EMERGING METHOD FOR EARLY DETECTION OF FOREST FIRE

INTRODUCTION:

1.1PROJECT OVERVIEW:

Wildfires are unplanned and unwanted fires, including lightning-caused fires, unauthorized human-caused fires, and escaped prescribed fire projects. Nationwide data compiled by the National Interagency Fire Center (NIFC) indicate that the number of annual wildfires is variable but has decreased slightly over the last 30 years and that the number of acres impacted annually, while also variable, generally has increased (see Figure 1). Since 2000, an annual average of 70,685 wildfires burned an annual average of 7.1million acres. This figure is more than double the average annual acreage burned in the 1990s (3.3 million acres), although a greater number of fires occurred annually in the 1990s (78,600 on average). Utilizing RF and designed antennae, our prototype solution would aim to reduce these statistics by assessing wildfire locations, direction, and severity in order to brief and notify the necessary fire response teams to prevent greater ecological, infrastructural, and societal damages. The Feather LoRa transceiver/receiver system would adequately be able to perform this task using its accessible frequencies in combination with the various detection sensors and modules for the Arduino microcontroller that we have shown to be quite effective at doing so. We have presented our findings and testing results in this presentation.

1.2 PURPOSE:

These solutions mainly aim to **mitigate the damage caused by the** fires, using methods for their early detection

2.LITERATURE SURVEY:

2.1 EXSISTING PROBLEM:

Forest and urban fires have been and still are serious problem for many countries in the world. Currently, there are many different solutions to fight forest fires. These solutions mainly aim to mitigate the damage caused by the fires, using methods for their early detection. In this paper, we discuss a new approach for fire detection and control, in which modern technologies are used. In particular, we propose a platform that uses Unmanned Aerial Vehicles (UAVs), which constantly patrol over potentially threatened by fire areas. The UAVs also utilize the benefits from Artificial Intelligence (AI) and are equipped with on-board processing capabilities. This allows them to use computer vision methods for recognition and detection of smoke or fire, based on the still images or the video input from the drone cameras. Several different scenarios for the possible use of the UAVs for forest fire detection are presented and analyse in the paper, including a solution with the use of a combination between a fixed and rotary-wing drones. Keywords – early forest fire detection platform, drones, UAVs, artificial intelligence, computer vision

2.2 REFERENCES:

[1]Official webpage of the European Forest Fire Information System at: <http://effis.jrc.ec.europa.eu/>

[2] Jesús San-Miguel-Ayanz, Tracy Durrant, Roberto Boca, Giorgio Libertà, Alfredo Branco, Daniele de Rigo, Davide Ferrari, Pieralberto Maianti, Tomàs Artés Vivancos, Hugo Costa, Fabio Lana, Peter Löffler, Daniel Nuijten, Anders Christofer Ahlgren, Thaïs Leray; Forest Fires in Europe, Middle East and North Africa 2017. EUR 29318 EN, ISBN 978-92-79-92831-4, doi: 10.2760/663443

[3] Chen, Thou-Ho, et al. "The smoke detection for early fire-alarming system base on video processing." Intelligent Information Hiding and Multimedia Signal Processing, 2006. IIH-MSP'06. International Conference on. IEEE, 2006.

[4] Noda, S., and K. Ueda. "Fire detection in tunnels using an image processing method." Vehicle Navigation and Information Systems Conference, 1994. Proceedings., 1994. IEEE, 1994.

[5] Chen, Thou-Ho, Cheng-Liang Kao, and Sju-Mo Chang. "An intelligent real-time fire-detection method based on video processing." Security Technology, 2003. Proceedings. IEEE 37th Annual 2003 International Carnahan Conference on. IEEE, 2003.

[6] Wang, Da-Jinn, Yen-Hui Yin, and Tsong-Yi Chen. "Smoke Detection for Early Fire-Alarming System Based on Video Processing." Journal of Digital Information Management 6.2 (2008).

[7] Ivanov, Alexander, and Penka Georgieva. "КЛАСИФИКАЦИЯ С КОНВОЛЮЦИОННИ НЕВРОННИ МРЕЖИ." КОМПЮТЪРНИ НАУКИ И КОМУНИКАЦИИ 7.1 (2018): 46-52.

[8] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[9] An open source machine learning framework for everyone, Tensorflow Community - <https://www.tensorflow.org/>

[10] LabelImg, graphical image annotation tool - <https://github.com/tzutalin/labelImg>

[11] Official webpage of ALTi unmanned aerial systems available at <https://www.altiuas.com/>

[12] Official webpage of NextVision NightHawk2 camera available at <https://www.nextvision-sys.com/nighthawk-2>

[13] Official webpage of DJI M210 RTK v2 camera drone available at <https://www.dji.com/bg/matrice-200-series-v2>

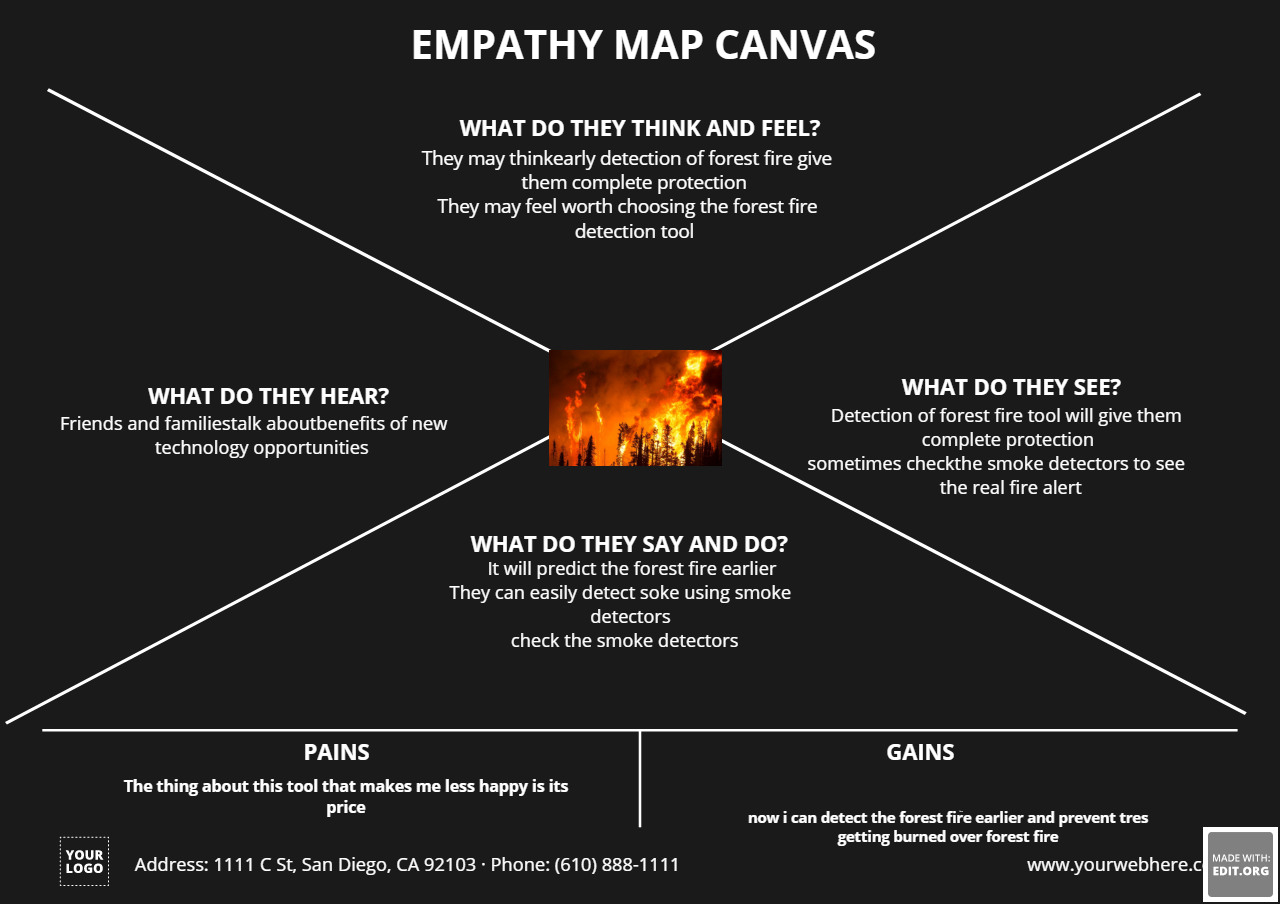
[14] Official webpage of DJI Manifold available at https://www.dji.com/bg/manifold

