Univerzitet "Džemal Bijedić" u Mostaru
Fakultet informacijskih tehnologija

Godina studija: četvrta

Korištenje *computer vision* tehnologija u detekciji oštećenja objekta u industrijskom proizvodnom okruženju

Završni rad nakon prvog ciklusa

Mentor:
Doc. Dr. Elmir Babović

Student:
Mursel Musabašić, IB190138

Mostar, mart 2022

IZJAVA O AUTORSTVU

Ja, Mursel (Ramiz) Musabašić, student Fakulteta informacijskih tehnologija, Univerziteta „Džemal Bijedić" u Mostaru, pod punom moralnom, materijalnom i krivičnom odgovornošću,

izjavljujem

da je rad pod naslovom:

" Korištenje computer vision tehnologija u detekciji oštećenja objekta u industrijskom proizvodnom okruženju"

u potpunosti rezultat sopstvenog istraživanja, gdje su korišteni sadržaji (tekst, ilustracije, tabele itd.) drugih autora jasno označeni pozivanjem na izvor i ne narušavaju bilo čija vlasnička ili autorska prava.

Mostar, mart 2022.

_____

Mursel Musabašić, IB190138

Sadržaj

Slike

Sažetak

Mašinsko učenje stiče sve veći popularitet na IT tržištu, ali isto tako i primjena mehanizama koji čine sastavni dio tog učenja najviše utiču na razvoj četvrte industrijske revolucije (Industry V4). Zajedno sa virtuelnom i proširenom realnosti (Virtual & Augmented Reality), te industrijskim Internet of Things (IIoT), i umjetna inteligencija (Artifical Inteligence) predstavlja jedan od spomenutih temelja četvrte industrijske revolucije. Umjetna inteligencija u sklopu Industrije v4 se bazira na mašinskom učenju. U svom najprostijem značenju "mašinsko učenje" je jedan dio umjetne inteligencije koja se bazira na statističkim podacima koji su dati kompjuterskim sistemima da "uče" sa potrebnim podacima, a bez potrebe programiranja.

Upravo zbog mogućnosti da mašine uče prepoznati određene šablone i da shodno tome stiću znanje i iskustvo, ova tehnologija je našla svoje mjesto u Industriji v4. Bitno je napomenuti da umjetna inteligencija (u nastavku teksta AI) nije vizija budućnosti, već nešto što se danas već koristi u raznim poslovnim procesima. Mašinsko učenje prepoznaje karakteristike i veze u ogromnoj količini podataka tzv. "Big data", a koji mogu biti struktuirani ili ne-struktuirani, te koristeći specifične algoritme izvodi opće zaključke o tim podacima (generalizacija).

U nastavku rada ukratko će biti obrađena tehnika prepoznavanja oštećenih industrijskih predmeta koristeći OpenMV kameru i Edge Impulse ML web platformu.

U procesu treniranja modela koristi se *supervised learning* baziran na problemu klasifikacije slika, tj. ovo je učenje na osnovu primjera, gdje primjeri predstavljaju slike oštećenih i neoštećenih industrijskih predmeta, te slika kada predmet nije vizuelno prikazan.

Abstract

Machine learning is gaining more and more popularity in the IT market, but also the application of the mechanisms that are an integral part of that learning has the greatest impact on the development of the fourth industrial revolution (Industry V4). Together with virtual and augmented reality (Virtual & Augmented Reality), and industrial Internet of Things (IIoT), and artificial intelligence (Artifical Intelligence) is one of the mentioned foundations of the fourth industrial revolution. Artificial intelligence within Industry v4 is based on machine learning. In its simplest sense, "machine learning" is a piece of artificial intelligence that is based on statistics given to computer systems to "learn" with the necessary data, without the need for programming.

Precisely because of the possibility for machines to learn to recognize certain patterns and to acquire knowledge and experience accordingly, this technology has found its place in Industry v4. It is important to note that artificial intelligence (hereinafter AI) is not a vision of the future, but something that is already used in various business processes today. Machine learning recognizes characteristics and connections in a huge amount of so-called data. "Big data", which can be structured or non-structured, and using specific algorithms draws general conclusions about this data (generalization).

In the continuation of the paper, the technique of recognizing damaged industrial objects using the OpenMV camera and the Edge Impulse ML web platform will be briefly discussed.

In the process of model training, supervised learning based on the problem of image classification is used, ie. this is an example-based learning, where the examples represent images of damaged and undamaged industrial objects, and when the image of the object is not visually displayed.

1. Uvod

Ovaj dokument ima za cilj da objasni i vizuelno prikaže na koji način je moguće implementirati simulaciju uređaja za detekciju oštećenja na industrijskim predmetima. Kompletan proces implementacije jednog ovakvog projekta biti će objašnjen korak po korak.

Prvo ćemo se osvrnuti na mehanički dio implementacije projekta gdje će biti objašnjeno koji su uređaji ili pojedini mehanički dijelovi ugrađeni i koja im je svrha u projektu.

Potom ćemo se osvrnuti na softverski dio projekta gdje će biti objašnjeno na koji način se kontroliraju navedeni uređaji putem OpenMV Cam mikrokontrolerske ploče, te proces učenja/treniranja modela koji će se koristiti za prepoznavanje oštećenja na industrijskim predmetima.

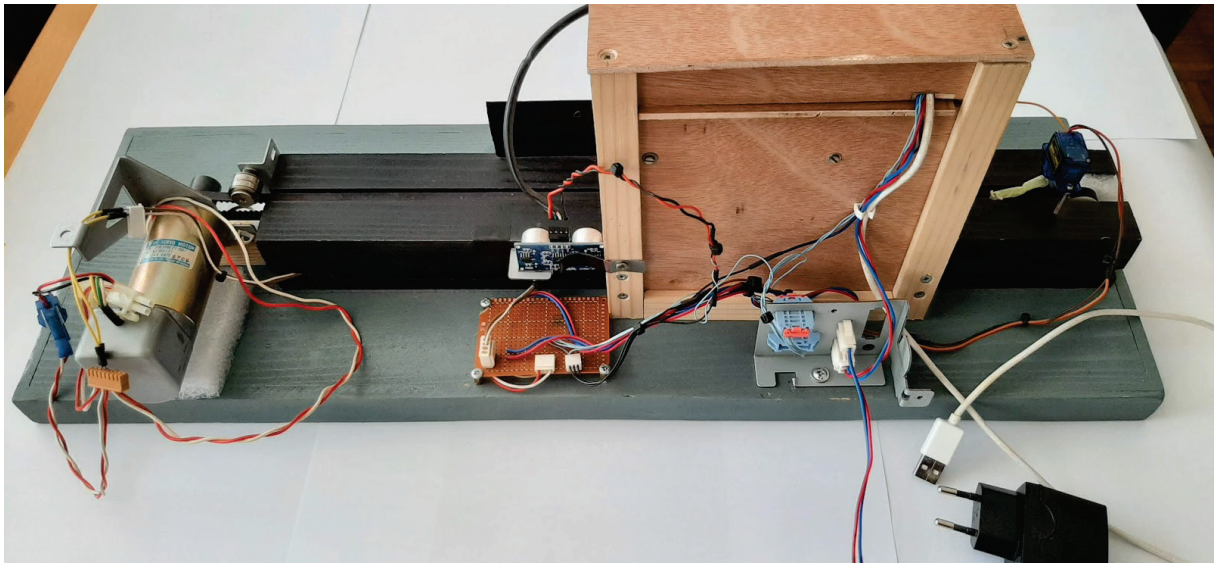
Naposljetku u kraćim crtama osvrnuti ćemo se na kompletan proces koji se dešava „iza scene“, a koji ima za cilj implementaciju jednog ovakvog uređaja.

2. Mehanički dio projekta

Kada govorimo o konstrukciji uređaja potrebno je napomenuti da su profesionalni uređaji ovakvog tipa sagrađeni od metala ili tvrde plastike. Metal ili tvrda plastika se koriste kao potpora kod stabilnosti rada uređaja, jer obično takvi uređaji imaju ugrađen više od jednog, u većini slučajeva, elektromotora za ispravan rad hidrauličkih ili pneumatskih dijelova mašine. Kada povrh toga mašina ima ugrađene mjerne senzore onda je konstrukcija od pomenutih materijala neizostavna stvar.

Iako uređaj koje je prezentovan u nastavku teksta nije konstruisan od metala ili određene tvrde plastike, već od drveta (puno drvo i špera) ipak može poslužiti kao „proof-of-concept" da je moguće implementirati jedan ovakav tip i na profesionalnom nivou.

Bitno je napomenuti da uređaj radi parcijalnu detekciju oštećenja, tj. uređaj će izvršiti *skeniranje* gornjeg dijela predmeta koji je u tom momentu postavljen na traci. Za detekciju kompletne površine predmeta potrebna je sasvim drugačija izrada konstrukcije uređaja.



Slika 1 - Uređaj za detekciju oštećenja na industrijskim predmetima

2.1 Konstrukcija uređaja

Prethodno smo spomenuli da je konstrukcija izrađena od drveta. Podnožje konstrukcije čini drvena ploča dužine 590mm i širine 200mm, te služi kao potpora za sve ostale dijelove uređaja. Ploča je izrađena od brodskog poda i par letvica koje služe da bi povezali i bolje učvrstili komade brodskog poda. Na sredini ploče pričvršćen je drveni komad (gredica) dimenzija 460x62x45 (DxŠxV u mm). Navedeni komad je naknadno obrađen cirkularom gdje su napravljene tri kanalice čitavom dužinom. Dvije kanalice sa strane služe ili mogu poslužiti kod kabliranja, dok kroz kanalicu koja se nalazi na sredini sa gornje strane prolazi remen koji služi kao primjer proizvodne trake. Sa jedne strane (tzv. ulazna strana) te gredice nalazi se DC motor koji pogoni remen, te ležajevi koji su montirani na nosač koji je pričvršćen na pomenuti komad te blagim pritiskom na remen vrše stabilizaciju remena prilikom rada uređaja. Sa druge strane (izlazna strana) je montiran plastični ležaj preko kojeg prelazi remen i mikro servo motor koji produženim krakom odvaja oštećene od neoštećenih predmeta. Plastični ležaj je pričvršćen i za gredicu i za podnožje uređaja.

Na sredini ove gredice konstruisana je drvena kutija dimenzija 170x100x180 (DxŠxV u mm) napravljena tako da sve četiri strane predstavljaju klizne špere koje se mogu rasformirati kada je potrebno uraditi određenu izmjenu ili popravku. Kutija se sastoji od dva nivoa gdje se prvi nivo nalazi odmah iznad remena i u njoj su ugrađene dvije auto-LED diode, dok na drugom nivou se nalazi OpenMV H7 kamera sa ugrađenim motornim štitom/ekspanziom i kablovi za spajanje kamere, LED-a i drugih uređaja montiranih na konstrukciji. Dva termina „ulazna strana" i „izlazna strana" usko su povezane sa navedenom kutijom i predstavljaju ulaznu stranu sa koje dolaze predmeti na remenu/traci i ulazu u kutiju na proces detekcije oštećenja i mjesto gdje izlaze iz kutije (izlazna strana) prema procesu selekcije na loše i ispravne predmete. Drugi nivo kutije je pokriven sa šperom i učvršćen šarafima, te pored toga što služi kao poklopac, ujedno daje i čvrstoću i stabilnost kutije prilikom pomjeranja ili rada ugrađenih motora.

