedy. “Explaining and Harnessing Adversarial Examples”. In: *arXiv 1412.6572* (2014). [\[pdf\]](#). arXiv: 1412.6572 [stat.ML].
- [Goo19a] Google, ed. *TensorFlow*. 2019. URL: <https://www.tensorflow.org/>.
- [Goo19b] Google, ed. *TensorFlow*. 2019. URL: <https://www.tensorflow.org/lite>.
- [Goo20a] Google, ed. *TensorFlow - Convolutional Neural Network Example*. 2020. URL: \url{https://www.tensorflow.org/tutorials/images/cnn}.
- [Goo20b] Google, ed. *TensorFlow Lite Guide*. 2020. URL: \url{https://www.tensorflow.org/lite/guide}.
- [Goo21] Google, ed. *Willkommen bei Colaboratory*. 2021. URL: <https://colab.research.google.com/>.
- [Gu+18] J. Gu et al. “Recent Advances in Convolutional Neural Networks”. In: *Pattern Recognition* 77 (2018), pp. 354–377. ISSN: 0031-3203. DOI: [10.1016/j.patcog.2017.10.013](https://doi.org/10.1016/j.patcog.2017.10.013). URL: <http://arxiv.org/pdf/1512.07108v6.pdf>.
- [Gup20] D. Gupta. *Activation Functions – Fundamentals Of Deep Learning*. 2020. URL: <https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/>.
- [Gö] U. Göttingen, ed. *Kapitel 3: Erste Schritte der Datenanalyse*. URL: <https://www.uni-goettingen.de/de/document/download/9b4b2033cba125be183719130e524467.pdf/mvsec3.pdf>.
- [He+16] K. He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [\[pdf\]](#). 2016. DOI: [10.1109/cvpr.2016.90](https://doi.org/10.1109/cvpr.2016.90).
- [How+] A. G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. [\[pdf\]](#). URL: <http://arxiv.org/pdf/1704.04861v1.pdf>.
- [Int19] Intel Corporation, ed. *OpenVino - System Requirements*. 2019. URL: <https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit/system-requirements.html>.
- [JES20] Y. Jia and E. Evan Shelhamer. *Caffe*. Ed. by B. A. I. Research. 2020. URL: <http://caffe.berkeleyvision.org/>.
- [JL07] J. Janssen and W. Laatz. *Statistische Datenanalyse mit SPSS für Windows*. Vol. 6. Springer-Verlag, 2007.

- [KH09] A. Krizhevsky and G. Hinton. *Learning Multiple Layers of Features from Tiny Images*. 2009. URL: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>.
- [KK07] I. Kononenko and M. Kukar, eds. *Machine Learning and Data Mining*. Woodhead Publishing, 2007. ISBN: 978-1-904275-21-3.
- [KM06] L. A. Kurgan and P. Musilek. “A Survey of Knowledge Discovery and Data Mining Process Models”. In: *The Knowledge Engineering Review* 21.1 (2006), pp. 1–24. ISSN: 0269-8889. DOI: [10.1017/S0269888906000737](https://doi.org/10.1017/S0269888906000737).
- [KSH12a] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012, pp. 1097–1105. DOI: [0.1145/3065386](https://doi.org/10.1145/3065386). URL: proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.
- [KSH12b] A. Krizhevsky, I. Sutskever, and G. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Neural Information Processing Systems* 25 (Jan. 2012). DOI: [10.1145/3065386](https://doi.org/10.1145/3065386). URL: http://www.cs.toronto.edu/~kriz/imagenet_classification_with_deep_convolutional.pdf.
- [KSH17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. *CIFAR-10 and CIFAR-100 datasets*. 2017. URL: <https://www.cs.toronto.edu/~kriz/cifar.html>.
- [Kag] *Iris Flower Dataset*. 2018. URL: <https://www.kaggle.com/arshid/iris-flower-dataset>.
- [Kri08] D. Kriesel. *Ein kleiner Überblick über Neuronale Netze*. Eingesehen am 28.12.2019 [online]. 2008. URL: <http://www.dkriesel.com/index.php>.
