

Development of a convolutional neural network (CNN) algorithm for fire disaster management in commercial complexes in Nigeria

Item Type	Journal article
Authors	Olatunji, Ezekiel;Oladokun, Victor Oluwasina
Citation	Olatunji, E. and Oladokun, V.O. (2025) Development of a convolutional neural network (CNN) algorithm for fire disaster management in commercial complexes in Nigeria. International Journal of Building Pathology and Adaptation, Vol. ahead-of-print No. ahead-of-print. https://doi.org/10.1108/IJBPA-10-2024-0222
DOI	10.1108/IJBPA-10-2024-0222
Publisher	Emerald
Journal	International Journal of Building Pathology and Adaptation
Download date	2026-04-15 06:28:00
License	https://creativecommons.org/licenses/by-nc/4.0/
Link to Item	https://wlv.openrepository.com/handle/2436/626170





**DEVELOPMENT OF A CONVOLUTIONAL NEURAL NETWORK
(CNN) ALGORITHM FOR FIRE DISASTER MANAGEMENT IN
COMMERCIAL COMPLEXES IN NIGERIA**

Journal:	<i>International Journal of Building Pathology and Adaptation</i>
Manuscript ID	IJBPA-10-2024-0222.R1
Manuscript Type:	Original Article
Keywords:	Convolutional Neural Network, CNN algorithm, fire detection, fire disaster management, commercial complexes

SCHOLARONE™
Manuscripts

Table 1: Search term, image sample and description of various dataset categories

S/N	Date	Search Term	Image Sample	General Description	Image Source	Sample	Action
1	24-Apr-2023	commercial complex fires		Result showed fire burning in commercial complexes from an external view	A&J Restoration, 2023 https://ajrestores.com/wp-content/uploads/2018/08/shutterstock_412794292.jpg	Property	30 images retained.
2	24-Apr-2023	commercial complex internal fires		Result revealed a mixture of images displayed in the 'commercial complex fires' search term and fire images burning within product shelves as seen.	Shutterstock, 2023 https://www.shutterstock.com/image-photo/warehouse-fire-inside-523768042		15 images retained.

1
2
3 24- fires within
4 Apr- supermarket
5 2023 product
6 section



7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

Result revealed fire burning within product stands and those burning outside some popular malls like NEXT Supermarket in Nigeria

Thecitizen.com, 2023
https://d3ebobe8115pwo.cloudfront.net/wp-content/uploads/2022/08/082522_early-photo-of-Walmart-fire-in-paper-goods-aisle_E.jpg

10 images retained.

4 24- Walmart on
Apr- fire
2023



This produced better result with images of fire burning within product sections of different supermarkets (Walmart). Result is partly because there are more fire image data available in developed countries.

Global News
<https://globalnews.ca/news/7472295/walmart-store-arson-kitchener-waterloo-opens/>

30 images retained.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

5 24- fires within
Apr- supermarket
2023 product
section in
Nigeria



This produced 21 images https://i0.wp.com/med 15 images
on the internet showing the ia.ghgossip.com/wp- retained.
scarcity of fire images content/uploads/2021/
within commercial 07/27071953/superma
complexes in Nigeria. The rket-
particular image displayed fire.jpg?fit=1200%2C
was of a lady holding an 800&ssl=1
ignited lighter

6 19- Fire and
Jun- Smoke
2023



Result showed many https://encrypted- 150 images
images with smoke and fire tbn3.gstatic.com/imag retained.
combined in different es?q=tbn:ANd9GcSM
scenarios. Some had X5haKvoNEqM-
humans, others do not. XdSXBN-
SR6JmN1b5S4-
2wbOetQ16a889LqAf

1
2
3 7 22 - inner shelves
4 May commercial
5 - mall images
6
7
8 2023



From the sample shown, https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcQp5_D8qKnv_hMQxcsd5pGDqKPLgEhN_SwM3Warb5SM6F_crlCmW this search term generated 35 images with humans and shelves in a supermarket setting. One noticeable thing was that fire was absent in all images making it fit for the 'no fire' image folder.

35 images retained.