2.3 PROBLEM STATEMENT DEFINITION:

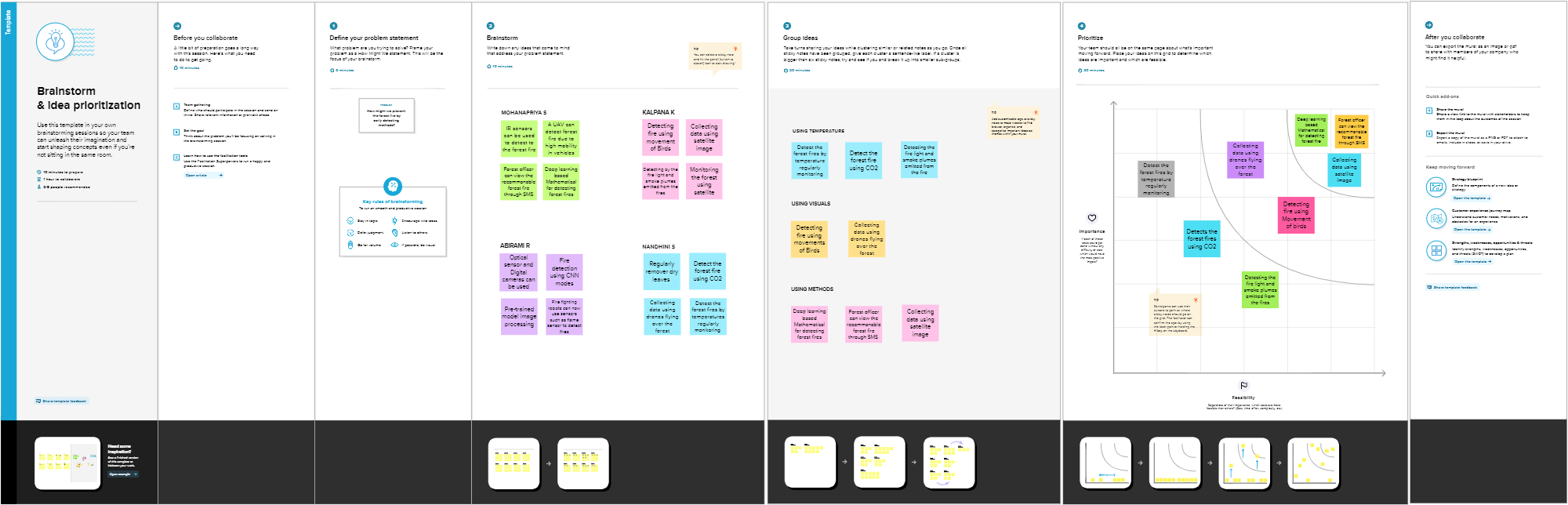
* The user interacts with a web camera to read the video.
* Once the input image from the video frame is sent to the model, if the fire is detected it is showcased on the console, and alerting sound will be generated and an alert message will be sent to the Authorities.

3.IDEATION AND PROPOSED SOLUTION:

3.1 EMPATHY MAP CANVAS:



3.2 IDEATION AND BRAINSTORMING:

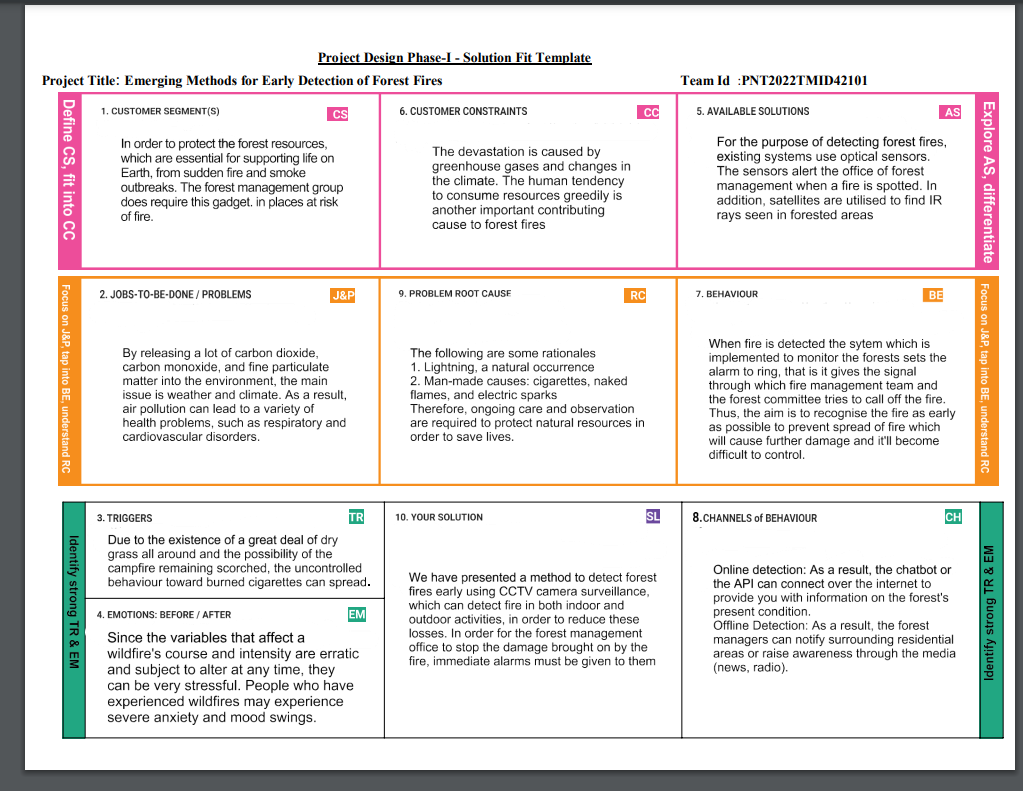


3.3 PROPOSED SOLUTION:

Fire detection systems for outdoor environment could be implemented by using specialized cameras, which are able to capture multispectral images. The biggest challenge that arises in these setups is where to place the camera(s) in order to have the best viewon the observed territory. Since these systems have their limitations, since they provide stationary point of view, we have decided to investigate a new approach. The platform that is proposed in this paper will use unmanned aerial vehicles, which are going to patrol above the desired territory and will constantly observe for fire-related events. The drones will be equipped with specialized optical and thermal cameras and will be able to capture video or still images. In addition, the drones will also have constant bidirectional connection to the base station and they will be able to provide a feedback about their observations. We have architected our proposed model based on LTEM technology which will be mounted on the belt of forest animals. When animals are moving in the forest and come in the range of stationary nodes where ZigBee wireless sensors are deployed then the modules of LTE-M will collect data from ZigBee wireless sensors and will send it to the cloud where the sensor data will be analysed accordingly [11, 12, 13]. Figure 1 shows the WSN system where sensor nodes collect the data from the environment and transmit the data to PROPOSED SOLUTION the sink node of the cluster. Then sink node will collect the information through all the sensors and form a database. After some threshold time LTE-M module will be transferred the data to the cloud server.

3.4 PROBLEM SOLUTION FIT:

Fire detection systems for outdoor environment could be implemented by using specialized cameras, which are able to capture multispectral images. The biggest challenge that arises in these setups is where to place the camera(s) in order to have the best viewon the observed territory. Since these systems have their limitations, since they provide stationary point of view, we have decided to investigate a new approach. The platform that is proposed in this paper will use unmanned aerial vehicles, which are going to patrol above the desired territory and will constantly observe for fire-related events. The drones will be equipped with specialized optical and thermal cameras and will be able to capture video or still images. In addition, the drones will also have constant bidirectional connection to the base station and they will be able to provide a feedback about their observations. We have architected our proposed model based on LTEM technology which will be mounted on the belt of forest animals. When animals are moving in the forest and come in the range of stationary nodes where ZigBee wireless sensors are deployed then the modules of LTE-M will collect data from ZigBee wireless sensors and will send it to the cloud where the sensor data will be analysed accordingly [11, 12, 13]. Figure 1 shows the WSN system where sensor nodes collect the data from the environment and transmit the data to PROPOSED SOLUTION the sink node of the cluster. Then sink node will collect the information through all the sensors and form a database. After some threshold time LTE-M module will be transferred the data to the cloud server.