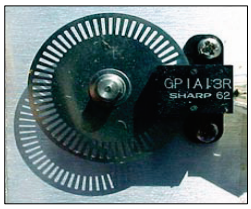
Bitno je napomenuti i skrenuti pažnju pošto je središnja kutija izgrađena od drveta, njenim čestim otvaranjem može doći do pomjeranja ležišta kamere, te time i narušiti preciznost detekcije oštećenja na predmetima.

2.2 DC servo motori

U uređaj su ugrađena dva servo motora gdje servo motor oznake DSE38BE27-001 pokreće remen, dok drugi servo motor (SG90) služi za sortiranje predmeta na loše i ispravne. Motorima se upravlja korištenjem PWM (Pulse Width Modulation) tehnikom.

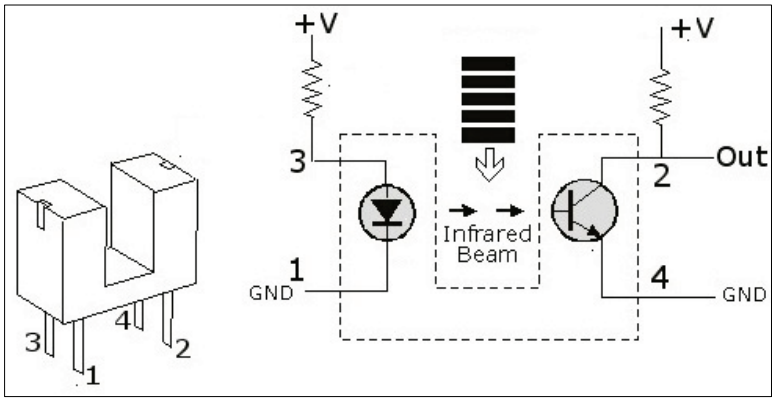
2.2.1 DC servo motor – DSE38BE27-001

DSE38BE27-001 je DC servo motor sa maksimalnim naponom do 24V, te maksimalnom brzinom od 4400 obrtaja u minuti (rpm). Servo motor je spojen sa senzorom za povratnu informaciju o položaju (Photointerrupter), a to je omogućeno sa ugrađenim enkoderom položaja oznake GP1A30R. Iako uređaj za detekciju oštećenih poredmeta ne koristi funkciju detekcije položaja, napomenuti ćemo da se u osnovi radi o optički-baziranom foto senzoru koji se sastoji od IR LED-a s jedne strane i tranzistora sa druge strane optičke sklopke. Primjer GP1A30R enkodera položaja motora je prikazan na slici 2, a šema komponente je prikazana na slici 4.



Slika 2 - GP1A30R

Slika 3 - DSE38BE27-001 DC



Slika 4 - Šema foto-senzora

2.2.2 DC servo motor – SG90

SG90 je servo motor iz grupe mikro servo motora, te zbog svoje veličine 23x12x28.5 (DxŠxV u mm), težine od svega 9 grama, velike izlazne snage od 2,5 kg/cm, ulaznog napona od 4,8 – 6v, te kao takav idealan je za početnike u elektronici (hobisti). SG90 podržava rotiranje do 180° tj. po 90° u oba pravca. Obzirom da se radi o kontrolnom motoru SG90 očekuje da mu se pošalje signal svakih 20ms, te od dužine trajanja impulsa zavisi i njegova rotacija. Za neutralnu poziciju (pozicija 0) trajanje signala iznosi 1,5ms, za sve pozicije do 90° skroz na lijevo trajanje signala mora biti manje od 1,5ms ili preko 1,5ms za sve pozicije motora do 90° skroz na desno. Signal o kojem govorimo jeste dobiven posredstvom metode modulacije širine impulsa (eng: Pulse Width Modulation), a o kojem će biti nešto više napisano kasnije u nastavku teksta.

Slika 5 - Mikro-servo motor oznake SG90

2.3 Ultrazvučni senzor - HC-SR04P

Kako bi se postiglo što preciznije zaustavljanje predmeta ispod objektiva kamere koristi se ultrazvučni senzor oznake HC-SR04P. Ovaj senzor koristeći sonar detektuje predmete ispred sebe u dometu od 2 do 400cm. Senzor se sastoji od dva modula (vizuelno izgledaju kao dva mala bubnja), i to *emitera* i *prijemnika*. Kako bi aktivirali senzor potrebno je na PIN *emitera/TRIG* dovesti 5V u trajanju od 10 mikrosekundi. Nakon aktiviranja senzor šalje 8 impulsa od 40KHz valne dužine koja je nečujna za ljude te čeka na njihovu refleksiju. Nakon što zaprimi povratne informacije/eho, te informacije šalje mikrokontroleru na izračun. U slučaju da je vrijeme čekanja na eho traje više od 30ms smatra se da ne postoji objekat ispred senzora. Bitno je napomenuti da je potrebno rezultat računanja

rastojanja podijeliti sa 2, jer se računa kompletno vrijeme od slanja do prijema informacija o refleksiji. Senzor se napaja sa 5V i spada u kategoriju nisko preciznih uređaja (> 3mm) iz razloga što se oslanja na zvuk, a ne na laser.

Slika 6 - Ultrazvučni senzor HC-SR04

2.4 Kamera - OpenMV Cam H7

Za vizuelno praćenje, kontrolu eksternih uređaja i motornog štita, te identifikaciju oštećenja na predmetima s pomoću mašinskog učenja koristi se OpenMV Cam H7 kamera. Naime, OpenMV Cam H7 i slične proizvode nazivaju još i „Arduino of Machine Vision" zbog brojnih mogućnosti kada je u pitanju mašinsko učenje i programiranje koristeći MicroPython kao efikasnu implementaciju Python-a namjenjenog za mikrokontrolerske ploče.

OpenMV Cam H7 je mikrokontrolerska ploča bazirana na ARM Cortex M7 (STM32H7) procesoru koji radi na 480 MHz, sa 1MB statičkog RAM-a (SRAM) i 2MB fleš memorije. Na svim PIN-ovima izlazni napon je 3.3V, dok su isti tolerantni na 5V. Mikrokontroler (MCU) ima ugrađene slijedeće interfejse: UART (x2), I2C (x2), SPI (x1), CAN (x1), te TIM/PWM (x3). Pored toga posjeduje 12 bitni ADC/DAC, 3 PIN-a za kontrolu servo motora i interapcije i PWM na svim PIN-ovima (10 I/O PIN-ova), te konektor za 3.7V litij-polimer bateriju (punjivu bateriju). OpenMV Cam H7 dolazi sa ugrađenom RPC (Remote Procedure Call) bibliotekom koja omogućava vezu između mikrokontrolerske ploče i računara

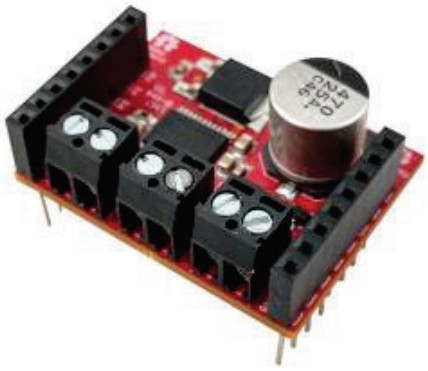
(Desktop PC), malih računara (npr: RaspberryPI), te drugih mikrokontrolerskih ploča kao što su Arduino ili ESP8266.

Kamera je bazirana na OV7725 senzoru koji pruža mogućnost kreiranja slika rezolucije od 640x480 pri brzini od 75 FPS (frejmova po sekundi), pa sve do 150 FPS kada je u pitanju rezolucija manja od 320x240. OpenMV Cam H7[1] pruža brojne mogućnosti, a neke od tih su: TensorFlow Lite, detekciju promjena na slikama (Frame Differencing), praćenje boja i markera (Color and Marker Tracking), detekciju lica i osoba (Face and Person detection), detekciju 2D barkodova (2D code detection), detekciju geometrijskih oblika (Line, Circe and Rectangle detection), slikanje i snimanje (Image Capturing and Video recording), itd.

U vrijeme pisanja ovog teksta OpenMV Cam H7 kamera više nije u prodaji. Nasljednik ove kamere je novija i bolja verzija (OpenMV Cam H7 R2[2]) čija najvažnija značajka se ogleda u novom kamera modulu/komponenti MT9M114.

2.4.1 Motorni štit za OpenMV kameru

Motorni štit (motor shield) koristi 5V regulator za napajanje OpenMV kamere, kao i driver za napajanje dva motora koja ne troše više od 2A struje. Koristi dva PIN-a za H-Bridge (H-most), te po jedan dodatni PIN za regulaciju PWM signala.



Slika 7 - OpenMV Motor Shield

[1] OpenMV H7 offical page - https://openmv.io/collections/cams/products/openmv-cam-h7
[2] OpenMV H7 R2 offical page - https://openmv.io/collections/cams/products/openmv-cam-h7-r2

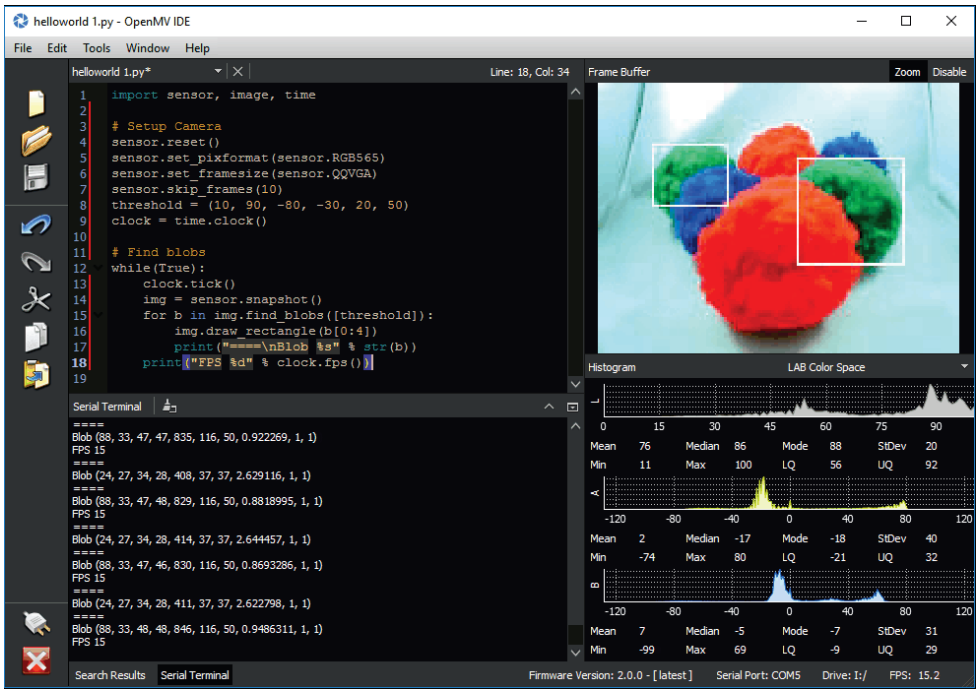
3. Softverski dio projekta

3.1 MicroPython - Python za mikrokontrolere

Kontrola uređaja kao i sam algoritam za prepoznavanje je implementiran koristeći MicroPython, kako ga neki nazivaju i skriptnim jezikom visokog nivoa. U osnovi MicroPython je mikro Python standardna biblioteka * i optimiziran je kako bi radio na hardveru mikrokontrolera. Iako kompaktne veličine (dovoljno je 256kB prostora i 16kB RAM-a), MicroPython je puni Python kompajler i runtime.