19 8 13- internal
20 June shelves
21 - commercial
22 2023 complex



The images generated by https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTMxRuLzLOI8KAAMhJbaJchK9_aBEMogVC4tGNA_7mC7HXGNEnF this search term only displayed the supermarket setting without humans in it, making it fit for situations where CCTV monitoring is on-going even though business is closed for the day.

40 images retained.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

Table 2: Detailed augmentation(s) for each search term category

S/N	Search Term	Initial number of images	Data Augmentation Applied	Total Images	Appendix
1	commercial complex fires	30 images retained.	None	30 images	
2	commercial complex internal fires	15 images retained.	All 15 images were flipped and added to original set	30 images	Appendix 1
3	fires within supermarket product section	10 images retained.	Original 10 images were zoomed and later rotated, giving additional 20 images	30 images	Appendix 2
4	Walmart on fire	30 images retained.	None	30 images	
5	fires within supermarket product section in Nigeria	15 images retained.	All 15 images were stretched and added to original dataset	30 images	Appendix 3
6	Fire and Smoke	150 images retained.	None	150 images	

7	inner shelves commercial mall images	35 images retained.	All 35 images were affined and added to initial dataset.	70 images	Appendix 4
8	internal shelves commercial complex	40 images retained.	35 images were cropped and 5 blurred giving additional 40 images to initial dataset.	80 images	Appendix 5
Total				450 images	

Table 3. Classification Report

	Precision	Recall	F1-score	Support
0	0.85	0.65	0.73	34
1	0.37	0.64	0.47	11
Accuracy			0.64	45
Macro Avg	0.61	0.64	0.60	45

Weighted Avg	0.73	0.64	0.67	45
--------------	------	------	------	----

Table 4. Confusion Matrix

	Actual Positive	Actual Negative
Predicted Positive	22	12
Predicted Negative	4	7

1
2
3
4
5 Epoch 1/100
6 13/13 - 102s - loss: 0.7445 - accuracy: 0.6130 - val_loss: 0.9304 - val_accuracy: 0.5000 - 102s/epoch - 8s/step
7 Epoch 2/100
8 13/13 - 107s - loss: 0.5875 - accuracy: 0.7013 - val_loss: 0.6940 - val_accuracy: 0.5000 - 107s/epoch - 8s/step
9 Epoch 3/100
10 13/13 - 97s - loss: 0.5595 - accuracy: 0.6961 - val_loss: 0.6003 - val_accuracy: 0.5000 - 97s/epoch - 7s/step
11 Epoch 4/100
12 13/13 - 105s - loss: 0.5761 - accuracy: 0.6987 - val_loss: 0.5347 - val_accuracy: 0.7500 - 105s/epoch - 8s/step
13 Epoch 5/100
14 13/13 - 101s - loss: 0.5312 - accuracy: 0.6883 - val_loss: 0.5666 - val_accuracy: 0.5000 - 101s/epoch - 8s/step
15 Epoch 6/100
16 13/13 - 97s - loss: 0.5157 - accuracy: 0.7065 - val_loss: 0.4656 - val_accuracy: 0.7500 - 97s/epoch - 7s/step
17 Epoch 7/100
18 13/13 - 108s - loss: 0.4841 - accuracy: 0.7221 - val_loss: 0.5674 - val_accuracy: 0.5500 - 108s/epoch - 8s/step
19 Epoch 8/100
20 13/13 - 95s - loss: 0.5046 - accuracy: 0.7481 - val_loss: 0.3693 - val_accuracy: 0.8500 - 95s/epoch - 7s/step
21 Epoch 9/100
22 13/13 - 96s - loss: 0.4793 - accuracy: 0.7662 - val_loss: 0.3682 - val_accuracy: 0.8000 - 96s/epoch - 7s/step
23 Epoch 10/100
24 13/13 - 104s - loss: 0.4470 - accuracy: 0.7818 - val_loss: 0.3537 - val_accuracy: 0.7500 - 104s/epoch - 8s/step
25 Epoch 11/100
26 13/13 - 99s - loss: 0.3933 - accuracy: 0.8026 - val_loss: 0.3943 - val_accuracy: 0.8000 - 99s/epoch - 8s/step
27 Epoch 12/100
28 13/13 - 96s - loss: 0.4047 - accuracy: 0.8130 - val_loss: 0.3480 - val_accuracy: 0.8000 - 96s/epoch - 7s/step
29 Epoch 13/100
30 13/13 - 98s - loss: 0.4139 - accuracy: 0.7948 - val_loss: 0.4053 - val_accuracy: 0.7500 - 98s/epoch - 8s/step
31 Epoch 14/100
32 13/13 - 97s - loss: 0.4116 - accuracy: 0.7948 - val_loss: 0.1806 - val_accuracy: 0.9000 - 97s/epoch - 7s/step
33 Epoch 15/100
34 13/13 - 103s - loss: 0.3814 - accuracy: 0.8156 - val_loss: 0.3598 - val_accuracy: 0.8000 - 103s/epoch - 8s/step
35 Epoch 16/100
36 13/13 - 97s - loss: 0.3359 - accuracy: 0.8312 - val_loss: 0.3364 - val_accuracy: 0.7500 - 97s/epoch - 7s/step