4.FUNCTIONAL REQUIREMENTS:

4.1 FUNCTIONAL REQUIREMENTS:

● The system shall take training sets of fire images and recognize whether there is

a fire or the beginning of a fire (smoke) or if there is no fire

● The system shall send a notification to the admin when it recognizes a fire in the

image given

● The system shall take real inputs of satellite images and determine whether the

image contains a fire or not

● The system shall be able to take images with a variety of sizes and convert it to

one fixed image to be used throughout the application

● The system shall run as a service on either a Windows or Linux operating

system.

● In the event that the computer on which the system is running shuts down, the

system service should start automatically when the computer restarts

4.2 NON-FUNCTIONAL REQUIREMENTS:

Performance Requirements ● The system shall be able to analyze the image given has a fire or not in less than five minutes

● The system shall have an accuracy rate of at least 90% when attempting to detect if a given image has a fire or not

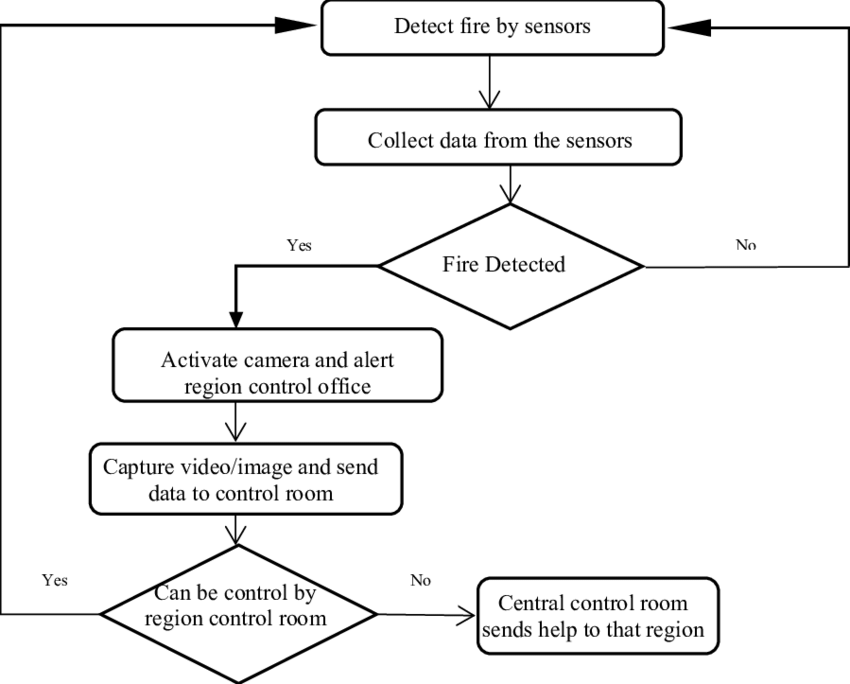
Software System Attributes

● Portability: The system shall be compatible with many API’s

● Testability: Putting in more training data into the system will improve the accuracy of its ability to detect a fire

5.PROJECT DESIGN:

5.1 DATA FLOW DIAGRAM:



5.2 SOLUTION AND TECHNICAL ARCHITECTURE:

Nowadays, two different types of sensor networks are available for fire detection, camera surveillance and wireless sensor network. The development of sensors, digital camera, image processing, and industrial computers resulted in the development of a system for optical, automated early recognition and warning of forest fires.

Different types of detection sensors can be used in terrestrial systems [[6](https://journals.sagepub.com/doi/full/10.1155/2014/597368#B6-2014-597368)]:

**(i)**

video-camera, sensitive to visible spectrum of smoke recognisable during the day and a fire recognisable at night,

**(ii)**

infrared (IR), thermal imaging cameras based on the detection of heat flow of the fire,

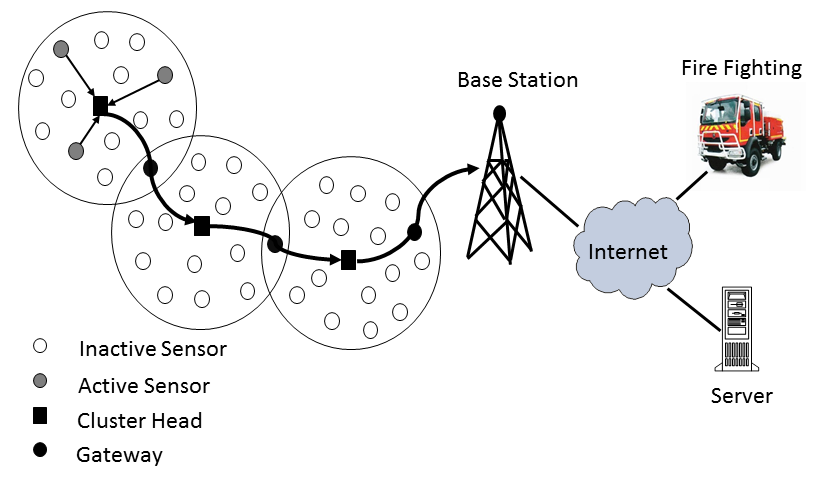
**(iii)**

IR spectrometers to identify the spectral characteristics of smoke,

**(iv)**

light detection and ranging systems—LIDAR (detection of light and range) that measure laser rays reflected from the smoke particles.

The variant optical systems working according to different algorithms designed by the producers, all have the same general concept in smoke and fire glow detection. Simply, the camera produces images every while. The image consists of a number of pixels, where the processing unit tracks the motion in images and checks how many pixels contain smoke or fire glow and then the processing unit sends the results for another algorithm to decide whether or not to produce an alarm for the operator.



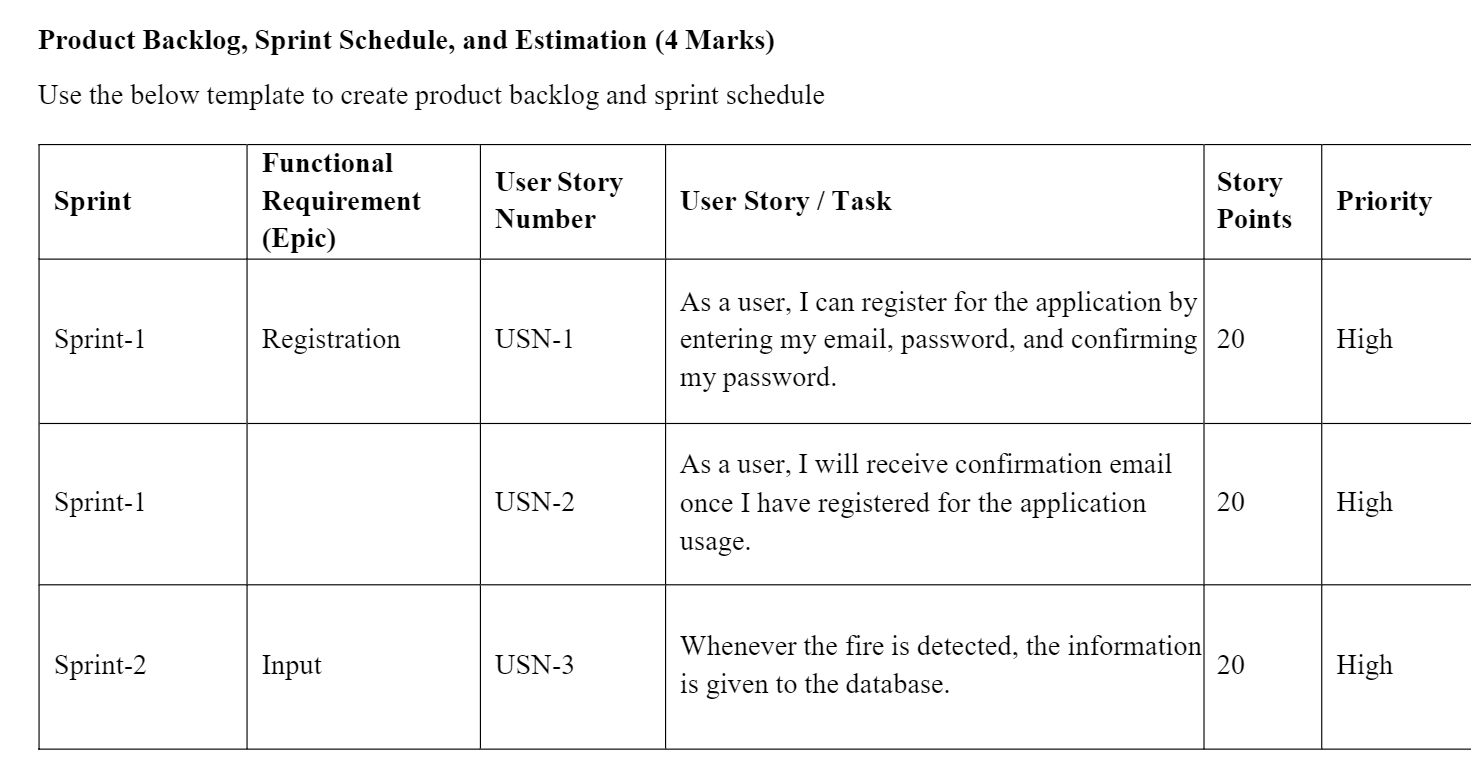
5.3 USER STORIES:

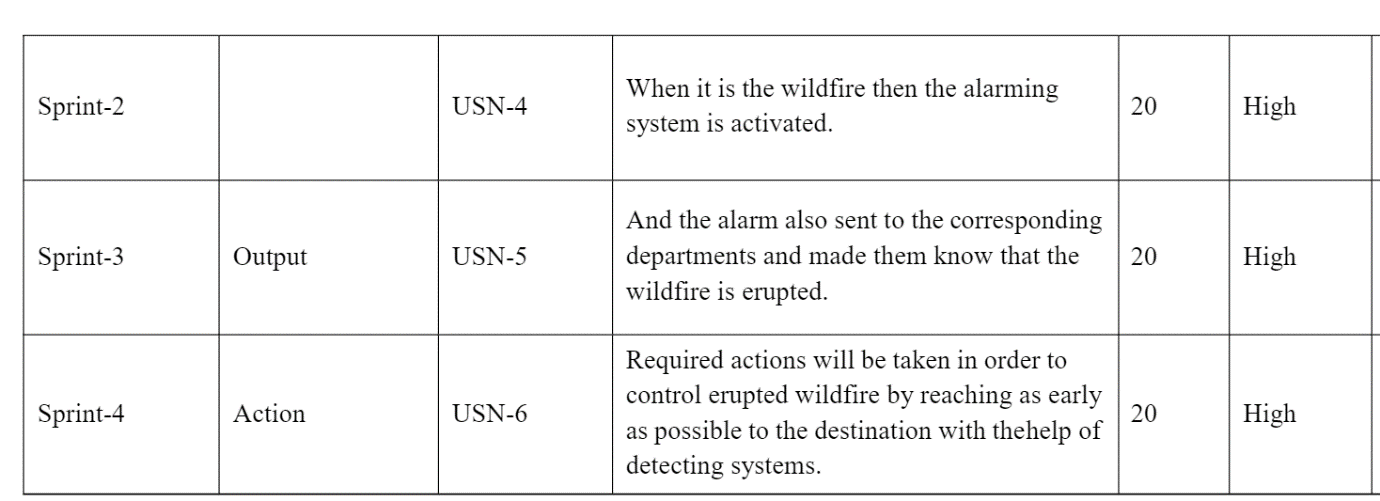
smoke detectors were a great buy. One big caveat: they are so sensitive that one placed on a ceiling more than 5ft from a bathroom door alerted after someone showered (with the bathroom fan on and window open 4") and opened the bathroom door after about 5 min of drying off and applying body lotion. That is kinda nuts, right? It will only get worse in the winter, with no window open! Another one alerted when a visitor vaped a 3-4 times with medium clouds in the living room, but this detector was at least 20 ft down a hallway from the room we were in. Patio door was open, living room ceiling fan was running. The detector in the living room (cathedral ceiling 10 ft away from vaper) didn't alert, though? Sorry, but that's just crazy sensitive.

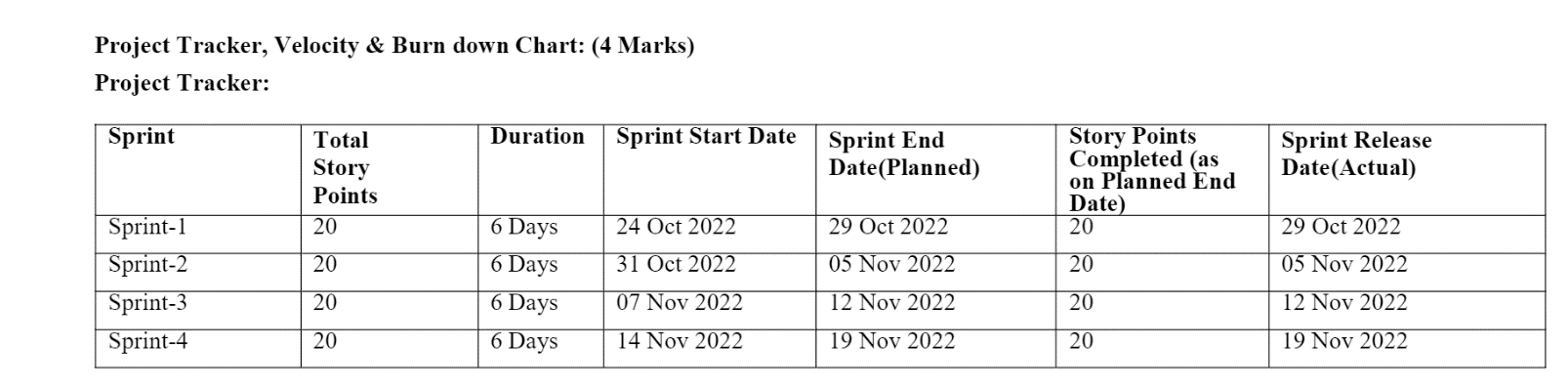
When purchasing a property replaced all smoke/fire detectors and carbon dioxide detectors.  
  
On each detector wrote a year x10 years. The last note says Replace. These are meant to last ten years.  
  
Like, car tires should be replaced six years after the manufacture date.

6.PROJECT PLANNING AND SCHEDULING:

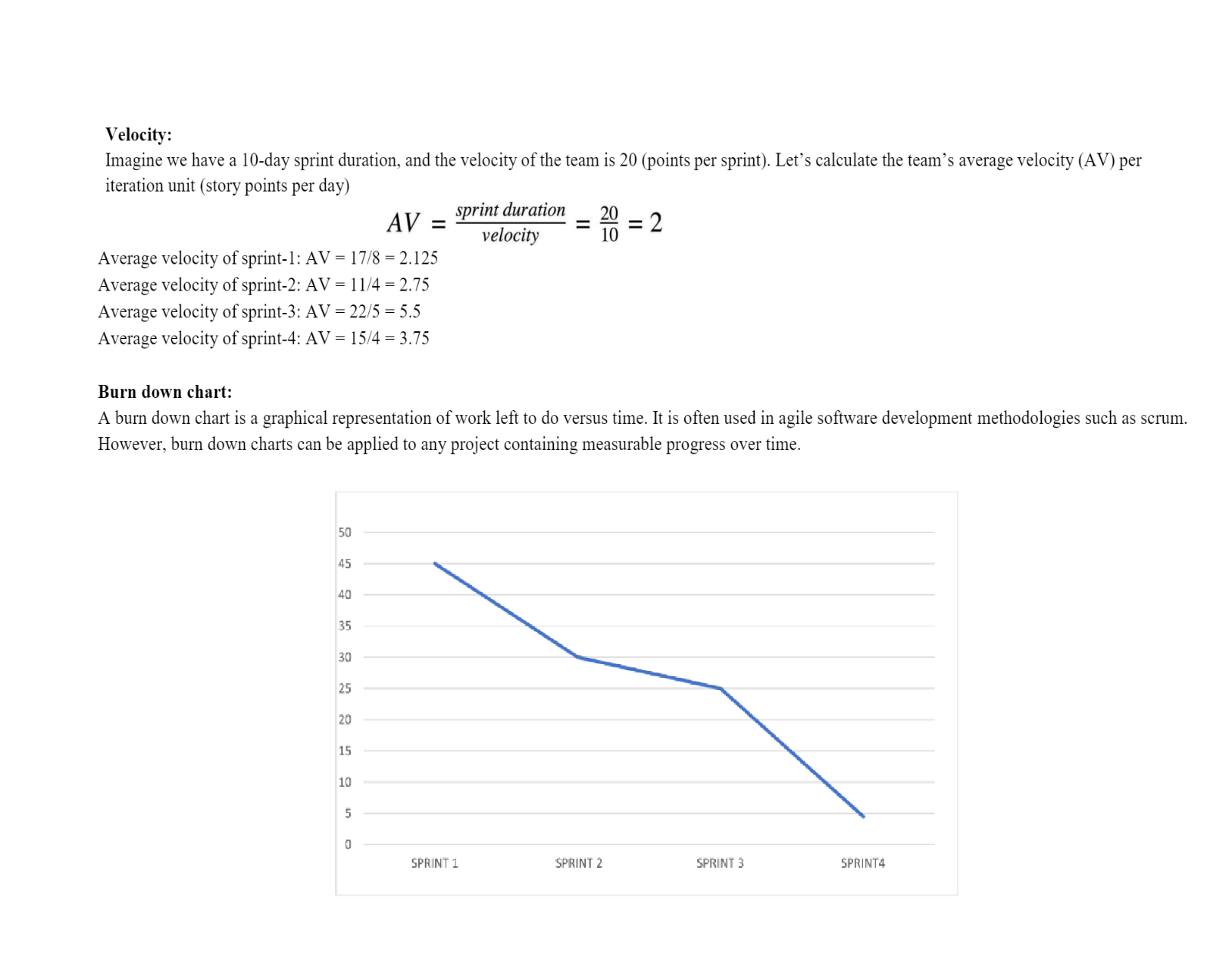
6.1 SPRINT PLANNING AND ESTIMATION:







6.2 SPIRINT DELIVERY PLAN AND 6.3 REPORTS FROM JIRA



7.CODING AND SOLUTIONING:

7.1 FEATURE1, 7.2 FEATURE2, 7.3 DATABASE SCHEMA

Smoke alarms detect particles in the air. They most commonly do this using two types of detection technologies.

First, there are ionization detectors. These use a small bit of safely shielded radioactive material that electrically charges, or ionizes, the air molecules between two metal plates. This produces a small electric current flowing from one plate to the other in the air. When particles enter the chamber, they attract the ions and carry them away, reducing the current. When the number of particles entering the chamber is enough to reduce that current below a certain amount, the device will register those particles as smoke and the alarm will sound. (And about that radioactive material? Most of its radiation is blocked inside the device, and even then, the radiation levels in the device are much lower than the natural background radiation to which we are exposed every day.)

The other type of commonly used detection technology is called photoelectric. This technology works by detecting light that is reflected off particles from a light beam inside the sensing chamber. When no particles are present in the sensing chamber, the light from the beam does not strike the light detector, indicating all clear. When there are particles present and the amount of light registered by the light detector reaches a certain threshold level, the alarm sounds.







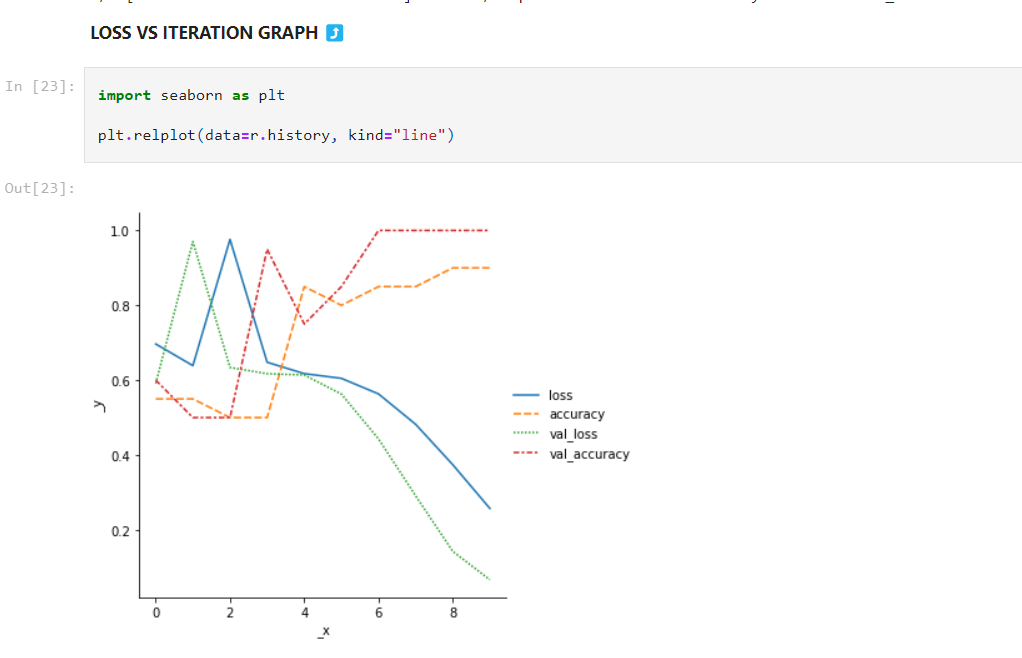


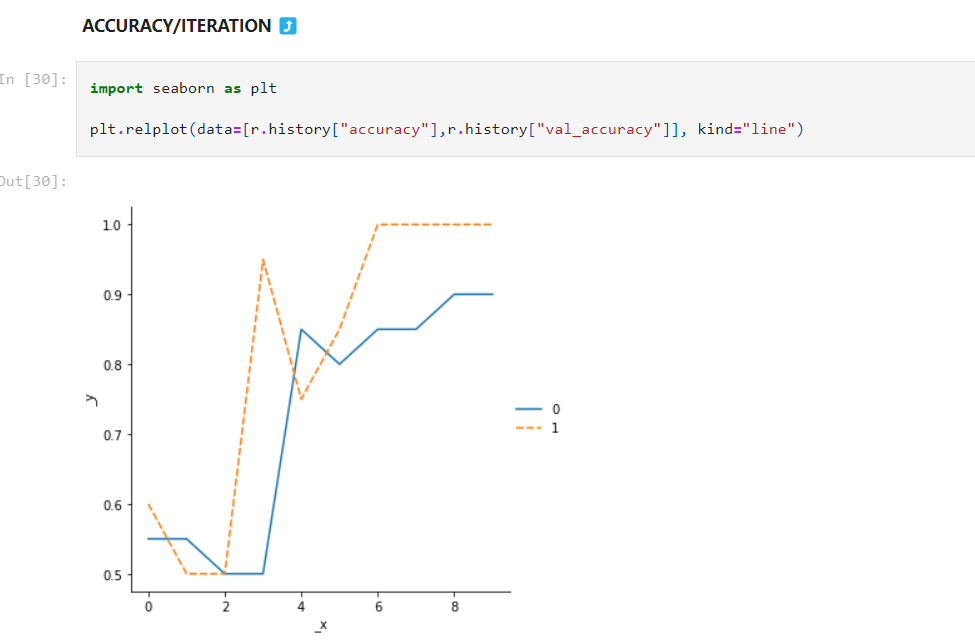


8.TESTING

8.1 TEST CASES

8.2 USER ACCEPTANCE AND TESTING









9.RESULT:

9.1 PERFORMANCE METRICS:

Conclusions & Future Work To limit the damage caused by forest fires and to control the start of fires and its spread, we have presented in this study a method of early detection of forest fires. This method is based on three steps: Estimate the general risk level of the forest, assess and predict in several places the existence or not of fires, and alert the necessary first responders to quell the spread of the fires. The originality of this work lies in the use of a wireless sensor and RF network distributed over the entire forest area and the deep learning methods to predict in real-time a possible origination and predicted path of the forest fire

10.ADVANTAGES AND DISADVANTAGES:

Forest fires may be ignited due to climatic changes or human activities .Moreover, the forest fire is observed only when it has already spread through a large area, making the process of stopping these fires very difficult . Thus, these fires result in devastating losses and large damages to both the environment and atmosphere.