3.2 OpenMV IDE

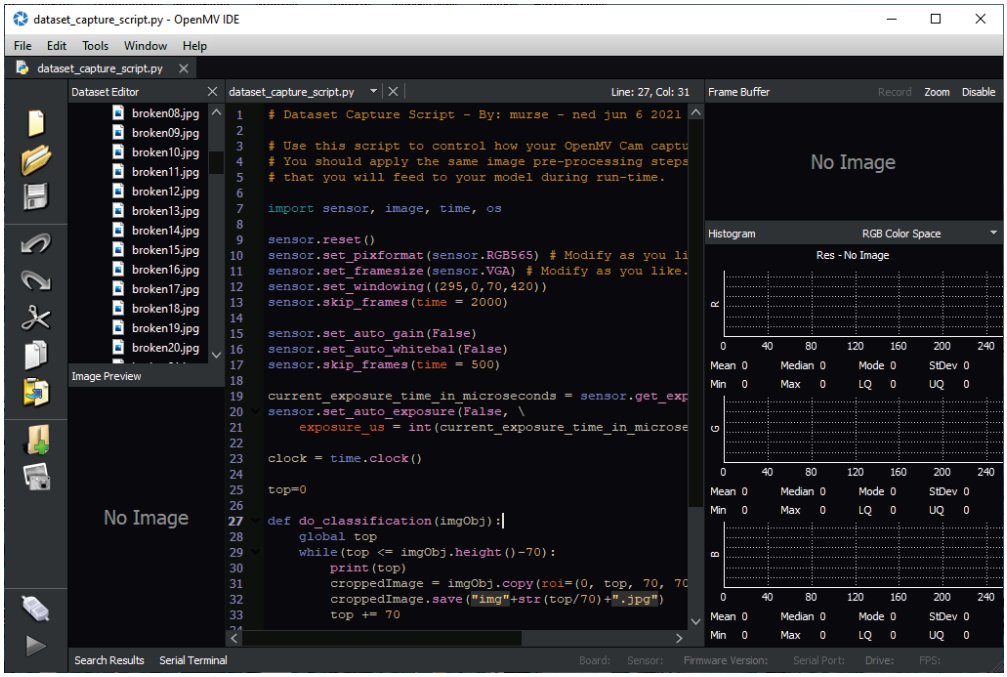
OpenMV IDE je integrisano razvojno sučelje prvenstveno namjenjeno za programiranje OpenMV mikrokontrolerske ploče, te pored toga sadrži i Dataset editor za kreiranje klasa i uzoraka za potrebe mašinskog učenja. Dataset editor pruža mogućnost *upload*-a podataka na Edge Impulse portal gdje je moguće uraditi treniranje modela koristeći vizuelno okruženje bez potrebe za kodiranjem.



Slika 8 - OpenMV IDE

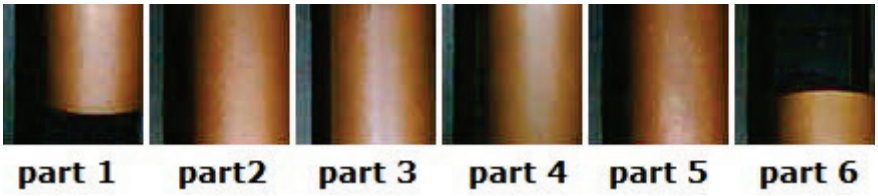
U svrhu treniranja modela za prepoznavanje oštećenja na industrijskim predmetima upravo je korišten Dataset editor OpenMV IDE-a gdje je su kreirane

4 klase pod nazivima: *bad*, *broken*, *good* i *neutral*, a putem Edge Impulse cloud-based alata izvršeno je treniranje modela pri tome pazeći da krajnji rezultat odgovara za OpenMV Cam H7 verziju mikrokontrolerske ploče.



Slika 9 - OpenMV Dataset editor

Izgled uzorka se može vidjeti na slici 7, a kreiranje uzoraka predmeta je urađeno izmjenjenim kôdom algoritma za prepoznavanje oštećenja. Uređaj bi pozicionirao predmet ispod kamere, uslikao istu, te od jedne slike predmeta kreirao 6 sličica koje su kasnije razvrstane po klasama u datasetu. Sve operacije do razvrstavanja radi algoritam kreiran za te svrhe, dok razvrstavanje se radi „ručno" od strane korisnika. Pored toga postoje primjeri za sve mogućnosti koje pruža OpenMV kamera, te od verzije 2.8.1 i primjere za Nano 33 BLE Sense, Nano RP2040, i Portenta H7 mikrokontrolerske ploče.



Slika 10 - Primjer uzorka predmeta

3.3 Inicijalizacija OpenMV kamere

Kao prvo potrebno je inicijalizirati OpenMV kameru (*sensor.reset()*), te nakon toga postavljamo format piksela na GRAYSCALE gdje ćemo putem kamere dobijati slike u nijansama sive boje (1 kanal, vrijednost piksela se kreće od 0 – 255, gdje je pri vrijednosti 0 piksel potpuno crn, dok pri vrijednosti od 255 poprima potpuno bijelu boju).

```
import time, utime, pyb, sensor, image, os, tf

from pyb import Pin

sensor.reset()
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.VGA) # 640x480
sensor.set_windowing((295,0,70,420)) # crop to 70x420
sensor.skip_frames(time = 3000) # stabilizacija opcija na kameri
```

Slika 11 - Inicijalizacija OpenMV kamere

Nakon toga postavljamo *frame size* tj. veličinu video okvira modula kamere, a ona iznosi 640 x 480 piksela. Sa ovim video okvirom, iako dobijemo dosta kvalitetnu sliku određenog objekta ispod modula kamere, zahvatamo i nepotreban prostor okolo industrijskog objekta koji posmatramo. Kako bi smo prevazišli taj problem koristimo tehniku isjecanja slike (crop) na približnu rezoluciju koju odredimo testiranjem, a ona u ovom slučaju iznosi 70 x 420 piksela. Funkcija *set_windowing* zaprima parametar *roi* (region of interest) oblika {x,y,w,h}, gdje **x** i **y** predstavljaju koordinate trenutnog frame-a, a **w** i **h** su vrijednosti broja piksela po širini i visini. U našem slučaju isjecanje počinje pri vrijednosti x= 295 piksela horizontalno, tj. 295 piksela od gornjeg lijevog ugla frame-a. Nakon što apliciramo zadane vrijednosti, sačekamo 3 sekunde da se senzor kamere stabilizira sa zadatim vrijednostima. U tom periodu kamera neće davati nikakav video output.

Prilikom testiranja video outputa utvrđeno je da postoji problem refleksije svjetlosti LED sijalica na predmetima koji su u tom trenutku promatrani, te je pokušano sa plastičnim zaštitnim staklom koje se koristi na klasičnim kućanskim LED sijalicama, ali je navedena zaštita previše prigušivala svijetlost.

Rješenju se pristupilo korištenjem metoda koje je pružao OpenMV firmware putem sopstvene python biblioteke.

```
sensor.set_auto_gain(False)
sensor.set_auto_whitebal(False)
sensor.skip_frames(time = 2000)

current_exposure_time_in_microseconds = sensor.get_exposure_us()
sensor.set_auto_exposure(False, \
    exposure_us = int(current_exposure_time_in_microseconds * 0.6))
```

Slika 12 - Postavke ekspozicije kamere

Prilikom boot-a, kamera automatski uključi tri opcije: *white balance*, *auto gain* i *exposure.* White balance je opcija koja treba da objekte, koji su bijele boje u stvarnom svijetu, prikaže bijelim i putem kamere, dok auto gain predstavlja metodu poboljšanja slike pri odsustvu svijetlosti u stvarnom svijetu. Exposure (izloženost) predstavlja količinu svijetlosti koja dopire do senzora kamere u određenom vremenskom periodu. Što je senzor više izložen svijetlosti, tj. što je više blenda senzora otvorena to više svijetlosti senzor zaprima. U tom slučaju potrebno je odrediti vremenski period tokom kojeg će senzor zaprimati svijelost. To ujedno i predstavlja glavni parametar koji je potrebno izmjeniti u odnosu na onaj što kamera postavi prilikom startanja tj. boot-a. Formula za ekspoziciju je:

$$H = E * t \tag{1}$$

, gdje parametar E predstavlja količinu izloženosti svijetlosti, dok parametar t predstavlja vrijeme izloženosti svijetlosti u mikrosekundama (po OpenMV standardu). Obzirom da je trenutno vrijeme izloženosti svijetlosti predugo isto je reducirano umnožkom od 0.6 mikrosekundi. Nakon osnovnih postavki samog uređaja, deklarišemo dvije varijable u koje pohranjujemo putanju do našeg modela tj. naziv datoteke, te koje smo klase definisali prilikom kreiranja modela, a koje su pohranjene u datoteku labels.txt.

```
net = "trained.tflite"
labels = [line.rstrip('\n') for line in open("labels.txt")]
```

Slika 13 – Putanje do modela mreže i klasa

3.4 Upravljanje DC motorima

Prethodno smo naveli da se sa DC motorima upravlja koristeći tehniku *modulacije širine impulsa* (Pulse Width Modulation – PWM). U osnovi PWM signali predstavljaju digitalne signale koji se pretvaraju u analogne ili drugačije rečeno vrši se konverzija električne energije u kinetičku energiju, te shodno tome PWM koristimo kada želimo da upravljamo sa analognim uređajima. Tehnika modulacije širine impulsa podrazumjeva pulsiranje ili konstantno isključivanje i uključivanje jednosmjernog napona pri određenoj frekvenciji i sa određenom dužinom trajanja (duty cycle). Za pogon oba motora kreiran je Timer objekat sa identifikacionim brojem 4 (Timer4) koji je potrebno da radi na 50Hz. Frekvencija od 50Hz pri pogonu servo motora se koristi još od 90-tih godina prošlog vijeka i predstavlja stari standard koji je usvojen od strane inžinjera kao najpouzdaniji pri radu servo motora. Timer-i predstavljaju brojače koji broje impulse koji su generirani od strane oscilatora (kvarcni kristal) koji je sastavni dio tog timera i koji svojim vibriranjem (savija se i skuplja) povećava ili smanjuje električnu struju tj. oscilira u pravilnim valovima. Svaki timer ima samo jedan brojač, ali ga dijeli sa jednim ili više kanala koje je moguće čitati na jednom ili više PIN-ova.