Figure 1. Training Results for each Epoch

Source: Google Colaboratory Notebook

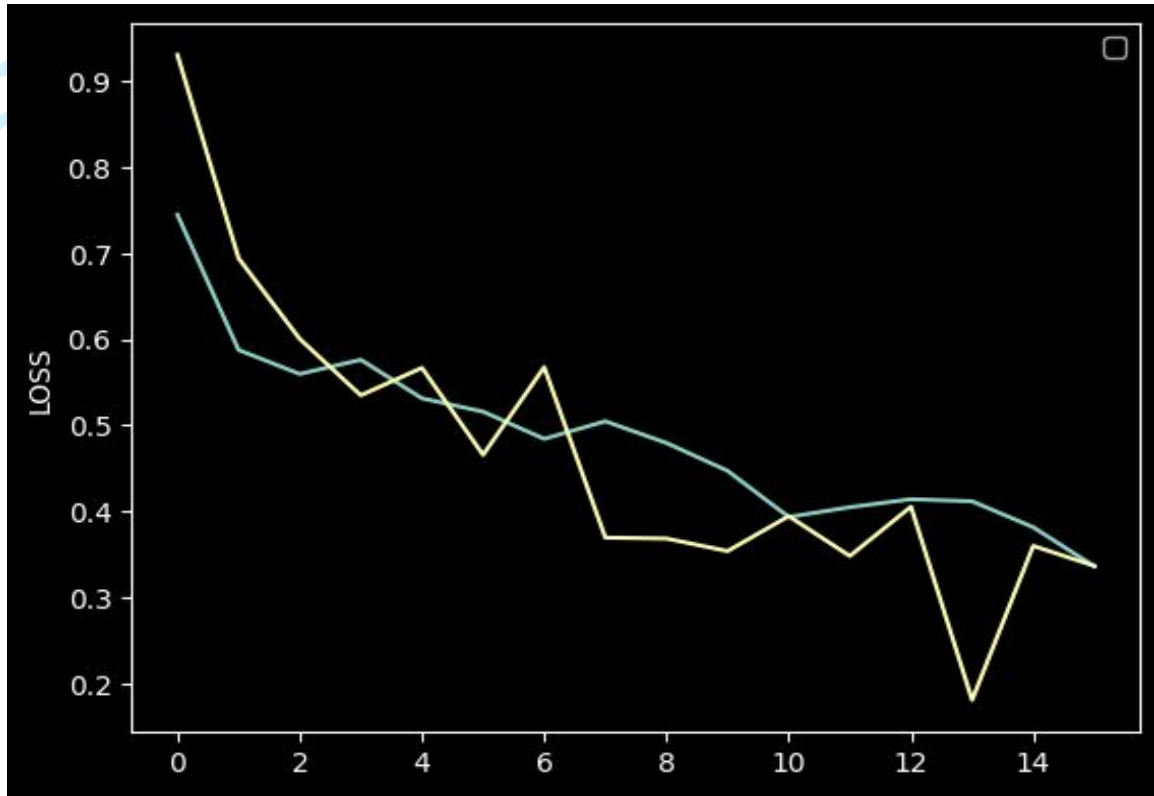


Figure 2 Training Loss (Green) vs. Validation Loss (Yellow)