The very huge area of forest is destroyed by fire every year. Monitoring of the potential risk is sand **an early detection of fire can significantly shorten the reaction time and also reduce the potential damage as well as the cost of firefighting**.

11.CONCLUSION:

To limit the damage caused by forest fires and to control the start of fires and its spread, we have presented in this study a method of early detection of forest fires. This method is based on three steps: Estimate the general risk level of the forest, assess and predict in several places the existence or not of fires, and alert the necessary first responders to quell the spread of the fires. The originality of this work lies in the use of a wireless sensor and RF network distributed over the entire forest area and the deep learning methods to predict in real-time a possible origination and predicted path of the forest fire.

12.FUTURE SCOPE:

The current system will be implemented on a large scale with multiple sensor nodes to power and augment the data set in order to improve the accuracy and collaboration of data between multiple nodes. We plan in future work to use wind direction sensors to properly estimate and locate the start of the fire, and to collaborate with SpaceX’s Star Link Program to monitor rural forest areas as well.

13.APPENDIX:

Source code:

DATABASE COLLECTION:

{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"provenance": [],

"collapsed\_sections": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "markdown",

"source": [

"# \*\*DATA COLLECTION\*\*"

],

"metadata": {

"id": "ANAn53JJCh0B"

}

},

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {

"id": "BXCT4OadBFNK"

},

"outputs": [],

"source": [

"import tensorflow as tf\n",

"import numpy as np\n",

"from tensorflow import keras\n",

"import os\n",

"import cv2\n",

"from tensorflow.keras.preprocessing.image import ImageDataGenerator\n",

"from tensorflow.keras.preprocessing import image\n",

"import matplotlib.pyplot as plt"

]

},

{

"cell\_type": "code",

"source": [

"train = ImageDataGenerator(rescale=1/255)\n",

"test = ImageDataGenerator(rescale=1/255)\n",

"\n",

"train\_dataset = train.flow\_from\_directory(\"/content/drive/MyDrive/IBM/Dataset/train\_set\",\n",

" target\_size=(150,150),\n",

" batch\_size = 32,\n",

" class\_mode = 'binary')\n",

"\n",

"test\_dataset = test.flow\_from\_directory(\"/content/drive/MyDrive/IBM/Dataset/test\_set\",\n",

" target\_size=(150,150),\n",

" batch\_size =32,\n",

" class\_mode = 'binary')"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "T7KK5YZOBKXq",

"outputId": "4a4e56a9-8cbc-4b46-e64e-6eee6fff20b4"

},

"execution\_count": 5,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Found 20 images belonging to 2 classes.\n",

"Found 20 images belonging to 2 classes.\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"test\_dataset.class\_indices"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "uEP-FODSCUdj",

"outputId": "15658cf1-ebf3-4afd-e9f2-82647c91a0cb"

},

"execution\_count": 6,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"{'Fire': 0, 'NoFire': 1}"

]

},

"metadata": {},

"execution\_count": 6

}

]

}

]

}

Both kinds of detectors can detect either slow-burning “smoldering” fires or fast-burning “flaming” fires, but each technology has its particular strengths. Ionization-based alarms tend to detect small black soot particles from flaming fires more quickly because they are produced in greater numbers and take away more current from between the plates. Photoelectric detectors tend to be more sensitive to particles that are larger in size and white or light-colored, and thus more reflective, like those emitted by smoldering fires.

[The NFPA advises](https://www.nfpa.org/Public-Education/Staying-safe/Safety-equipment/Smoke-alarms/Ionization-vs-photoelectric) that people have both ionization and photoelectric units in their homes. And dual-sensor alarms that combine both technologies are also available.

As important as smoke alarms are for protecting your family and your property, many times they can be a nuisance. Smoke alarms near kitchens can detect the particles coming off your food as it cooks, even if you don’t burn it. Sometimes something as simple as turning on a toaster can set them off.

So as with many safety measures, smoke detectors have a trade-off. They can be made sensitive enough to detect almost any smoke. But if they did, they would detect the smoke you don’t want them to detect (such as from cooked food) and even other things such as dust. Less sensitive detectors would have fewer nuisance alarms, but in an actual fire, they may not go off in time to save lives or property. Or they may not give off a signal at all.

Researchers are developing new tests and standards to make smoke alarms better at detecting the kinds of smoke we want them to detect and not the kinds we don’t, so we’re never tempted to disable the alarms and put ourselves in danger. As a result, the next generation of smoke detectors promises to cut down on the number of nuisance alarms while also signaling real fires more quickly. And with fire, time is everything when it comes to saving lives and property.

{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"provenance": [],

"collapsed\_sections": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "markdown",

"source": [

"# \*\*DATA COLLECTION\*\*"

],

"metadata": {

"id": "ANAn53JJCh0B"

}

},

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {

"id": "BXCT4OadBFNK"

},

"outputs": [],

"source": [

"import tensorflow as tf\n",

"import numpy as np\n",

"from tensorflow import keras\n",

"import os\n",

"import cv2\n",

"from tensorflow.keras.preprocessing.image import ImageDataGenerator\n",

"from tensorflow.keras.preprocessing import image\n",

"import matplotlib.pyplot as plt"

]

},

{

"cell\_type": "code",

"source": [

"train = ImageDataGenerator(rescale=1/255)\n",

"test = ImageDataGenerator(rescale=1/255)\n",

"\n",

"train\_dataset = train.flow\_from\_directory(\"/content/drive/MyDrive/IBM/Dataset/train\_set\",\n",

" target\_size=(150,150),\n",

" batch\_size = 32,\n",

" class\_mode = 'binary')\n",

"\n",

"test\_dataset = test.flow\_from\_directory(\"/content/drive/MyDrive/IBM/Dataset/test\_set\",\n",

" target\_size=(150,150),\n",

" batch\_size =32,\n",

" class\_mode = 'binary')"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "T7KK5YZOBKXq",

"outputId": "4a4e56a9-8cbc-4b46-e64e-6eee6fff20b4"

},

"execution\_count": 5,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Found 20 images belonging to 2 classes.\n",

"Found 20 images belonging to 2 classes.\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"test\_dataset.class\_indices"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "uEP-FODSCUdj",

"outputId": "15658cf1-ebf3-4afd-e9f2-82647c91a0cb"

},

"execution\_count": 6,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"{'Fire': 0, 'NoFire': 1}"

]

},

"metadata": {},

"execution\_count": 6

}

]

},

{

"cell\_type": "markdown",

"source": [

"# \*\*MODEL BUILDING AND FITTING\*\*"

],

"metadata": {

"id": "tGRG7iuECvqH"

}

},

{

"cell\_type": "code",

"source": [

"model = keras.Sequential()\n",

"model.add(keras.layers.Conv2D(32,(3,3),activation='relu',input\_shape=(150,150,3)))\n",

"model.add(keras.layers.MaxPool2D(2,2))\n",

"model.add(keras.layers.Conv2D(64,(3,3),activation='relu'))\n",

"model.add(keras.layers.MaxPool2D(2,2))\n",

"model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))\n",

"model.add(keras.layers.MaxPool2D(2,2))\n",

"model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))\n",

"model.add(keras.layers.MaxPool2D(2,2))\n",

"model.add(keras.layers.Flatten())\n",

"model.add(keras.layers.Dense(512,activation='relu'))\n",

"model.add(keras.layers.Dense(1,activation='sigmoid'))"

],

"metadata": {

"id": "bYe4luWkCza1"