OpenMV Cam H7 mikrokontrolerska ploča pruža generiranje PWM signala na tri PIN-a, a u našem slučaju za korištenje PWM signala pri kontroli servo motora koji pogoni remen koristimo PIN-7 (timer 4 na kanalu 1), dok za kontrolu servo motora za razdvajanje predmeta koristimo PIN-9 (timer 4 na kanalu 3). Radni ciklus (duty cycle) timer-a na kanalu 1 iznosi 50%, što nam govori da je 50% vremena strujno kolo otvoreno (nema napona), a 50% vremena je zatvoreno (postoji napon). Do ovog podatka se došlo prilikom testiranja brzine okretaja motora, a što predstavlja optimalnu vrijednost kako bi motor uopće mogao raditi. Obzirom da motor za pogon remena radi na 24V napona, te mu je potrebna veća jačina struje, korišten je motorni štit koji dozvoljava veći protok struje, i to do 2A.

OpenMV *motor shield* je baziran na TB6612 driveru, a inicijaliziramo ga shodno uputama proizvođača na već predefiniranim PIN-ovima (P3 i P2). Oba izlaza su konfigurisana da rade u output načinu rada sa *push-pull* konfiguracijom sa isključenim pull-up/down otpornicima na zadanim PIN-ovima.

```
# pins for control to drive the H-Bridge sides
pinDir0 = pyb.Pin('P3', pyb.Pin.OUT_PP, pyb.Pin.PULL_NONE)
pinDir1 = pyb.Pin('P2', pyb.Pin.OUT_PP, pyb.Pin.PULL_NONE)
```

Slika 14 - H-Most konfiguracija

Push-pull konfiguracija podrazumjeva tehniku korištenja dva tranzistora, s tim da je jedan povezan na GND, dok je drugi povezan na VIN (napon). Tranzistor (PMOS) je povezan na VIN i kada je aktiviran pušta protok struje od ulaza (VIN) prema izlaznom pinu (OUPUT). U tom trenutku drugi tranzistor (NMOS) nije aktivan. U slučaju da je NMOS tranzistor aktiviran, tada je PMOS deaktiviran i struja teče od izlaznog pina prema GND.

Da bi PWM mogao raditi potrebno je da kreiramo *timer* objekat putem kojeg vršimo kontrolu brzine okretanja motora. Kao što je ranije naglašeno servo motori rade na 50Hz, tako da je timer (4) inicijaliziran sa navedenom frekvencijom.

```
# timer object running at 50 Hz (servo requires 50Hz)
tim = pyb.Timer(4, freq=50)

# initialize motor and servo timer channels for pwm output
motor_ch = tim.channel(1, pyb.Timer.PWM, pin=pyb.Pin("P7"), pulse_width_percent=50)
servo_ch = tim.channel(3, pyb.Timer.PWM, pin=pyb.Pin("P9"), pulse_width_percent=100)
```

Slika 15 - PWM konfiguracija

Nakon toga kreirana su dva PWM kanala (1 i 3) na PIN-ovima P7 i P9, gdje za motor koji pogoni remen postavljen na 50% PWM signala, dok servo motor za odvajanje predmeta sa remena na 100% PWM signala.

3.5 Upravljanje ultrasoničnim senzorom

Za pravilan rad ultrasoničnog senzora, isti se napaja sa 5V, te spojen sa mikrokontrolerskom pločom na dva PIN-a: PIN 5 i 4. PIN-5 služi kao *trigger* i inicijaliziran je kao *output* PIN sa *push-pull* kontrolom, a inicijelno u PULL_DOWN modusu rada, dok PIN-4 je inicijaliziran samo kao *input*, a koristimo ga da bi zaprimili *echo* signale/refleksiju koji su prethodno izgenerisani na *trigger* PIN-u. Ultrasonični senzor se koristi za detekciju predmeta na remenu/traci, te pri detekciji predmeta vrši se vremensko kašnjenje od 620ms (vrijednost dobivena testiranjem), te nakon toga motor se zaustavlja pri tome pozicionirajući predmet ispod objektiva kamere.

```
# trigger and echo pins of ultrasonic sensor
trig_pin = Pin('P5', Pin.OUT_PP, Pin.PULL_DOWN)
echo_pin = Pin('P4', Pin.IN, Pin.PULL_NONE)
```

Slika 16 - Konfiguracija ultrasoničnog senzora

3.6 Treniranje modela

Prikupljanje podataka je urađeno na način da je postojeći python kôd modifikovan kako bi uređaj umjesto detekcije oštećenja služio za slikanje predmeta, modifikaciju slike i njenu pohranu u prethodno predviđenu klasu. Prikupljanje podataka (slika) vrši se koristeći Dataset editor koji je sastavni dio/modul OpenMV IDE-a. Kreirane su 4 klase: good, bad, neutral i broken. Sakupljeno je ukupno 1687 uzoraka. Proces sakupljanja se radi na slijedeći način:

1. Nakon pozicioniranja predmeta, uslika se kompletan predmet u rezoluciji 420x70px
2. Potom se slika „izreže" na 6 slika rezolucije 70x70px
3. Slike se pohrane na disk u direktorij koji je prethodno kreiran kao klasa kroz dataset editor
4. Na kraju se vrši manuelna selekcija slika (odgovarajuće slike se ostavljaju, dok druge uklanjaju)

Nakon što smo kreirali potreban dataset isti se koristi putem Edge Impulse web portala gdje ćemo na jednostavan način uraditi treniranje modela. Edge Impulse je, kako smo već naveli, *cloud-based* razvojna platforma za mašinsko učenje koja cilja *embedded* uređaje. Edge Impulse je baziran na TensorFlow eko-sistemu putem kojeg je moguće raditi treniranje i kreiranje (*deployment*) *deep-learning* modela za ugrađene sisteme.

Prvi korak kod treniranja modela jeste *upload* dataset-a i kreiranje potrebnih klasa. U slučaju da se *upload* podataka radi direktno iz OpenMV IDE-a u tom slučaju klase će biti automatski kreirane, u protivnom potrebno je kreirati klasu i nakon toga uraditi *upload* podataka.

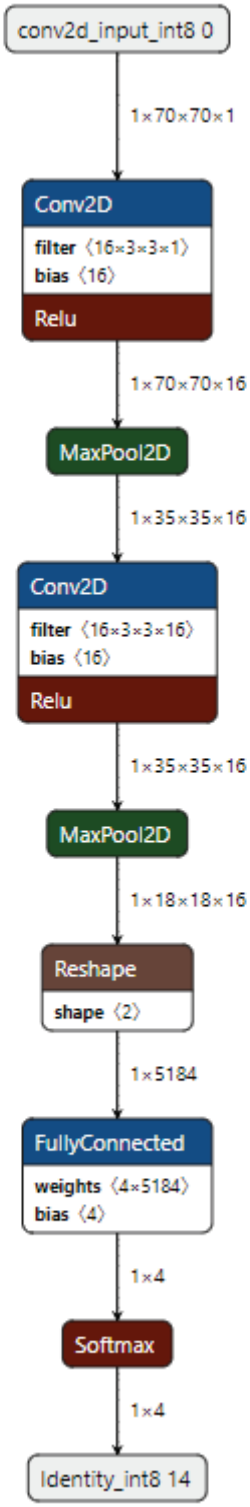
Po *upload*-u podataka potrebno je kreirati/dizajnirati *impuls*. Impuls je u osnovi korak u kojem se uzimaju neobrađeni podatci (*raw data*), primjenjuje se procesiranje signala (*signal processing*) kako bi izdvojili obilježja podataka (*data features*), a zatim se primjenjuje neki od algoritama/metoda za klasifikaciju novih podataka. Uprošteno rečeno vršimo modifikaciju slika (veličina slike, način promjene veličine), biramo NN algoritam (u našem slučaju je klasifikacijska

metoda implementirana korištenjem Keras biblioteke), te potom snimimo impuls i otpočnemo treniranje modela.

Kako se model u osnovi trenira?

Naš model predstavlja sekvencijalni model, a to znači da postoji niz slojeva (*layers*) gdje svaki sloj ima jedan ulazni tenzor i jedan izlazni tenzor. Tenzor predstavlja generalizaciju određenog matematičkog objekta kao što su vektori ili matrice. U osnovi tenzor je reprezentacija, npr.: vektora koji se označava sa (x1,x2), a ne formalno sa (X, Y). Sloj u modelu predstavlja strukturu koja prihvata podatke iz prethodnog sloja i proslijeđuje ih slijedećem sloju. Postoje tri tipa slojeva u neuronskim mrežama: ulazni sloj (input layer), skriveni sloj (hidden layer), izlazni sloj (output layers). Ulazni sloj je sloj koji zaprima inicijalne podatke. Skriveni sloj zaprima informacije od ulaznog sloja i odgovoran je za kompletan proračun. Izlazni sloj daje rezultate po osnovu ulaznih podataka koje je zaprimio od skrivenog sloja. Sloj se u osnovi sastoji od čvorova (*nodes*). Svaki čvor je moguće poistovjetiti sa njegovim biološkim rođakom - neuronom. Konekcije između samih čvorova možemo poistovjetiti sa sinapsama. Putem njih se prenose parametri tj. ulazno/izlazne vrijednosti između čvorova. Svaki čvor zaprima ulazne vrijednosti parametara i daje izlazne vrijednosti uz primjenu funkcija (softmax, relu, sigmoid, i sl) koje se još nazivaju i aktivacijskim funkcijama.

Struktura naše neuronske mreže prikazana je na slici 8.