Source: Google Colaboratory Notebook

```

Model_Results_Three = Model_Three.evaluate(Test_IMG_Set)
print("LOSS: " + "%.4f" % Model_Results_Three[0])
print("ACCURACY: " + "%.2f" % Model_Results_Three[1])

```

```

2/2 [=====] - 4s 571ms/step - loss: 0.5452 - accuracy: 0.7333
LOSS: 0.5452
ACCURACY: 0.73

```

Figure 3. Model Evaluation Metrics and Results

Source: Google Colaboratory Notebook

PREDICTION FOR DIFFERENT SOURCE

- FIRE

```
[ ] image_path = Path("/content/drive/My Drive/Colab Notebooks/Fire Datasets/fire2.jpg")
img = load_img(image_path,target_size=(256,256))
x = img_to_array(img)
x = np.expand_dims(x,axis=0)
```

```
▶ Diff_Pred = Model_Three.predict(x)
Diff_Pred = Diff_Pred.argmax(axis=-1)
print(Diff_Pred)
```

```
↳ 1/1 [=====] - 0s 74ms/step
[0]
```

Figure 4. Correct Prediction on unseen data

DEVELOPMENT OF A CONVOLUTIONAL NEURAL NETWORK (CNN) ALGORITHM FOR FIRE DISASTER MANAGEMENT IN COMMERCIAL COMPLEXES IN NIGERIA

ABSTRACT

Purpose: This study aims to develop a Convolutional Neural Network (CNN) algorithm for fire disaster management in commercial complexes in Nigeria, addressing the limitations of traditional fire detection systems.

Methodology: Secondary data was utilized due to the absence of observational data, with images collected from Google Images. The dataset comprised two classes: fire and non-fire images. Google Colaboratory was employed for algorithm development, taking advantage of its cloud-based environment and integration with deep learning frameworks like TensorFlow. Dataset augmentation and preprocessing were performed to mitigate overfitting and further improve model generalization. The CNN architecture consisted of multiple convolutional and fully connected layers, designed to maximize feature extraction from input images.

Findings: The CNN model demonstrated a good performance, achieving an accuracy of 73% on a separate test dataset. Metrics like precision, F1-score and recall indicated satisfactory performance in classifying fire images. The model exhibited potential in accurately detecting fire instances in commercial complex environments.

Originality/Value: This research contributes to the advancement of the fire disaster management narrative in the Nigerian local context by proposing a computer vision-based approach using RCNNs for fire disaster management in commercial complexes in Nigeria. The dataset augmentation technique used showcases the rigour and reflects the inadequacy of such datasets in Nigeria. Future implementation of this algorithm will involve significant investment in the creation of an image or video database for the algorithm to work on real life scenarios. Its implementation could improve safety measures in commercial complexes, mitigating economic losses and safeguarding lives and properties.

Keywords: Convolutional Neural Network, CNN, fire disaster management, fire detection, commercial complexes

INTRODUCTION

In recent years, humanistic disasters have become a major concern in the world. Unlike natural event that occur with little or no human intervention, humanistic disasters are initiated by people and have threatened the health, well-being, and safety of people. Fire is one of the humanistic disasters that is a major cause of financial losses in an economy (Mwedzi et al., 2019). Various places like schools, offices, residential areas, companies, and commercial complexes have faced this unfortunate incident at a point in time. **Fire incidents often disproportionately affect children and older adults who depend on support (Kumar et al., 2023).** In Nigeria, fire catastrophes in markets and commercial complexes have become a threat to lives and destroyed properties worth billions of Naira (Oladokun & Emmanuel, 2014; Oladokun & Isohola, 2010).

Fire incidents in commercials complexes have become a major discouragement to potential investors despite the significant role these complexes play in the economy (Murray & Watson,

2019). Investments worth billions of Naira are wiped out within a day and recovery from such incidents is sometimes impossible due to the poor economy in Nigeria. In November, Lawal & Ajayi (2019) estimated the nation's financial losses to be around 5 trillion naira from 2016; this amount could have salvaged some part of the Nigerian economy. That same year, the media reports between 2019 to 2021 concluded that 31 out of 68 total fire outbreaks were commercial fires (Abeku et al., 2021).