- [Kru+15] R. Kruse et al. *Computational Intelligence*. Wiesbaden: Springer Fachmedien Wiesbaden, 2015. ISBN: 978-3-658-10903-5. DOI: [10.1007/978-3-658-10904-2](https://doi.org/10.1007/978-3-658-10904-2).
- [Kä11] W.-M. Kähler. *Statistische Datenanalyse: Verfahren verstehen und mit SPSS gekonnt einsetzen*. Vol. 7. Springer-Verlag, 2011.
- [LCB13] Y. LeCun, C. Cortes, and C. Burges. *MNIST handwritten digit database*. 2013. URL: <http://yann.lecun.com/exdb/mnist/>.
- [LP20] López de Lacalle, Luis Norberto and J. Posada. “New Industry 4.0 Advances in Industrial IoT and Visual Computing for Manufacturing Processes”. In: (2020).
- [LWX19] Y. Lu, S. Xie, and S. Wu. “Exploring Competitive Features Using Deep Convolutional Neural Network for Finger Vein Recognition”. In: *IEEE Access* PP (Mar. 2019). [[pdf](#)], pp. 1–1. DOI: [10.1109/ACCESS.2019.2902429](https://doi.org/10.1109/ACCESS.2019.2902429).
- [LeC+98] Y. LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: [10.1109/5.726791](https://doi.org/10.1109/5.726791).
- [Lin+14] T. Lin et al. “Microsoft COCO: Common Objects in Context”. In: *The Computing Research Repository (CoRR)* abs/1405.0312 (2014). arXiv: [1405.0312](https://arxiv.org/abs/1405.0312). URL: <http://arxiv.org/abs/1405.0312>.
- [Lin+21] T.-Y. Lin et al. *COCO - Common Objects in Context*. 2021. URL: <https://cocodataset.org/>.

-
- [MP43] W. S. McCulloch and W. Pitts. “A Logical Calculus of the Ideas Immanent in Nervous Activity”. In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133. ISSN: 1522-9602. DOI: [10.1007/BF02478259](https://doi.org/10.1007/BF02478259).
- [MR10] O. Maimon and L. Rokach. *Data Mining and Knowledge Discovery Handbook*. Boston, MA: Springer US, 2010. ISBN: 978-0-387-09822-7. DOI: [10.1007/978-0-387-09823-4](https://doi.org/10.1007/978-0-387-09823-4).
- [MXN20] A. S. F. A. MXNet, ed. *MXNet*. 2020. URL: <https://mxnet.apache.org/>.
- [McC+06] J. McCarthy et al. “A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955”. In: *AI Magazine* 27.4 (2006). DOI: doi.org/10.1609/aimag.v27i4.1904.
- [Mck] “Smartening up with Artificial Intelligence (AI) - What’s in it for Germany and its Industrial Sector?” In: McKinsey & Company, 2017.
- [Mic19] U. Michelucci. *Advanced Applied Deep Learning*. Berkeley, CA: Apress, 2019. ISBN: 978-1-4842-4975-8. DOI: [10.1007/978-1-4842-4976-5](https://doi.org/10.1007/978-1-4842-4976-5).
- [Mic20] Microsoft, ed. *The Microsoft Cognitive Toolkit*. 2020. URL: <https://docs.microsoft.com/en-us/cognitive-toolkit/>.
- [Moe18] J. Moeser. *Künstliche neuronale Netze - Aufbau & Funktionsweise*. 2018. URL: <https://jaai.de/kuenstliche-neuronale-netze-aufbau-funktion-291/>.
- [Mun96] D. C. Munson. “A note on Lena”. In: *IEEE Transactions on Image Processing* 5.1 (1996), pp. 3–3. DOI: [10.1109/TIP.1996.8100841](https://doi.org/10.1109/TIP.1996.8100841).
- [NVI15] NVIDIA Corporation, ed. *NVIDIA TensorRT*. 2015. URL: \url{https://developer.nvidia.com/tensorrt}.