Slika 8 - Vizuelni prikaz NN

3.7 Algoritam za prepoznavanje oštećenja

Algoritam radi na način da se prvo inicijalizira servo motor za odvajanje predmeta na početnu poziciju (slika 17), a nakon toga ultrasonični senzor shodno datom objašnjenju prethodno u tekstu. Kod za inicijalizaciju je prikazan na slici 18.

```
def initServoPos():
    global initServo
    if(initServo == 0):
        servo_ch.pulse_width(1300)
        initServo = 1
```

Slika 17 - Inicijalizacija motora za odvajanje

```
# stabilize pin for 5us
# start trigger for 10us
def startScan():

    trig_pin.value(0)
    pyb.udelay(5)
    trig_pin.value(1)
    pyb.udelay(10)
    trig_pin.value(0)
```

Slika 18 - Startna inicijalizacija HC-SR04P senzora

Pribavlja se distanca koju je senzor zabilježio, te se vrši provjera da li postoji prepreka između senzora i graničnika na uređaju. Prethodno smo naveli da senzor zaprima vrijednost na *echo* pinu, tako da se putem metode (slika 19) vrši kalkulacija distance od predmeta do senzora.

```
def getDistance():

    global end_time, distance, start_time

    while(echo_pin.value()==0):
        start_time = pyb.micros()

    while(echo_pin.value() == 1):
        end_time = pyb.elapsed_micros(start_time)

    distance = end_time * 0.5 * 0.0343

    return distance
```

Slika 19 - Izračun distance (HC-SR04P)

Ako vrijednost distance iznosi između 4 i 6 uređaj šalje *stop* signal (slika 20) servo motoru koji pogoni remen/traku.

```
# start motor
def motorStart():
    motor_ch.pulse_width_percent(30)

#stop motor
def motorStop():
    motor_ch.pulse_width_percent(0)
```

Slika 20 - Start/stop kretanja motora

Po zaustavljanju uređaj radi *capturing* predmeta ispod objektiva i vrši prvu klasifikaciju. Na ovom mjestu se vrši provjera da li vrijednost rezultata klasifikacije odgovara klasi „neutral", tj. da li je predmet ispravno pozicioniran ispod objektiva kamere. U slučaju da postoji prazan prostor prije početka predmeta ili s njegovog kraja, uređaj će pomjerati predmet naprijed ili nazad putem metoda prikazanih na slici 21, te vršiti svaki put klasifikaciju dok se predmet nepozicionira ispod objektiva kamere u cijelosti.

```
def moveMotorBack():
    pinDir0.value(0)
    pinDir1.value(1)
    motorStart()
    print("move back")

def moveMotorForward():
    pinDir0.value(1)
    pinDir1.value(0)
    motorStart()
    print("move forward")
```

Slika 21 - Određivanje pravca okretanja motora

Nakon pozicioniranja predmeta vrši se klasifikacija o tome da li se radi o slomljenom (broken), neispravnom (bad) ili ispravnom predmetu.

Ako vrijednost klasifikacije pokaže neku od prethodno navedenih vrijednosti uređaj šalje *start* signal (slika 20) o pokretanju servo motora, uz prethodno definisano kašnjenje, predmet dolazi do servo motora koji u osnovi vrši klasifikaciju tj. odvajanje na ispravne i neispravne predmete (slika 22).

```
# activate servo for bad
def servoSignalBad():
    pyb.delay(500)
    servo_ch.pulse_width(1500)
    moveMotorForward()
    motorStart()
    pyb.delay(1200)
    servo_ch.pulse_width(3000)

# activate servo for good
def servoSignalGood():
    pyb.delay(500)
    servo_ch.pulse_width(3000)
    moveMotorForward()
    motorStart()
    pyb.delay(990)
    servo_ch.pulse_width(1500)
```

Slika 22 - Metode za odvajanje predmeta sa trake

```
def do_classification(imgObj):
    global top, scoreBad, scoreBroken, scoreGood, scoreNeutral, c, f
    scoreBad = 0
    scoreGood = 0
    scoreNeutral = 0
    scoreBroken = 0
    top = 0
    c = 0
    while(top <= imgObj.height()-70):
        croppedImage = imgObj.copy(roi=(0, top, 70, 70))
        #croppedImage.save("img"+str(time.ticks_us())+".jpg")
        obj = model.classify(croppedImage, min_scale=1.0, scale_mul=0.8, x_overlap=0.5, y_overlap=0.5)

        scoreBad += obj[0].output()[0]
        scoreBroken += obj[0].output()[1]
        scoreGood += obj[0].output()[2]
        scoreNeutral += obj[0].output()[3]

        tempScores = [scoreBad,scoreBroken,scoreGood,scoreNeutral]

        # neutral?
        if(tempScores.index(max(tempScores)) == 3):
            break

        # broken?
        if(tempScores.index(max(tempScores)) == 1):
            break

        top+=70
        c+=1

    scores = [scoreBad,scoreBroken,scoreGood,scoreNeutral]
    #sprint(scores)
    max_value = max(scores)
    max_index = scores.index(max_value)
    label = labels[max_index]
    #print(label)
    return label
```

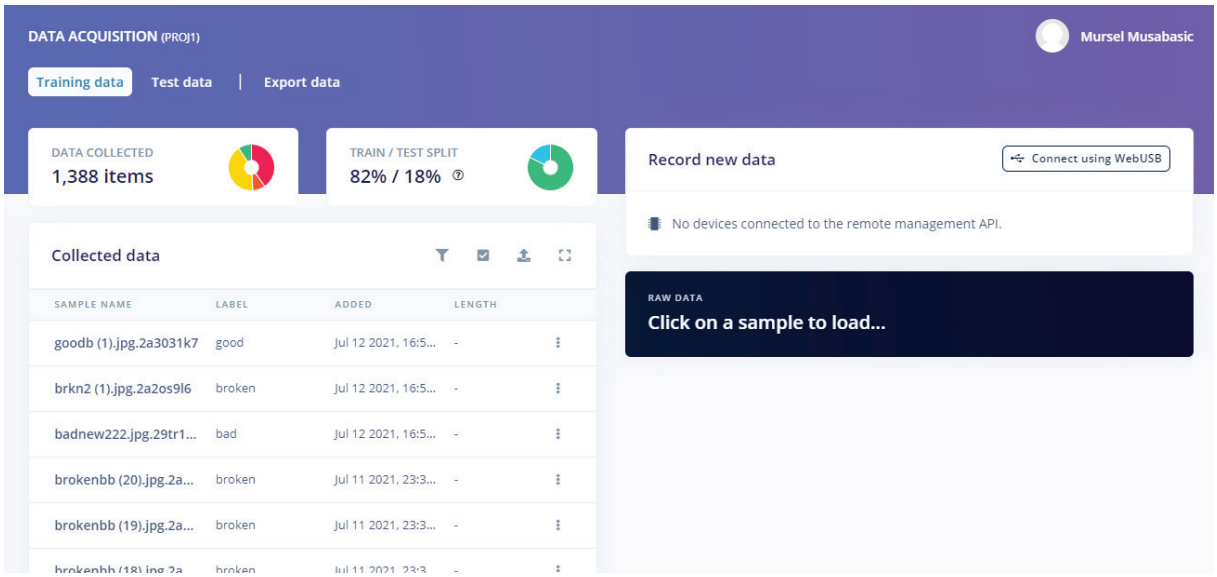
Slika 23 - Metoda za klasifikaciju predmeta

3.8 Edge Impulse web platforma

Za lakše učenje modela korištena je web platforma naziva Edge Impulse od isto imene kompanije koja je osnovana 2019. godine od strane Zach Shelby and Jan Jongboom. Web platforma je zamišljena na način da se kompletan proces učenja modela, od prikupljanja do podataka, izvršava na serveru na vrlo intuitivan način. Pored učenja modela moguće je vršiti, kako prikupljanje, tako i testiranje modela korištenjem uređaja koji egzistiraju na strani klijenta/korisnika kao što su razne razvojne ploče (OpenMV putem dataset editora), mobilnih telefona (web camera/audio), kompjutera (web camera/audio), te drugih brojnih uređaja koji su indirektno podržani, ali i cloud data servisa.

U našem slučaju za učenje modela sakupljeno je 1687 slika, gdje je od toga na *train* podatke odvojeno 1350 slika, a na *test* podatke odvojeno je 337 slika, ili u omjeru 80% / 20%. Naknadnim testiranjem i analizom ovaj omjer je postavljen na 82% / 18%.

Prilikom upload-a podataka moguće je odabrati da li želite da sistem koristi omjer 80/20, ili da korisnik odabere da li se radi o *training* ili *testing* podacima. Pored odabira tipa seta podataka moguć je odabir i imena klase navedenih podataka (bad, good, itd).



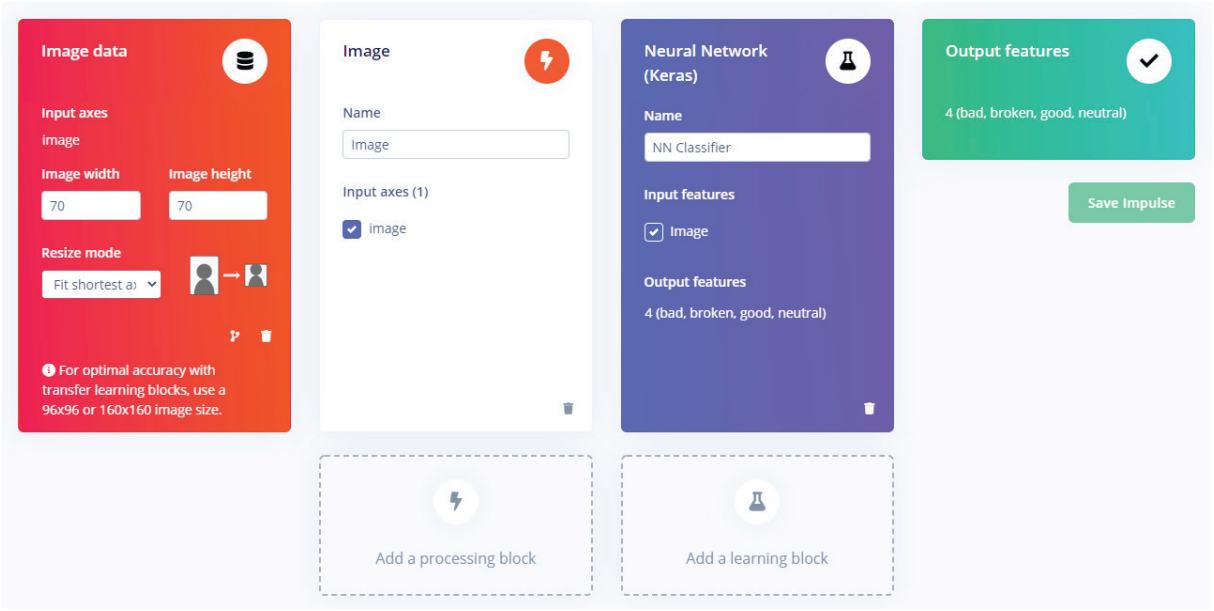
Slika 24 - Izgleda interfejsa akvizicije podataka

Slijedeći korak jeste kreiranje tzv. impulsa. Impuls u osnovi predstavlja nekoliko koraka koje je potrebno provesti kako bi model bio spreman za učenje.

Ako postoji potreba da se veličina slika izmjeni to je moguće uraditi na ovom koraku u sklopu kartice „Image data", zajedno sa odabirom *resize* moda. Posle toga se odabere tzv. *processing block*, tj. o kakvom tipu ulaznih podataka se radi. Obzirom da vršimo procesiranje nad slikama odabran je Image kao tip podatka (slika), a moguće je odabrati između ostalog i neke druge kao što su audio podaci, spektrogram, podatke spektro analiza ili čak custom podatke.