Hence a need to improve the safety of lives and properties especially in commercial complexes, as this will cater for almost 50% of the fire outbreaks. According to EPS Security (2019), over 100,000 business fires occur annually leading to over 2.4 billion dollars losses in commercial properties per year. In addition, 30.4% office fires, 61% restaurant fires and 50% of motel and hotel fires are caused by cooking. 18% of fires are caused by electrical malfunctions. Other causes of fires in commercial complexes are heating equipment and human error. The Next Cash and Carry Supermarket fire in FCT Abuja occurred on a Sunday morning on the 26th of December 2021 around 8am (Chime, 2021), whereas the supermarket opens from 9am to 9pm daily, this means the fire occurred before the start of business. In addition, the Ebeano Supermarket fire in Abuja also occurred on a Saturday evening around 7pm according to the CCTV footage from the findings of Punch Nigeria (Punch Newspapers, 2021). This brings to an important point that most of these fires occur during the close of business or where there is either few staff members left in the facility or during the weekend.

These timelines further emphasize the reason for this research. Since most fires occur during the close of business or when supervision by human workers is low, there should be a way to monitor the environment automatically. With the advancement in technology, fire detection systems have been built to cater for this loophole. These systems play an important role in safeguarding many places against fire. It is worthy of note that a vital and critical aspect of fire detection is to identify a developing fire early and alert the occupants and intervention systems (Automated Systems, Fire Emergency Organizations and other Fire Personnel) early (Gunawaardena et al., 2016). Conventional fire detection systems forecasts and detects fire by using a combination of factors or by-products of combustibles that can be detected by multiple sensors. These by-products of combustibles include temperature, heat, smoke and Carbon-monoxide density (Chen et al., 2003).

However, the signals sent to the sensors are also affected by environmental conditions like dust, weather, humidity, electronic disturbance, poor installation positions, human error and other man-made activity, making fire detection more complex than the regular signal detection, especially because of slow-smouldering fires. Also, sensor signal processing fails in some situations like distinguishing the heat produced from a fire outbreak and that of a gas or electric cooker in a restaurant or kitchen. These situations and more lead to false alarms and miswarnings which makes the whole system inefficient, ineffective, and unreliable (Chen et al., 2003). In line with this, sensor signal processing is also deficient in determining the fire size, location, and growth rate.

1
2
3 **For the reader's convenience, a glossary defining key terms and acronyms used**
4 **throughout this paper is included in Appendix 7.**
5
6
7

8 **Traditional Fire Detection in Commercial Complexes**

9 Traditional fire detection using heat, smoke and CO detectors are widely used in indoor
10 environments and residential building; however, they are ineffective for commercial
11 complexes because they require closer proximity to the fire scenes (Shi et al., 2017; Zhou et
12 al., 2010). Besides, they require the fire to burn for a while before it can detect the smoke and
13 heat generated and then trigger the fire alarm (Zhang et al., 2016). These sensors keep looking
14 at a point and unfortunately the fire may not affect that point. In terms of reliability, the
15 positional distribution of these sensors come to play as they must be densely distributed for the
16 system to be more accurate (Gunawaardena et al., 2016).
17
18
19
20

21 Recently, the Federal Capital Territory Administration (FCTA) mandated the use of active
22 CCTV (Closed-circuit Television) in all public spaces including schools, hospitals, offices and
23 commercial buildings within the city to strengthen national security. Furthermore, the Abuja
24 Metropolitan Management Council (AMMC) added that the use of CCTV will be included in
25 the mandatory requirements for building plan approval in the city (Yahaya, 2022). It's only a
26 matter of time before this same plan spreads across the nation. Before this recent development
27 from FCTA, most commercial complexes in Nigeria resort to CCTV technology due to safety
28 reasons like preventing theft, abduction, sexual harassment and many more. Some fire
29 management strategies even go as far as using camera technology to monitor fire incidents at
30 various locations within the complex.
31
32
33
34