- [NVI20] NVIDIA Corporation, ed. *NVIDIA CUDA Toolkit 11.0.161*. [pdf]. 2020.
- [Nam17] I. Namatevs. “Deep Convolutional Neural Networks: Structure, Feature Extraction and Training”. In: *Information Technology and Management Science* 20.1 (2017). DOI: [10.1515/itms-2017-0007](https://doi.org/10.1515/itms-2017-0007).
- [Nie15] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: <http://neuralnetworksanddeeplearning.com/chap1.html>.
- [Nwa+18] C. Nwankpa et al. “Activation Functions: Comparison of trends in Practice and Research for Deep Learning”. In: (Nov. 8, 2018). arXiv: [1811.03378 \[cs.LG\]](https://arxiv.org/abs/1811.03378). URL: <https://arxiv.org/pdf/1811.03378v1.pdf>.
- [PN20] I. Preferred Networks, ed. *Chainer*. 2020. URL: <https://chainer.org/>.
- [PUC02] S. Pyo, M. Uysal, and H. Chang. “Knowledge Discovery in Database for Tourist Destinations”. In: *Journal of Travel Research* 40.4 (2002), pp. 374–384. ISSN: 0047-2875. DOI: [10.1177/0047287502040004006](https://doi.org/10.1177/0047287502040004006).
- [Ped+11] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [Pla19] J. Van der Plas. *Get started with Google Colaboratory (Coding TensorFlow)*. 2019. URL: <https://www.youtube.com/watch?v=inN8seMm7UI> (visited on 08/23/2022).
- [QBL18] H. Qi, M. Brown, and D. G. Lowe. “Low-Shot Learning with Imprinted Weights”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5822–5830.
- [RG20] S. Romero and L. George. *Updating the Portenta bootloader*. 2020. URL: <https://www.arduino.cc/pro/tutorials/portenta-h7/por-ard-bl>.

- [Raj20] A. Raj. "AlexNet CNN Architecture on Tensorflow (beginner)". In: (2020). Ed. by Kaggle. URL: <https://www.kaggle.com/vortexkol/alexnet-cnn-architecture-on-tensorflow-beginner>.
- [Red+15] J. Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. [pdf]. 2015. DOI: [10.48550/ARXIV.1506.02640](https://doi.org/10.48550/ARXIV.1506.02640). URL: <https://arxiv.org/pdf/1506.02640.pdf> (visited on 02/17/2021).
- [Red20] J. Redmon. *MNIST in CSV*. 2020. URL: <https://pjreddie.com/projects/mnist-in-csv/>.
- [Red21] J. Redmon. *YOLO: Real-Time Object Detection*. 2021. URL: <https://pjreddie.com/darknet/yolo/>.
- [Ric83] E. Rich. *Artificial Intelligence*. McGraw-Hill Inc., US, 1983. ISBN: 9780070522619.
- [Run15] T. A. Runkler. *Data Mining – Modelle und Algorithmen intelligenter Datenanalyse*. Vol. 2. Springer Fachmedien, 2015.
- [SH06] L. Sachs and J. Hedderich. *Angewandte Statistik – Methodensammlung mit R*. Vol. 12. Springer-Verlag, 2006.
- [SIG20a] K. SIG, ed. *Keras Dense Layer*. 2020. URL: \url{https://keras.io/api/layers/core_layers/dense/}.
- [SIG20b] K. SIG, ed. *Keras*. 2020. URL: <https://keras.io/>.
- [SJM18] N. Sharma, V. Jain, and A. Mishra. "An Analysis Of Convolutional Neural Networks For Image Classification". In: *Procedia Computer Science* 132 (2018), pp. 377–384. ISSN: 18770509. DOI: [10.1016/j.procs.2018.05.198](https://doi.org/10.1016/j.procs.2018.05.198).
- [SKP15] F. Schroff, D. Kalenichenko, and J. Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [pdf]. 2015, pp. 815–823.