},

"execution\_count": 7,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\*\*COMPILE THE MODEL ✈\*\*"

],

"metadata": {

"id": "S7fEAbEnC7u0"

}

},

{

"cell\_type": "code",

"source": [

"model.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])"

],

"metadata": {

"id": "xKNjrrgNC1T-"

},

"execution\_count": 8,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\*\*FIT THE MODEL ⚡\*\*"

],

"metadata": {

"id": "gARkf8tRDGfn"

}

},

{

"cell\_type": "code",

"source": [

"r = model.fit(train\_dataset,\n",

" epochs = 10,\n",

" validation\_data = test\_dataset)"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "qXOnG9aWDL0B",

"outputId": "b5f89445-2806-4d89-c16c-54f44f6c3134"

},

"execution\_count": 9,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Epoch 1/10\n",

"1/1 [==============================] - 12s 12s/step - loss: 0.6967 - accuracy: 0.5500 - val\_loss: 0.5912 - val\_accuracy: 0.6000\n",

"Epoch 2/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.6388 - accuracy: 0.5500 - val\_loss: 0.9704 - val\_accuracy: 0.5000\n",

"Epoch 3/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.9761 - accuracy: 0.5000 - val\_loss: 0.6340 - val\_accuracy: 0.5000\n",

"Epoch 4/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.6478 - accuracy: 0.5000 - val\_loss: 0.6171 - val\_accuracy: 0.9500\n",

"Epoch 5/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.6174 - accuracy: 0.8500 - val\_loss: 0.6139 - val\_accuracy: 0.7500\n",

"Epoch 6/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.6048 - accuracy: 0.8000 - val\_loss: 0.5628 - val\_accuracy: 0.8500\n",

"Epoch 7/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.5632 - accuracy: 0.8500 - val\_loss: 0.4429 - val\_accuracy: 1.0000\n",

"Epoch 8/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.4820 - accuracy: 0.8500 - val\_loss: 0.2908 - val\_accuracy: 1.0000\n",

"Epoch 9/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.3741 - accuracy: 0.9000 - val\_loss: 0.1425 - val\_accuracy: 1.0000\n",

"Epoch 10/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.2571 - accuracy: 0.9000 - val\_loss: 0.0666 - val\_accuracy: 1.0000\n"

]

}

]

}

]

}

{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"provenance": [],

"collapsed\_sections": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "markdown",

"source": [

"# \*\*DATA COLLECTION\*\*"

],

"metadata": {

"id": "ANAn53JJCh0B"

}

},

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {

"id": "BXCT4OadBFNK"

},

"outputs": [],

"source": [

"import tensorflow as tf\n",

"import numpy as np\n",

"from tensorflow import keras\n",

"import os\n",

"import cv2\n",

"from tensorflow.keras.preprocessing.image import ImageDataGenerator\n",

"from tensorflow.keras.preprocessing import image\n",

"import matplotlib.pyplot as plt"

]

},

{

"cell\_type": "code",

"source": [

"train = ImageDataGenerator(rescale=1/255)\n",

"test = ImageDataGenerator(rescale=1/255)\n",

"\n",

"train\_dataset = train.flow\_from\_directory(\"/content/drive/MyDrive/IBM/Dataset/train\_set\",\n",

" target\_size=(150,150),\n",

" batch\_size = 32,\n",

" class\_mode = 'binary')\n",

"\n",

"test\_dataset = test.flow\_from\_directory(\"/content/drive/MyDrive/IBM/Dataset/test\_set\",\n",

" target\_size=(150,150),\n",

" batch\_size =32,\n",

" class\_mode = 'binary')"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "T7KK5YZOBKXq",

"outputId": "4a4e56a9-8cbc-4b46-e64e-6eee6fff20b4"

},

"execution\_count": 5,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Found 20 images belonging to 2 classes.\n",

"Found 20 images belonging to 2 classes.\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"test\_dataset.class\_indices"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "uEP-FODSCUdj",

"outputId": "15658cf1-ebf3-4afd-e9f2-82647c91a0cb"

},

"execution\_count": 6,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"{'Fire': 0, 'NoFire': 1}"

]

},

"metadata": {},

"execution\_count": 6

}

]

},

{

"cell\_type": "markdown",

"source": [

"# \*\*MODEL BUILDING AND FITTING\*\*"

],

"metadata": {

"id": "tGRG7iuECvqH"

}

},

{

"cell\_type": "code",

"source": [

"model = keras.Sequential()\n",

"model.add(keras.layers.Conv2D(32,(3,3),activation='relu',input\_shape=(150,150,3)))\n",

"model.add(keras.layers.MaxPool2D(2,2))\n",

"model.add(keras.layers.Conv2D(64,(3,3),activation='relu'))\n",

"model.add(keras.layers.MaxPool2D(2,2))\n",

"model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))\n",

"model.add(keras.layers.MaxPool2D(2,2))\n",

"model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))\n",

"model.add(keras.layers.MaxPool2D(2,2))\n",

"model.add(keras.layers.Flatten())\n",

"model.add(keras.layers.Dense(512,activation='relu'))\n",

"model.add(keras.layers.Dense(1,activation='sigmoid'))"

],

"metadata": {

"id": "bYe4luWkCza1"

},

"execution\_count": 7,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\*\*COMPILE THE MODEL ✈\*\*"

],

"metadata": {

"id": "S7fEAbEnC7u0"

}

},

{

"cell\_type": "code",

"source": [

"model.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])"

],

"metadata": {

"id": "xKNjrrgNC1T-"

},

"execution\_count": 8,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\*\*FIT THE MODEL ⚡\*\*"

],

"metadata": {

"id": "gARkf8tRDGfn"

}

},

{

"cell\_type": "code",

"source": [

"r = model.fit(train\_dataset,\n",

" epochs = 10,\n",

" validation\_data = test\_dataset)"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "qXOnG9aWDL0B",

"outputId": "b5f89445-2806-4d89-c16c-54f44f6c3134"

},

"execution\_count": 9,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Epoch 1/10\n",

"1/1 [==============================] - 12s 12s/step - loss: 0.6967 - accuracy: 0.5500 - val\_loss: 0.5912 - val\_accuracy: 0.6000\n",

"Epoch 2/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.6388 - accuracy: 0.5500 - val\_loss: 0.9704 - val\_accuracy: 0.5000\n",

"Epoch 3/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.9761 - accuracy: 0.5000 - val\_loss: 0.6340 - val\_accuracy: 0.5000\n",

"Epoch 4/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.6478 - accuracy: 0.5000 - val\_loss: 0.6171 - val\_accuracy: 0.9500\n",

"Epoch 5/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.6174 - accuracy: 0.8500 - val\_loss: 0.6139 - val\_accuracy: 0.7500\n",

"Epoch 6/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.6048 - accuracy: 0.8000 - val\_loss: 0.5628 - val\_accuracy: 0.8500\n",

"Epoch 7/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.5632 - accuracy: 0.8500 - val\_loss: 0.4429 - val\_accuracy: 1.0000\n",

"Epoch 8/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.4820 - accuracy: 0.8500 - val\_loss: 0.2908 - val\_accuracy: 1.0000\n",

"Epoch 9/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.3741 - accuracy: 0.9000 - val\_loss: 0.1425 - val\_accuracy: 1.0000\n",

"Epoch 10/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.2571 - accuracy: 0.9000 - val\_loss: 0.0666 - val\_accuracy: 1.0000\n"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"\*\*LOSS VS ITERATION GRAPH ⤴\*\*"

],

"metadata": {

"id": "kD2KcKiLEo\_k"

}

},

{

"cell\_type": "code",

"source": [

"import seaborn as plt\n",

"\n",

"plt.relplot(data=r.history, kind=\"line\")"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 387

},

"id": "YHzoKL\_REupV",

"outputId": "01a01820-4e6d-4631-dfdb-4a45be634231"

},

"execution\_count": 23,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"<seaborn.axisgrid.FacetGrid at 0x7fbea1851bd0>"

]

},

"metadata": {},

"execution\_count": 23

},

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 461x360 with 1 Axes>"

],

"image/png": "\n"

},

"metadata": {

"needs\_background": "light"

}

}

]