35 However, this is ineffective in a situation where there are hundreds of cameras displayed
36 simultaneously on a single monitoring screen because it takes minutes to locate the location on
37 fire as cameras must take turns before they are displayed (Wu et al., 2019). In cases where there
38 are multiple dedicated monitoring screens, the human operator in charge still spends time trying
39 to locate the camera. Time is of utmost importance in fire detection as any delay in the system
40 can cause damage, hence a need to build a system with minimum detection latency. **Prior**
41 **studies have examined early fire detection systems, and recent research has further**
42 **explored AI-driven approaches for enhanced early detection. For example, IoT-based**
43 **fuzzy models and sensor-integrated AI systems have been developed for fire detection in**
44 **Nigerian gas plants and East African local markets (Lule et al., 2020; Ohanu & Egesimba,**
45 **2024). Additionally, remote sensing and AI-powered surveillance have been applied to**
46 **mitigate banditry (Funtua et al., 2025), model forest fire risks, and enhance urban safety**
47 **in Nigeria (Ogunbadewa, 2015; Enoh et al., 2021; Ojomah et al., 2025). However, there**
48 **remains a lack of consolidated research linking fire detection to commercial complexes**
49 **in the Nigerian context, where datasets are limited and traditional systems still suffer**
50 **from false alarms and detection delays.** In addition, sensor signal processing come with high
51 implementation cost (sensor density for complexes) and even wasted investments especially
52 when there is no eventual fire outbreak at the sensor location. This study seeks to augment the
53 preexisting system by introducing a computer vision deep learning network algorithm for fire
54
55
56
57
58
59
60

1
2
3 disaster management in commercial complexes. This algorithm uses images from the
4 preexisting CCTV camera used for security surveillance in most public places to detect fires
5 early and reduce false alarms.
6
7

8
9 To sum it up, sensor signal processing is deficient in dealing with this problem due to the
10 following: 1) inefficiencies in distinguishing between cooking fires and real fire outbreaks 2)
11 inability to determine fire size, location and growth rate 3) the need for sensors to be in close
12 range because they only detect a fire point and cannot take care of slow-smouldering fires
13 whose characteristics do not become obvious immediately burning starts, thereby causing
14 detection latency (Chen et al., 2003). Hence the need to develop a better and more efficient
15 system for fire detection in the Nigerian context. The algorithm developed in this study will
16 help to minimize false alarms and encourage investors to make their business case without fear
17 of losing commercial properties to fire. As a nation, this will help and save cost of fire
18 interventions from the government for the economy because financial resources can be diverted
19 to grow other sectors of the economy.
20
21
22

23 **Computer Vision Based Fire Detection**

24
25 Fire detection systems built on computer vision have overcome all these limitations of
26 traditional or conventional fire detection systems because it detects the combustible itself and
27 not only its by-product when used in hybrid systems (multi-sensor and computer vision).
28 Computer vision is an artificial intelligence branch concerned with teaching robots to view the
29 world as humans do. Other tasks covered include image detection, tagging, classification,
30 recognition, image analysis, natural language processing, video analysis, and so on (Bhatt et
31 al., 2021). The bulk of deep learning methodologies in the computer vision industry are built
32 using Convolutional Neural Networks (CNN). CNN is a deep learning system that analyses an
33 input picture and assigns varying levels of relevance (weights and learnable biases) to objects
34 within the image, allowing one object characteristic to be distinguished from another. It is the
35 most effective learning technique for processing image input when compared to other deep
36 learning approaches (Khan et al., 2020). It has also exhibited exceptional image recognition,
37 classification, segmentation, and retrieval capabilities (Liu et al., 2019).
38
39
40
41
42

43
44 **While CNN and RCNN architectures form the foundation of many fire detection systems,**
45 **alternative models like YOLO (You Only Look Once) and U-Net have shown significant**
46 **promise, particularly in scenarios requiring real-time performance and spatial precision**
47 **(Lee & Kim, 2020). YOLO's unified architecture offers rapid object detection by**
48 **processing images in a single pass, which is highly advantageous in time-sensitive fire**
49 **incidents. Comparative studies have shown that YOLO outperforms CNNs in detection**
50 **speed without a substantial drop in accuracy (Raj et al., 2021). Similarly, U-Net's ability**
51 **to perform fine-grained segmentation makes it suitable for mapping fire boundaries**
52 **within complex structures (Zhang et al., 2021). Recent hybrid models such as YOLOv5-**
53 **U-Net and ATT Squeeze U-Net have specifically addressed fire detection and**
54 **segmentation challenges in structured environments like forests and urban settings**
55 **(Mseddi et al., 2021; Zhang et al., 2021; Iriawan et al., 2024). These models combine fast**
56 **object detection with high-resolution feature mapping, which is critical for distinguishing**
57
58
59
60

fire from confounding elements such as lighting artifacts or reflective surfaces common in indoor commercial environments.