- [SP06] V. Sahni and C. Patvardhan. "Iris Data Classification Using Quantum Neural Networks". In: *AIP Conference Proceedings* 864.1 (2006), pp. 219–227. DOI: [10.1063/1.2400893](https://doi.org/10.1063/1.2400893). eprint: <https://aip.scitation.org/doi/pdf/10.1063/1.2400893>. URL: <https://aip.scitation.org/doi/abs/10.1063/1.2400893>.
- [SR21] L. G. Sebastian Romero. *Creating a basic face filter*. 2021. URL: <https://www.arduino.cc/pro/tutorials/portenta-h7/por-openmv-fd>.
- [SSS19] F. Siddique, S. Sakib, and M. A. B. Siddique. "Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers". In: *2019 5th International Conference on Advances in Electrical Engineering (ICAEE)*. [pdf]. 2019, pp. 541–546.
- [SW16] M. Schutten and M. Wiering. "An Analysis on Better Testing than Training Performances on the Iris Dataset". In: *Belgian Dutch Artificial Intelligence Conference*. Link, Link. Sept. 2016.
- [SZ15] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. [pdf]. 2015. arXiv: [1409.1556](https://arxiv.org/abs/1409.1556) [cs.CV].
- [Sah18] S. Saha. *A Comprehensive Guide to Convolutional Neural Networks – the ELI5 Way*. 2018. URL: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> (visited on 01/25/2018).
- [San+18] M. Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *arXiv e-prints*, arXiv:1801.04381 (2018). [pdf]. arXiv: [1801.04381](https://arxiv.org/abs/1801.04381) [cs.CV].

- [Sha+16] W. Shang et al. “Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units”. In: *International Conference on Machine Learning (ICML)* (Mar. 2016). [pdf], pp. 2217–2225.
- [Sha18] S. Sharma. *Aktivierungsfunktionen: Neuronale Netze*. Ed. by AI-United. 2018. URL: <http://www.ai-united.de/aktivierungsfunktionen-neuronale-netze/>.
- [Sha+20] V. Shankar et al. “Evaluating Machine Accuracy on ImageNet”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. [pdf]. PMLR - Proceedings of Machine Learning Research, 2020, pp. 8634–8644. URL: <http://proceedings.mlr.press/v119/shankar20c.html>.
- [Sit+18] C. Sitawarin et al. *DARTS: Deceiving Autonomous Cars with Toxic Signs*. [pdf]. 2018. arXiv: 1802.06430 [cs.CR].
- [Sou20] F. O. Source, ed. *Caffe2*. 2020. URL: <https://caffe2.ai/>.
- [Sze+14] C. Szegedy et al. *Going Deeper with Convolutions*. [pdf]. 2014. DOI: 10.48550/ARXIV.1409.4842. arXiv: 1409.4842 [cs.CV].
- [TL20] M. Tan and Q. V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. [pdf]. 2020. arXiv: 1905.11946 [cs.LG].
- [TS19] S. Tan and A. Sidhu. “CUDA implementation”. In: Apr. 2019, pp. 63–67. ISBN: 978-3-030-17273-2. DOI: 10.1007/978-3-030-15585-8_5.
- [Tea20a] O. Team, ed. *OpenCV*. 2020. URL: <https://opencv.org/>.
- [Tea20b] O. Team, ed. *OpenVino*. 2020. URL: <https://www.openvino.org/>.
- [Tea21a] E. I. Team, ed. *Documentation*. 2021. URL: <https://docs.edgeimpulse.com/docs>.
- [Tea21b] E. I. Team, ed. *Object Detection*. 2021. URL: <https://docs.edgeimpulse.com/docs/object-detection>.
- [Tea21c] O. Team, ed. *OpenMV*. 2021. URL: https://docs.openmv.io/openmvcam/tutorial/openmvvide_overview.html.
- [The21] TheAIGuy. *YOLOv4-Cloud-Tutorial*. 2021. URL: <https://github.com/theAIGuysCode/YOLOv4-Cloud-Tutorial>.