},

{

"cell\_type": "markdown",

"source": [

"\*\*ACCURACY/ITERATION ⤴\*\*"

],

"metadata": {

"id": "OBVYXZMFGoN8"

}

},

{

"cell\_type": "code",

"source": [

"import seaborn as plt\n",

"\n",

"plt.relplot(data=[r.history[\"accuracy\"],r.history[\"val\_accuracy\"]], kind=\"line\")"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 387

},

"outputId": "169663a4-561c-4c4c-c053-386c6a36e0c2",

"id": "lKZQaOTrF0zQ"

},

"execution\_count": 30,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"<seaborn.axisgrid.FacetGrid at 0x7fbea1454c90>"

]

},

"metadata": {},

"execution\_count": 30

},

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 402.375x360 with 1 Axes>"

],

"image/png": "\n"

},

"metadata": {

"needs\_background": "light"

}

}

]

}

]

}

{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"provenance": [],

"collapsed\_sections": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "markdown",

"source": [

"# \*\*DATA COLLECTION\*\*"

],

"metadata": {

"id": "ANAn53JJCh0B"

}

},

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {

"id": "BXCT4OadBFNK"

},

"outputs": [],

"source": [

"import tensorflow as tf\n",

"import numpy as np\n",

"from tensorflow import keras\n",

"import os\n",

"import cv2\n",

"from tensorflow.keras.preprocessing.image import ImageDataGenerator\n",

"from tensorflow.keras.preprocessing import image\n",

"import matplotlib.pyplot as plt"

]

},

{

"cell\_type": "code",

"source": [

"train = ImageDataGenerator(rescale=1/255)\n",

"test = ImageDataGenerator(rescale=1/255)\n",

"\n",

"train\_dataset = train.flow\_from\_directory(\"/content/drive/MyDrive/IBM/Dataset/train\_set\",\n",

" target\_size=(150,150),\n",

" batch\_size = 32,\n",

" class\_mode = 'binary')\n",

"\n",

"test\_dataset = test.flow\_from\_directory(\"/content/drive/MyDrive/IBM/Dataset/test\_set\",\n",

" target\_size=(150,150),\n",

" batch\_size =32,\n",

" class\_mode = 'binary')"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "T7KK5YZOBKXq",

"outputId": "4a4e56a9-8cbc-4b46-e64e-6eee6fff20b4"

},

"execution\_count": 5,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Found 20 images belonging to 2 classes.\n",

"Found 20 images belonging to 2 classes.\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"test\_dataset.class\_indices"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "uEP-FODSCUdj",

"outputId": "15658cf1-ebf3-4afd-e9f2-82647c91a0cb"

},

"execution\_count": 6,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"{'Fire': 0, 'NoFire': 1}"

]

},

"metadata": {},

"execution\_count": 6

}

]

},

{

"cell\_type": "markdown",

"source": [

"# \*\*MODEL BUILDING AND FITTING\*\*"

],

"metadata": {

"id": "tGRG7iuECvqH"

}

},

{

"cell\_type": "code",

"source": [

"model = keras.Sequential()\n",

"model.add(keras.layers.Conv2D(32,(3,3),activation='relu',input\_shape=(150,150,3)))\n",

"model.add(keras.layers.MaxPool2D(2,2))\n",

"model.add(keras.layers.Conv2D(64,(3,3),activation='relu'))\n",

"model.add(keras.layers.MaxPool2D(2,2))\n",

"model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))\n",

"model.add(keras.layers.MaxPool2D(2,2))\n",

"model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))\n",

"model.add(keras.layers.MaxPool2D(2,2))\n",

"model.add(keras.layers.Flatten())\n",

"model.add(keras.layers.Dense(512,activation='relu'))\n",

"model.add(keras.layers.Dense(1,activation='sigmoid'))"

],

"metadata": {

"id": "bYe4luWkCza1"

},

"execution\_count": 7,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\*\*COMPILE THE MODEL ✈\*\*"

],

"metadata": {

"id": "S7fEAbEnC7u0"

}

},

{

"cell\_type": "code",

"source": [

"model.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])"

],

"metadata": {

"id": "xKNjrrgNC1T-"

},

"execution\_count": 8,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\*\*FIT THE MODEL ⚡\*\*"

],

"metadata": {

"id": "gARkf8tRDGfn"

}

},

{

"cell\_type": "code",

"source": [

"r = model.fit(train\_dataset,\n",

" epochs = 10,\n",

" validation\_data = test\_dataset)"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "qXOnG9aWDL0B",

"outputId": "b5f89445-2806-4d89-c16c-54f44f6c3134"

},

"execution\_count": 9,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Epoch 1/10\n",

"1/1 [==============================] - 12s 12s/step - loss: 0.6967 - accuracy: 0.5500 - val\_loss: 0.5912 - val\_accuracy: 0.6000\n",

"Epoch 2/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.6388 - accuracy: 0.5500 - val\_loss: 0.9704 - val\_accuracy: 0.5000\n",

"Epoch 3/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.9761 - accuracy: 0.5000 - val\_loss: 0.6340 - val\_accuracy: 0.5000\n",

"Epoch 4/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.6478 - accuracy: 0.5000 - val\_loss: 0.6171 - val\_accuracy: 0.9500\n",

"Epoch 5/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.6174 - accuracy: 0.8500 - val\_loss: 0.6139 - val\_accuracy: 0.7500\n",

"Epoch 6/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.6048 - accuracy: 0.8000 - val\_loss: 0.5628 - val\_accuracy: 0.8500\n",

"Epoch 7/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.5632 - accuracy: 0.8500 - val\_loss: 0.4429 - val\_accuracy: 1.0000\n",

"Epoch 8/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.4820 - accuracy: 0.8500 - val\_loss: 0.2908 - val\_accuracy: 1.0000\n",

"Epoch 9/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.3741 - accuracy: 0.9000 - val\_loss: 0.1425 - val\_accuracy: 1.0000\n",

"Epoch 10/10\n",

"1/1 [==============================] - 2s 2s/step - loss: 0.2571 - accuracy: 0.9000 - val\_loss: 0.0666 - val\_accuracy: 1.0000\n"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"\*\*LOSS VS ITERATION GRAPH ⤴\*\*"

],

"metadata": {

"id": "kD2KcKiLEo\_k"

}

},

{

"cell\_type": "code",

"source": [

"import seaborn as plt\n",

"\n",

"plt.relplot(data=r.history, kind=\"line\")"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 387

},

"id": "YHzoKL\_REupV",

"outputId": "01a01820-4e6d-4631-dfdb-4a45be634231"

},

"execution\_count": 23,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"<seaborn.axisgrid.FacetGrid at 0x7fbea1851bd0>"

]

},

"metadata": {},

"execution\_count": 23

},

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 461x360 with 1 Axes>"

],

"image/png": "\n"

},

"metadata": {

"needs\_background": "light"

}

}

]

},

{

"cell\_type": "markdown",

"source": [

"\*\*ACCURACY/ITERATION ⤴\*\*"

],

"metadata": {

"id": "OBVYXZMFGoN8"

}

},

{

"cell\_type": "code",

"source": [

"import seaborn as plt\n",

"\n",

"plt.relplot(data=[r.history[\"accuracy\"],r.history[\"val\_accuracy\"]], kind=\"line\")"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 387

},

"outputId": "169663a4-561c-4c4c-c053-386c6a36e0c2",

"id": "lKZQaOTrF0zQ"

},

"execution\_count": 30,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"<seaborn.axisgrid.FacetGrid at 0x7fbea1454c90>"

]

},

"metadata": {},

"execution\_count": 30

},

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 402.375x360 with 1 Axes>"

],

"image/png": "\n"

},

"metadata": {

"needs\_background": "light"

}

}

]

}

]

}

[**IBM-Project-34563-1660238466**](https://github.com/IBM-EPBL/IBM-Project-34563-1660238466)

**Emerging Methods for Early Detection of Forest Fires**

**TEAM ID:PNT2022TMID10426**

**Team members:**

**1.S.RISHIKA(TEAMLEADER)**

**2.VS.VYSHAKH**

**3.M.UDHAYA**

**4.RISHI**