Zadnji korak u kreiranju impulsa jeste odabir algoritma za učenje. U našem slučaju to je klasifikacija slika bazirana na Keras biblioteci. Sistem po osnovu prethodno unesenih podataka sam pruža odabir (jedan ili više) algoritma za učenje modela.

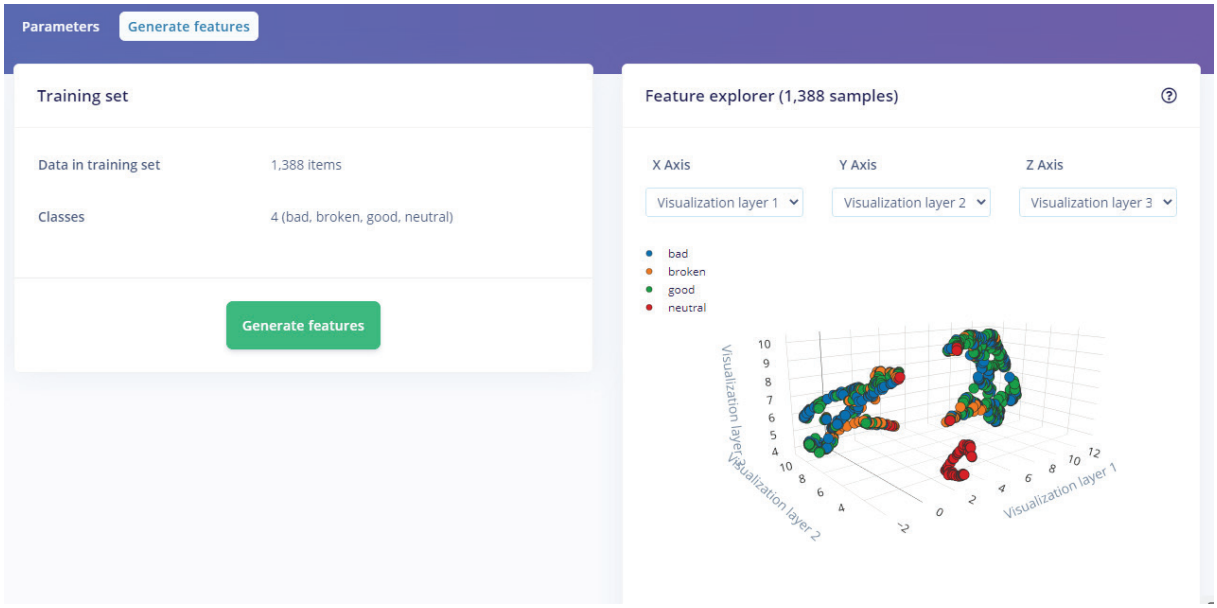
Na krajnjoj desnoj kartici prikazane su klase koje će uči u proces učenja modela.



Slika 25 - Kreiranje Impulse-a

Po kreiranju impulsa prelazi se na korak gdje će se aplicirati prethodno definisane opcije, a to su: izmjena veličine slika (resizing), normalizacija određenih podataka i primjena dubine boja (RGB ili grayscale), te po završetku će se kreirati 3D vizualni prikaz kompletnog dataset-a. Podaci su u sklopu

prikaza podijeljeni u klastere, te je moguće ranije uvidjeti da li su dovoljno razdvojeni kako bi lakše bili naučeni od strane modela.



Slika 26 - Generiranje feature-a

Sa svim prethodno procesiranim podacima slijedeći korak jeste treniranje/učenje neuronske mreže. Postavke parametara je moguće uraditi na dva načina, i to putem GUI-a ili ručno kroz python kod.

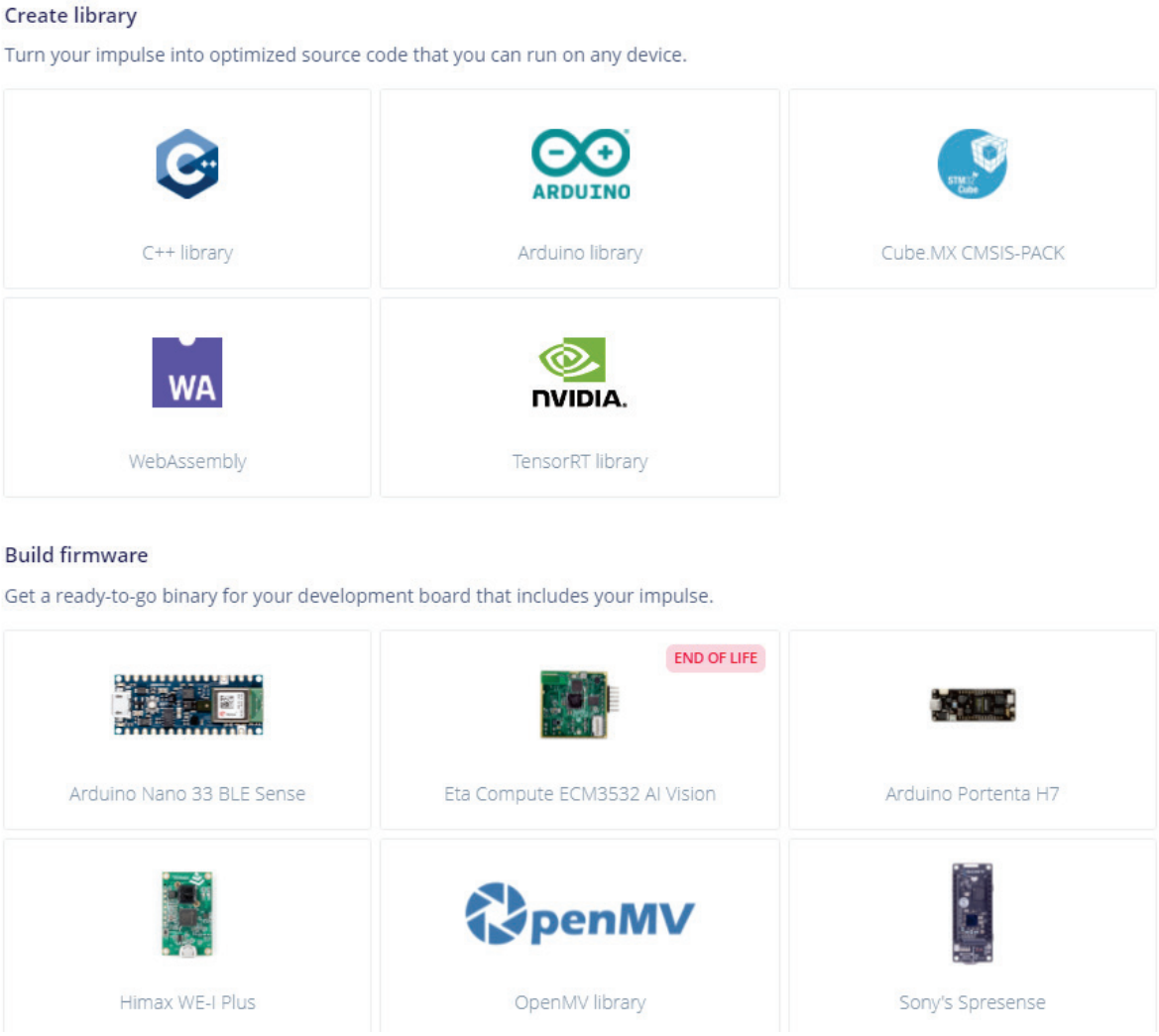
Slijedeći parametri su dostupni za odabir kroz GUI: broj iteracija, stopa učenja, procenat podataka za validaciju, te slojevi neuronske mreže. U vrijeme pisanja ovog rada dodana je opcija *auto-balance dataset*-a koja služi za spriječavanje overfitting-a ako postoji slučaj da za određenu klasu nema dovoljno podataka.

Nakon što proces treninga završi rezultati preciznosti modela će biti prikazani na istoj stranici zajedno sa matricom konfuzije/zabune.

Po završenom treningu modela isti je moguće testirati u sklopu sekcije „Model testing". Testiranje modela je moguće uraditi u realnom vremenu, ili naspram testnih podataka iz dataset-a.

Jedna od opcija web platforme jeste i „deployment“. Tu zaista postoje brojne mogućnosti, od generisanja izvornog kôda, do izrade (build) firmware-a.

Ovdje je moguće pokrenuti klasifikaciju u online modu direktno putem računara ili mobilnog telefona.



Slika 27 - Edge Impulse deployment

4. Obrazloženje algoritma sa gledišta mašinskog učenja

Computer Vision je oblast Umjetne inteligencije, te uključuje brojne zadatke kao što su Image Classification (klasifikacija slika), Object Detection (detekcija objekata), i sl. U sklopu rada korištena je klasifikacija slika koja se bazira na CNN algoritmu tj. Convolutional Neural Network. CNN je u osnovi mrežna arhitektura bazirana na deep learning-u (dubinskom učenju) gdje algoritam prima slike kao input podatke, a za razliku od drugih algoritama gdje je potrebno „ručno" podešavati filtere, u slučaju CNN-a mreža „uči" direktno iz navedenih input podataka.

Struktura CNN mreže se sastoji od nekoliko slojeva, a obrazložiti ćemo ih detaljnije osvrčući se na implementaciju algoritma u sklopu ovog rada, no za početak te nivoe možemo podijeliti na *input*, *convolution (konvolucijski)*, *pooling (nivo kompresije)*, *fully connected (potpuno povezani)* i *output*.

4.1 Input/ulazni nivo

U sklopu ovog rada kao input korištene su slike dimenzija 70x70px. Kako je prethodno objašnjeno slika dimenzija 70x420px je podijeljena na više manjih slika dimenzija 70x70px. Za input nivo dataset je podijeljen na train i test podatke, i to u odnosu 80/20, gdje je 80% podataka korišteno za *učenje* CNN algoritma, dok će se ostalih 20% koristiti za testiranje preciznosti naučenog algoritma/modela.

4.2 Konvolucijski nivo

Na koji način naš model može da prepozna oblike ili određene ivice na slici? Upravo sa konvolucijskim nivoom započinje proces učenja našeg modela. Na ovom nivou filteri se koriste kako bi se određena obilježja izdvojila iz input podataka. Filteri su obično manjih dimenzija od slika, u našem slučaju prvi konvolucijski nivo je definisan sa 16 filtera koji su veličine 3x3px, ali vrijednost padding-a je postavljena na „*same*" tako da output primjenjenog filtera daje output istih dimenzija kao i input prije filtera, a to je slika 70x70px. *Padding* je vrijednost koja služi u slučajevima kada želimo da smanjimo dimenzije slike nakon apliciranja filtera ili ostavimo ih istim kao u našem slučaju. Filteri se apliciraju na način da se postave u lijevo gornji ugao input slike i svakim

pomicanjem u desno za određeni broj piksela (*stride*) vršimo kalkulaciju gdje se dobije skalarna vrijednost. Pomicanje se radi dok se ne dođe do krajnje desne strane slike, te se pomicanje filtera uradi prema dole i postupak se ponovi ispočetka. Vrijednosti koje se dobiju zbrajaju se i proslijeđuju kao input aktivacijskoj funkciji.