CCTV cameras work by taking a predefined sequence of images transmitted wirelessly or using cables to a recording device and then to a monitoring screen where the visual display can be seen as a video. This research seeks to develop a deep learning CNN algorithm for fire disaster management in commercial complexes in Nigeria. Security personnel or department in the commercial complex can give access to past CCTV footages (video or sequence of images) through their backend storage which can serve as an input data for the CNN to identify fire features in the system. An efficient algorithm will automatically learn to detect fires in real time, reduce miswarnings and alert fire emergency personnel and organizations early.

The selection of Convolutional Neural Networks (CNNs) for this study is both strategic and contextually appropriate for the Nigerian setting. CNNs have demonstrated exceptional image classification and feature extraction capabilities, making them well-suited for distinguishing between fire and non-fire images in surveillance footage. Unlike more complex architectures such as YOLO or U-Net, CNNs offer a balance between performance and computational efficiency, which is crucial in Nigeria where technical infrastructure, such as high-performance GPUs or robust cloud connectivity, is not always readily available in many commercial settings.

Furthermore, the dataset used in this study was manually curated from secondary sources like Google Images and reflects a relatively small and imbalanced sample size- a condition under which CNNs are known to perform reliably when combined with augmentation techniques. Given the resource constraints and the absence of centralized fire image databases in Nigeria, a CNN model provides a more feasible entry point for local deployment, offering good accuracy relatively low implementation overhead. Thus, the use of CNN aligns with both the technical limitations and the urgent need for a deployable, data-efficient fire detection system tailored to commercial complexes in Nigeria.

METHODOLOGY

Data Collection

Due to the absence of observational data, secondary data was used. The dataset was generated from Google Images using the Google Search Engine. Images generated contained different classes of fire ranging from commercial complex fires to smoke to non-fire regions within commercial complexes. Data contained 2 classes (folders) with fire and non-fire images as separate folders. Non-fire images were added to build a proper algorithm and train it, so that the dataset would not be skewed. **Table 1** detailed the search term used to generate the dataset.

Images for respective search term were downloaded from Google using the Download All Images Extension on Google Chrome. This extension helped download the images in batches within the search term, reduced the rigour of downloading one image at a time and increased

1
2
3 the download speed. However, the images downloaded contained various unrelated images to
4 the commercial setting and fire. Hence, to address quality issues and avoid overfitting, only
5 those that fit the search term and context were retained after scanning through all images. The
6 'Action' column in **Table 1** shows the number of images retained.
7
8
9

10 Sometimes, overfitting can occur when there is a mismatch between training and real-world
11 data that the model is intended to generalize. **Where the training data does not adequately**
12 **represent the real-life scenerio or contains biases, the model may learn and amplify those**
13 **biases, leading to poor performance on unseen data (Balawejder, 2022). Several studies**
14 **have explored strategies to address dataset limitations, including the use of synthetic data**
15 **augmentation (Arlovic et al., 2025), crowd-sourced annotations (Wang et al., 2024), and**
16 **transfer learning from pre-trained models to improve generalization (Gupta & Mishra,**
17 **2024). These approaches aim to mitigate bias and enhance representativeness,**
18 **particularly when primary data sources like Google Images are inherently unbalanced or**
19 **lack diversity.**
20
21
22
23

24 Practically, CCTV footage from commercial complex can be used to generate these set of
25 images. The security administrators can generate a sequence of images from live sessions.
26 There are various tools to achieve this from a big data perspective, although it might involve
27 some additional software and cost. Finally, search terms 1 to 6 was combined into a folder
28 named 'Fire' while 7 and 8 made up the 'no fire' folder. The fire folder contained 200 images
29 while the 'no fire' folder contained 75 images at this initial stage.
30
31
32