- [UK21] A. Unwin and K. Kleinman. “The Iris Data Set: In Search of the Source of Virginica”. In: *Significance* 18 (2021). [pdf]. DOI: 10.1111/1740-9713.01589.
- [Wan+16] F. Wang et al. “Where does AlphaGo go: from church-turing thesis to AlphaGo thesis and beyond”. In: *IEEE/CAA Journal of Automatica Sinica* 3.2 (2016), pp. 113–120. DOI: 10.1109/JAS.2016.7471613.
- [Web97] A. Weber. *The USC-SIPI Image Database: Version 5*. link. 1997. URL: <http://sipi.usc.edu/services/database/Database.html>.
- [Wik20] Wikipedia, ed. *OpenCV*. Eingesehen am 16.12.2019 [online]. 2020. URL: <https://de.wikipedia.org/wiki/OpenCV>.
- [Wro98] S. Wrobel. “Data Mining und Wissensentdeckung in Datenbanken”. In: *Künstliche Intelligenz* 1 (1998). [pdf]. URL: <ftp://ftp.fhg.de/archive/gmd/ais/publications/1998/wrobel.98.data-mining.pdf>.
- [ZJ17] W. Zhiqiang and L. Jun. “A review of object detection based on convolutional neural network”. In: *2017 36th Chinese Control Conference (CCC)*. 2017, pp. 11104–11109. DOI: 10.23919/ChiCC.2017.8029130.

-
- [Zha+15] X. Zhang et al. *Accelerating Very Deep Convolutional Networks for Classification and Detection*. [\[pdf\]](#). 2015. arXiv: [1505.06798](#) [cs.CV].
 - [Zie15] W. Ziegler. *Neuronale Netze – Teil 2*. Eingesehen am 26.01.2020 [online]. 2015. URL: <https://entwickler.de/online/neuronale-netze-teil-2-159753.html>.
 - [fas20] fast.ai, ed. *FastAI*. 2020. URL: <https://www.fast.ai/>.

Index

- Libraries, 81
- File
- .cfg, 142, 143
 - .cli-config.yml, 108
 - /content, 137
 - arduino-cli-0.33.1.windows.exe, 107
 - arduino-cli.exe, 107
 - arduino-cli_0.33.0_Windows_64bit.zipCaffe, 83
 - 107
 - arduino_data, 108
 - Arduino_Pro_Tutorials, 219
 - C:\Users\<username>\.keras\datasets, 68
 - cli_test, 111
 - file_name, 74
 - imagenet-camera.py, 78
 - images_RF.zip, 148
 - Introductory Colab, 137
 - label.txt, 144
 - monitor_log.bat, 117
 - obj.data, 139, 142–144
 - obj.names, 139, 142–144
 - obj.zip, 139, 142
 - process.py, 139
 - README.md, 108
 - skikitlearn, 69
 - t10k-images-idx3-ubyte.gz, 67
 - t10k-labels-idx1-ubyte.gz, 67
 - test.txt, 139, 142–144
 - test.zip, 142
 - train-images-idx3-ubyte.gz, 67
 - train-labels-idx1-ubyte.gz, 67
 - train.txt, 129, 139, 142–144
 - yolov4- custom.cfg, 139
 - yolov4-custom.cfG, 139
 - yolov4.cfg, 143
 - YOLOV4.tex, 139
- AlexNet, 55, 56, 78
- Alom:2018, 55
- Amazon, 83
- AP, 17, 139
- API, 17, 82, 88
- Application Programming Interface
 - see* API, 17, 82
- Arduino Nano 33 BLE sense, 107
- Average Precision
 - see* AP, 17, 139
- backpropagation, 56
- Basemap, 85
- Blob-Algorithmus, 128
- Bokeh, 85
- Caffe, 83
- Caffe2, 83
- Canadian Institute For Advanced Research