Shodno parametrima filtera slijedeća formula se može primjeniti za izračun output slike/mape:

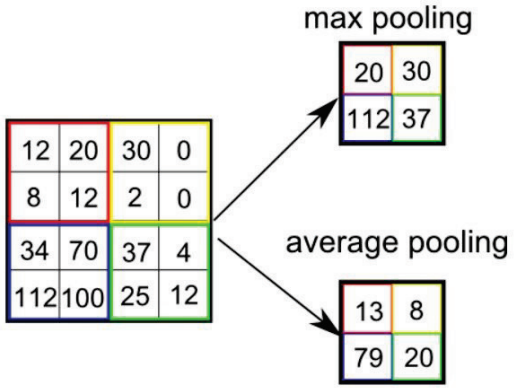
$$O_{n=\frac{I_S-F_S+2*P}{S}+1,}$$

(2)

gdje je I_s ulazna širina prethodne mape, F_s je širina trenutnog filtera, P je broj popunjavanja piksela sa nulama (*zero-padding*), te S je vrijednost koraka pomicanja filtera.

4.3 Nivo kompresije

Ovaj nivo zaprima output konvolucijskog nivoa i smanjuje/sažima ulaznu mapu. U našem slučaju dimenzije slike sa 70x70px se sažimaju na dimenzije 35x35px koristeći 2x2px filter i sa vrijednošću koraka pomicanja (stride) koji iznosi 2px. Obično se koristi *maxpooling* kao u našem slučaju gdje se uzimaju samo maksimalne dobijene vrijednosti iz prethodne mape. Bitno je napomenuti da ovaj nivo ukljanja šum (*noise*) i smanjuje broj parametara koje bi mreža trebala naučiti.



Slika 28 - Tipovi sažimanja[3]

[3] Types of pooling (https://miro.medium.com/max/2400/1*KQIEqhxzICU7thjaQBfPBQ.png)

4.4 Potpuno povezani nivo

Nakon što je CNN algoritam izdvojio/naučio određene parametre, iste proslijeđujemo potpuno povezanom nivou gdje se generira krajnji rezultat mreže. Obzirom da su prethodni nivoi (konvulacijski nivoi) generirali 2D podatke (matrice), poptupno povezani nivo prihvaća samo 1D podatke. Potpuno povezani nivo sadrži određene nivoe kao što su *Flatten*, *Dropout* i *Dense*. *Flatten* koristimo kako bi 2D podatke pretvorili u 1D vektor. *Dropout* nivo se koristi za uklanjanje mogućeg *overfitting*-a, dok *Dense* nivo određuje koju ćemo aktivacijsku funkciju koristiti u svrhu klasifikacije, te kojoj će klasi pripadati rezultat modela.

Pod *overfitting*-om se misli na model koji je naučen u toj mjeri (prenaučen) da može prepoznati samo one podatke iz jednog dataset-a, ali gubi preciznost na potpuno novim ulaznim podatcima, tj. otpočne pogrešnu klasifikaciju test podataka.

No prije same klasifikacije podaci prođu kroz dvije operacije gdje je potrebno uraditi, kao prvo linearnu transformaciju, te nakon toga ne-linearnu transformaciju.

Linearna transformacija se radi koristeći slijedeću formulu:

$$z = W^T X + b \quad (3)$$

gdje je ***X*** input koji predstavlja 1D vektor, ***W*** je težinska tačka, a ***b*** je tzv. bias vrijednost koja je u osnovi konstanta.

Početne vrijednosti težinskih tačaka se određuju slučajnim odabirom, kao i za konstantu b (bias). Vrijednosti za *W* i *b* se iskazuju u obliku matrica. Primjer u nastavku teksta pokazuje matricu (m, n) koja je izlazna vrijednost konvulacijskog nivoa, gdje vrijednost ***m*** predstavlja broj parametara (features), a ***n*** predstavlja broj neurona u mreži. Kao prvo podatke 2D matrice iz konvulacijskog nivoa pretvaramo u 1D niz/vektor.

$$\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} = X$$

(4)

Ako uzmemo za primjer da postoje 3 neurona onda će shodno tome matrica težinskih tačaka imati formu (4,3).

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix}$$

(5)

Prethodno definisane matrice uvrstimo u jednačinu (1), tako da dobijemo:

$$Z = \begin{bmatrix} w_{11} & w_{21} & w_{31} & w_{41} \\ w_{12} & w_{22} & w_{32} & w_{42} \\ w_{13} & w_{23} & w_{33} & w_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + BIAS[a_n, b_m]$$

(6)

Množenjem matrice *W* i vektora *X* dobijemo finalni oblik za $Z_{(3x4)}$:

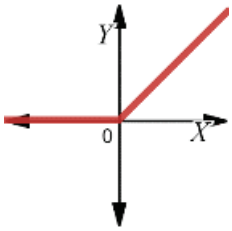
$$Z_{(3x4)} = \begin{bmatrix} w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + w_{41}x_4 \\ w_{12}x_1 + w_{22}x_2 + w_{32}x_3 + w_{42}x_4 \\ w_{13}x_1 + w_{23}x_2 + w_{33}x_3 + w_{43}x_4 \end{bmatrix}$$

(7)

Nakon linearne transformacije slijedeći korak je nelinearna transformacija koja predstavlja krajnji korak u procesu *propagacije naprijed (forward propagation)*. Kako bi dodali podacima nelinearnost služimo se tzv. aktivacijskim funkcijama. Nad dobivenom sumom neurona potrebno je aplicirati neku od aktivacijskih funkcija. Naime, na svaki konvulacijski nivo naše neuronske mreže se aplicira neka od aktivacijskih funkcija, a to zavisi od problema koji se riješava. U našem slučaju koristimo ReLu (Rectified Linear Unit) funkciju koja je definisana slijedećim izrazom:

$$f(x) = \max(0, x) = \begin{Bmatrix} 0, za\ x < 0 \\ x, za\ x \geq 0 \end{Bmatrix}$$

(8)



Slika 29 - ReLu funkcija[4]

Derivacija ReLu funkcije:

$$f(x) = \max(0, x) = \begin{cases} 0, za\ x < 0 \\ 1, za\ x \geq 0 \end{cases}$$

(9)

ReLu funkcija je trenutno najkorištenija aktivacijska funkcija u neronskim mrežama. Kod ReLu funkcije možemo uvidjeti da je nula granica odluke, te da za vrijednost *x* veću od nule rezultat funkcije je uvijek 1, a vrijednost *x* manju od nule vrijednost funkcije je 0 (nula). Za *x = 0* funkcija je nedefinisana, jer njena lijeva i desna derivacija nisu jednake.[5]

4.5 Output nivo

Nakon potpuno povezanog nivoa krajnji output našeg modela donosi *softmax* funkcija. Softmax funkcija se koristi kod višeklasne klasfikacije kao što je to slučaj kod našeg modela gdje svaka od 4 klase, kao output modela, poprimaju određenu vrijednost. Suma tih vrijednosti jednaka je 1. Za klasu koja poprimi najveću vrijednost od ostalih klasa kažemo da je predviđena klasa.

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

(10)

[4] Graf funkcije nacrtan koristeći Desmos Graph Calculator (https://www.desmos.com/calculator)
[5] Why is the ReLU function not differentiable at x=0? (https://sebastianraschka.com/faq/docs/relu-derivative.html)

U osnovi algoritam propagacije „unaprijed" se može svesti na slijedeće korake:

- Učitavamo podatke u našu varijablu *X*
- Primjenimo određeni filter u sklopu konvolucijskog nivoa
- Primjenimo *ReLu* funkciju na rezultat
- Definišemo nasumično odabrane vrijednosti težina i bias-a i izvršimo lineranu transformaciju
- Primjenimo *softmax* funkciju na prethodni rezultat, te time dobijemo finalnu izlaznu vrijednost algoritma propagacije „unaprijed"

Prethodne vrijednosti težinskih tačaka i bias-a predstavljaju parametre naše neuronske mreže. Nad ovim vrijednostima se vrše dodatne izmjene kako bi se postigla što bolja predikcija krajnjih rezultata. Sve se ovo dešava u sklopu algoritma propagacije „unazad".

Za izmjenu ovih vrijednosti obično se koristi tehnika *gradijentnog pada/spusta.* U našem radu korišten je Adam optimizacijski algoritam. Algoritam koristimo iterativno kako bi pri svakoj iteraciji izvršili izmjenu vrijednosti naših težinskih tačaka i bias-a. Naziv Adam nije akronim, a predstavlja izvedenicu od procjene adaptivnog momenta (adaptive moment estimation)[6].

Za razliku od klasičnog stohastičnog gradijentnog pada, gdje je vrijednost parametra stope učenja (learning rate) fiksan za sve NN parametere za sve iteracije, Adam kombinira dvije prednosti iz algoritama AdaGrad (Adaptive Gradient Algorithm) i RMSProp (Root Mean Square Propagation) kako bi izračunao vlastitu stopu učenja. AdaGrad algoritam zadržava stopu učenja po parametru gdje se pokazao dobro kod raspršenih podataka, [7] dok RMSProp algoritam isto zadržava stope učenja po parametru koji su prilagođeni na osnovu prosjeka prethodnih veličina gradijenata za težine.[8]

Adam algoritam se izvršava u nekoliko koraka, gdje prvo potrebno podesiti određene inicijalne vrijednosti potrebnih varijabli:

[6] Adam: A method for stochastic optimization – Diederik P. Kingma, Jimmy Lei Ba
[7] Overview of artifical neural network technologies, Tin Krambeger, Bojan Nožica, Ivica Dodig, Davor Cafuta
[8] Gentle Introduction to the Adam Optimization Algorithm for Deep Learning - Jason Brownlee

- Stopa učenja (lr = 0.001)
- Beta 1 (ß1 = 0.9)
- Beta 2 (ß2 = 0.999)

Ove vrijednosti ujedno predstavljaju i preporučene vrijednosti po autorima Adam algoritma.[6] Beta parametri kontrolišu stopu opadanja kod pokretnih prosjeka, i to na način da, ako je vrijednost beta1 parametra jednaka 0.9 to znači da će uzeti vrijednosti zadnjih 10 iteracija kako bi izračunao prosjek, dok će ostale prethodne odbaciti. Beta2 parametar je rezultat RMSProp dijela algoritma koji je korišten u sklopu Adam algoritma.

Nakon svake iteracije i izmjene naših parametara vršimo poziv softmax funkcije koja daje izlaz vrijednosti predikcije za naše četiri klase (bad, good, neutral i broken). Kako bi mogli utvrditi da li su ti rezultati najpouzdaniji moramo ih uporediti sa već predviđenim rezultatima iz našeg dataset-a. Vrijednost razlike između trenutno dobivenih klasa i predviđenih klasa jeste greška koju je potrebno umanjiti, a za njen izračun koristimo neku od *loss* funkcija – funkcija gubitka.

Obzirom da naš model kao output vrši multi-klasifikaciju slika, tako ćemo koristiti i funkciju gubitka koja odgovara našem algoritmu učenja – *Categorical Crossentropy* funkciju.

Categorical Crossentropy funkcija ima slijedeći oblik:

$$CE = -\sum_{i=0}^{N} p_k \log p'_k$$

(11)

U formuli (10) parametar *p* predstavlja vrijednost predviđene klase iz dataset-a, dok p' je vrijednost dobivena iz softmax funkcije.

Ukratko, optimizacijski algoritmi nam pomažu da minimiziramo rezultat funkcije gubitka tako što vrše izmjene i prilagođavanja vrijednosti težinskih tačaka u modelu. Promjene u funkciji gubitka nam govore da li se krećemo u dobrom pravcu ili ne.

5. Rezultati testiranja

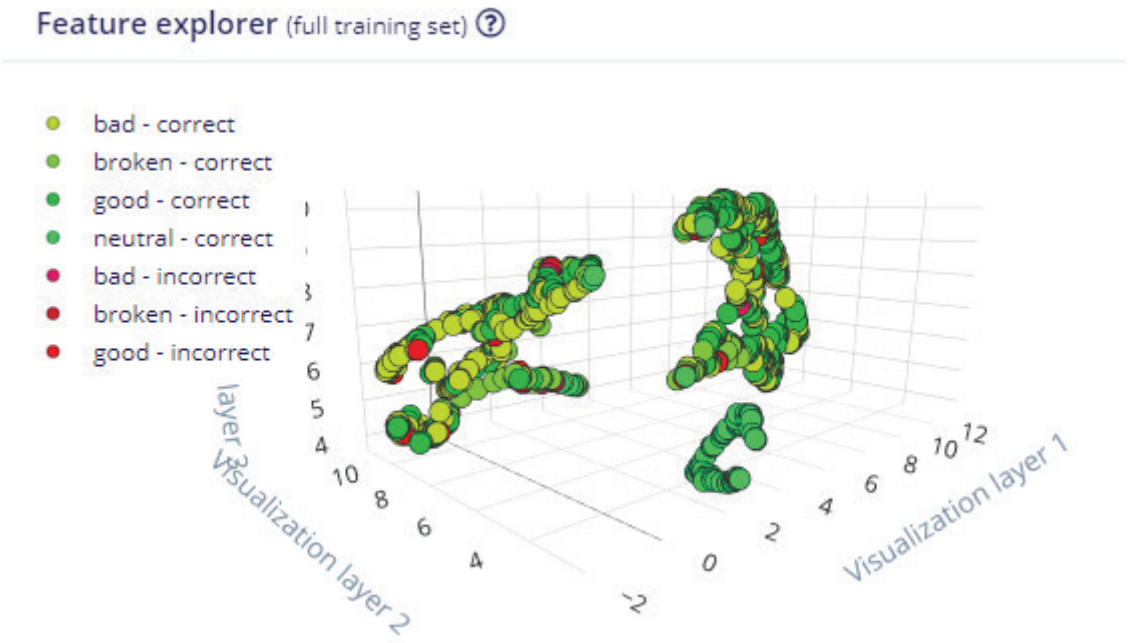
Proces učenja je dao određene rezultate o preciznosti klasifikacije i u našem slučaju tačnost klasifikacije iznosi 92,1%, ali isto tako nam nije pokazao više detalja o grešakama koje su nastale tokom procesa učenja. Upravo zato se u sklopu rezultata prikazuje i matrica konfuzije ili zbrke koja nam daje uvid u ne samo greške koje pravi klasifikator već, što je još važnije, u vrste grešaka koje se prave.

	BAD	BROKEN	GOOD	NEUTRAL
BAD	94.8%	2.1%	3.1%	0%
BROKEN	15.4%	61.5%	19.2%	3.8%
GOOD	3.2%	2.4%	94.4%	0%
NEUTRAL	0%	0%	0%	100%
F1 SCORE	0.93	0.68	0.94	0.98

Slika 30 - Rezultati treniranja modela

Kod rezultata treniranja, iako izgleda prihvatljivo, možemo primjetiti da klasa broken odstupa u određenom procentu od ostalih klasa. Procenat da će mreža prepoznati predmet kao „broken" iznosi 61,5%, a da će isti taj predmet prepoznati kao „bad" ili „good" iznosi 15,4%, i 19,2% respektivno. Kako bi povećali performanse modela prvi korak jeste analiza podataka u dataset-u, gdje se može uvidjeti zašto je rezultat predikcije predmeta kao „broken" u navedenom procentu. Prilikom akvizicije podataka primjetan je dosta manji broj kolekcije slika za klasu „broken" u odnosu na druge dvije klase: bad i good. Za treniranje i testiranje mreže odvojeno je ukupno 682 slika za bad, 135 slika za broken, 706 za good, te 164 slika za neutral klasu. Evidentno je da se algoritam „mučio" sa treniranjem klase „broken" upravo iz razloga nedovoljne količine podataka. Iako, klasa neutral isto tako ima manju količinu input podataka, za razliku od drugih klasa, slike navedene klase ne sadrže nikakve obrise promatranih predmeta, tako

da mreža vrlo lahko radi prepoznavanje. Pomoću *feature explorer-a* možemo steći bolji uvid u razdvojenost klasa.



Slika 31 - Feature explorer - Edge Impulse

Povećanje performansi modela u osnovi uključuje kostantnu izmjenu (tweaking) mreže, kako od povećanja ulazni podataka, tako i izmjena na samim slojevima mreže. U slučaju da mreža nije dovoljno naučena, onda je potrebno dodati broj iteracija, a ako i tada nije dovoljno naučena onda vjerovatno je razlog nedovljna količina ulazni podataka. Isto tako, ako je mreža naučena jako dobro, ali ne daje dovoljno dobre rezultate predikcije, onda je mreža i suviše dobro naučila karakteristike naših podataka da se loše ponaša prilikom predikcije novih podataka. Ovo se još zove i *overfitting*, te u tom slučaju se obićno pribjegava dopuni dataset-a sa novim podacima, ili se mijenja stopa učenja.

6. Nadogradnja projekta

Kao prvo uređaj bi trebalo uraditi u metalnoj izvedbi kako bi se kamera, kao njen osnovni sastavni dio, mogla stabilizirati još bolje, te time postići dosta veća predikcija anomalija na predmetima.

Pored toga uređaj bi se mogao nadograditi da može izvršavati klasifikaciju oblih/okruglih predmeta, jer trenutno je moguće vršiti detekciju samo jedne strane predmeta koji se nalaze na remenu, što ujedno predstavlja glavnu manu ovog projekta. Kako bi uređaj mogao raditi klasifikaciju navedenih predmeta, de-facto je potrebna izgradnja sasvim novog uređaja što vrlo vjerovatno iziskuje i timski rad.

Kao nadogradnju projekta možemo uvrstiti i mrežnu komunikaciju sa određenim web servisima/aplikacijama kako bi mogli upravljati podacima koje bi pribavljali tokom rada uređaja, i ne samo to, već i uzimanje uzoraka za naknadno učenje našeg modela, kako u u sklopu nadziranog učenja (supervised), tako i učenja bez nadzora (unsupervised).

7. Zaključak

Umjetna inteligencija je sastavni dio četvrte industrijske revolucije. Temelji bilo koje tehnologije sazdani su na dobro uređenim procedurama unutar kompanija, te sprezi između njenih sektora koje čine sistem kao cijelinu. Da bi se jedna kompanija utrkivala u vremenu četvrte industrijske revolucije potrebno je da pokrene značajne promjene u svojim proizvodnim procesima, a poboljša saradnju na nivou poslovnih procesa i lanca snabdijevanja. Pametne fabrike, umjetna inteligencija, virtuelna i proširena realnost, Internet of Things su tehnologije koje su u procesu evolucije i razvijaju se iz dana u dan. Za upravljanje ovim tehnologijama je potrebno znanje, a kako je dobro poznato, znanje je resurs koji se svojom potrošnjom samo više povećava. Upravo ovo znanje posjeduju ljudi. Kod ljudi postoji strah, ali ne od neznanja, već od nepoznatog. Sa dobro isplaniranom obukom, uposlenici će shvatiti da sve ove tehnologije pridonose u njihovom radu, pa tako i senzori na mašinama će prijevremeno javiti određenom aplikativnom sistemu da je moguć kvar mašine. Ili, ako je došlo do kvara, ukazati tačno na dio mašine koji je u kvaru.

Četvrta industrijska revolucija i popratne tehnologije donose komunikaciju u realnom vremenu, kako između ljudi i mašina, tako i između samih mašina, a sve u cilju smanjenja vremena proizvodnih procesa i bolje informisanosti o statusu svih procesa u kompaniji koji su potpomognuti pomenutim tehnologijama.

8. Literatura

1. **Quick reference for the OpenMV Cam** – (https://docs.openmv.io/openmvcam/quickref.html), Damien P. George, Paul Sokolovsky, OpenMV LLC, and contributors, last updated on 27 Jun 2021; accessed on 15. October, 2021.

2. **MicroPython documentation** – (http://docs.micropython.org/en/latest), Damien P. George, Paul Sokolovsky, and contributors, last updated on 15 Nov 2021; accessed on 15. November, 2021.

3. **Edge Impulse OpenMV Cam H7 Plus Documentation** – (https://docs.edgeimpulse.com/docs/openmv-cam-h7-plus)

4. **OpenMV forums** – (https://forums.openmv.io/)

5. **A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way –** (https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53), Sumit Saha, Dec 15, 2018; accessed on Dec 15, 2021.

6. **Why is the ReLU function not differentiable at x=0?** – (https://sebastianraschka.com/faq/docs/relu-derivative.html), Sebastian Raschka, accessed on Feb 10, 2022.

7. **Adam: A method for stochastic optimization** – (https://arxiv.org/pdf/1412.6980.pdf), Diederik P. Kingma, Jimmy Lei Ba, published as a conference paper at ICLR 2015, accessed on Feb 12, 2022.

8. **Gentle Introduction to the Adam Optimization Algorithm for Deep Learning** – (https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/), Jason Brownlee, July 3, 2017, accessed on Feb 11, 2022.

9. **Overview of artifical neural network technologies** – (https://polytechnicanddesign.tvz.hr/index.php/ojs/article/view/245/224), Tin Krambeger, Bojan Nožica, Ivica Dodig, Davor Cafuta, POLYTECHNIC & DESIGN, Vol. 7, No. 1, 2019, accessed on Feb 12, 2022.

10. **Object detection** – (https://docs.edgeimpulse.com/docs/object-detection#configuring-the-transfer-learning-model), Edge Impulse Guides, accessed on Feb 20, 2022.