 - see* CIFAR
 - see* Datensatz, 11, 12, 15, 17
- Canadian Institute for Advanced Research, 67
- Central Processing Unit
 - see* CPU, 17, 86
- Chainer, 83
- CLI, 12, 17, 107, 108, 111, 112
- CNN, 4, 15, 17, 34, 47–51, 54, 55, 57, 60–63
- CNTK, 83
- COCO, 15, 17, 72–77, 88, 139
- Command Line Interface
 - see* CCI, 12, 17, 107
- Common Objects in Context
 - see* COCO, 15, 17
- Compute Unified Device Architecture
 - see* CUDA, 17, 81
- Continuous Development, 107
- Continuous Integration, 107
- Convolutional Neural Network
 - see* CNN, 4, 15, 17
- CPU, 17, 86, 140
- Cross Stage Partial Network
 - see* CSP, 17, 139
- CSP, 17, 139
- CUDA, 17, 81, 82, 86, 140
- cv2, 87
- Dataset
- CIFAR-10, 67
 - CIFAR-100, 67
 - Fisher's Iris Data Set, 69–72
 - ImageNet, 77
 - MNIST, 66, 67
 - TensorFlow, 67

-
- UTKFace, 77
 Datensatz
 CIFAR, 11, 12, 15, 17, 62, 64, 67–69
 NIST Special Database 3, 17, 66
 NIST Special Database 1, 17, 66, 67
 UTKFace, 77
 Deeplearning4J, 83
 DeepScene, 78
 Edge IMPulse, 223
 Edge Impulse, 107, 225
 EfficientNet, 56
 Eli5, 86
 FastAI, 83
 feature maps, 58
 Frameworks, 81
 glob, 87
 GoogleNet, 78
 GPU, 17, 55, 83, 86, 139, 140, 145
 Graphics Processing Unit
 see GPU, 17, 55
 hyperparameter, 57
 hyperparameters, 57
 Image classification, 54
 ImageNet, 57, 78
 see Dataset 77
 Inception, 57, 78
 InceptionNet, 56
 IPython, 84
 jetson-inference, 78
 Jupyter-Notebook, 81
 Keras, 82
 LED, 17, 113
 LeNet, 55
 Light Emitting Diode
 see LED, 17
 LightGBM, 85
 math, 87
 matplotlib, 84
 Microsoft Cognitive Toolkit, 83
 Mlpy, 86
 MobileNet, 56
 MobileNetV2, 56, 227
 MonileNet, 57
 MXNet, 83
 NetworkX, 85
 Nicla Vision, 107
 NIST Special Database 1
 see Datensatz, 17, 66
 NIST Special Database 3
 see Datensatz, 17, 66
 NLTK, 85
 NumPy, 84
 Object detection, 54
 OpenCV, 86
 OpenNN, 87
 OpenVINO, 86
 os, 87
 PAN, 17, 139
 Pandas, 84
 Path Aggregation Network
 see PAN, 17, 139
 Pattern, 85
 pickle, 88
 PIL, 87
 Portenta H7, 107
 PyPI, 88
 Python Software Foundation, 88
 PyTorch, 82
 random, 87
 Rectified Linear Unit
 see ReLu, 17, 60
 ReLu, 17, 60, 64
 Residual Neural Network
 see ResNet, 17, 56
 ResNet, 17, 56, 57, 78
 ResNets, 78
 RLE, 17, 76
 Run-Length Encoding, 17, 76
 scikit-learn, 84
 SciPy, 84
 Scrapy, 85
 Seaborn, 85
 segmentation, 54
 SP, 17, 139
 SSD-Mobilenet-V2, 78
 Statsmodels, 86
 Streaming Processor
 see SP, 17, 139
 TensorFlow, 81
 TensorFlow Lite, 82
 TensorRT, 82
 Theano, 83
 UCI Machine Learning Repository, 69
 VGG, 56, 57, 78
 Video Processing Unit

- see* VPU, 17, 86
 - Virtual Machine
 - see* VM, 17, 141
 - VM, 17, 141
 - VPU, 17, 86
- YAML, 108
 - YOLO, 17, 138, 139, 143
 - You Only Look Once
 - see* YOLO, 17, 138