33 **Software Used for Writing the CNN Algorithm**

34 Google Colaboratory (Google Colab) was utilised for building this Convolutional Neural
35 Network (CNN) algorithm due to the platform's numerous advantages. Firstly, the accessibility
36 and convenience offered by this software was paramount in the decision-making process. The
37 cloud-based environment provided seamless access to take next steps on the project even if
38 computer experienced a crash or got bad along the line. It enabled working on the project
39 without the need for complex local machine setup and configuration, ensuring the algorithm's
40 development could proceed efficiently and unhindered by geographical constraints (Sharma,
41 2020).
42
43
44
45

46 Moreover, the cost-effectiveness of Google Colab was a significant factor. Google Colab
47 boasted a rich ecosystem of libraries and tools specifically designed for deep learning tasks.
48 The platform seamlessly integrated with popular frameworks like TensorFlow and PyTorch,
49 providing access to extensive functionalities and pre-trained models. These tools streamlined
50 the development process and improved the accuracy and performance of the CNN algorithm
51 (Das, 2022).
52
53
54

55 **Why Region-based Convolutional Neural Network (RCNN)**

56 Employing the RCNN methodology in training a fire dataset was a highly suitable approach
57 because RCNN effectively addresses the challenge of object detection and localization, making
58 it particularly relevant for identifying and analysing fires in images. The region proposal
59 network, RCNN can accurately localize fire instances within the dataset, contributing to robust
60

and precise fire detection (Gandhi, 2018). Moreover, RCNN demonstrated exceptional performance in handling complex and diverse datasets (Zhang et al., 2023). Fires can exhibit significant variations in terms of scale, shape, and appearance, making it crucial to employ a neural network capable of handling such complexities. RCNN's ability to learn discriminative features at multiple scales and capture contextual information enabled it to effectively model the variations present in fire images (Li & Zhao, 2020). Consequently, RCNN exhibited strong performance in accurately detecting fires across a wide range of scenarios, thereby justifying its suitability for training a fire dataset (Li & Zhao, 2020; L. Zhang et al., 2023).

Pre-process Dataset

At this stage, the images initially downloaded during data collection were pre-processed and transformed to avoid overfitting and reduce training time. Overfitting is a common problem that occurs when training machine learning algorithms. It happens when a model starts to fit the training data too closely (Rice et al., 2020). One of the reasons why overfitting can happen is training an algorithm with insufficient or perfect training data. When the training dataset is small, the model may have limited exposure to different patterns and variations. As a result, it may memorize the training examples instead of learning the underlying generalizable patterns. This leads to overfitting as the model becomes too specialized in the training data and fails to perform well on unseen data (Lawrence et al., 1997).

To mitigate overfitting, the size of the overall dataset was increased so that the training dataset can also increase. **Table 2** detailed the actual augmentation applied on each image search term category as a way of transforming the initial dataset and increasing it. These transformations are needed to introduce noise and reduce the perfectness of the dataset so that it can predict more accurately (DeVries & Taylor, 2017). Some of the additional transformations carried out were rotation, flipping, cropping, stretching, affine and many more. 450 images were generated in total containing 300 fire-related images (search terms 1 to 6 combined) and 150 non-fire images (search term 7 and 8 combined). The python code used to augment these set of images are detailed in **Appendices 1 to 5**. In addition, the dataset was resized to 256 x 256 x 3 pixels.

Split Dataset into Training and Test Sets

According to Muraina (2022), the 90:10 ratio of splitting data gave the best results across different classification approaches. So, the pre-processed dataset of this study was split into training and test dataset in a 90:10 ratio. The training dataset contained 405 images while the test dataset contained 45 images. Both datasets were split in such a way that all fire scenarios were replicated in each set. The training dataset were used in training the algorithm to classify fire while the test datasets were used to evaluate and assess the effectiveness and efficiency of the trained network. During training, 5% (0.05) of the training dataset were treated as the validation dataset which is used to get the accuracy of the algorithm on new dataset.

Design of the CNN Architecture

The RCNN (Region-based Convolutional Neural Network) architecture employed in the full code (**Appendix 6**) was a well-defined methodology for object detection and classification. The model was constructed using the Sequential API in Keras, and it consisted of multiple convolutional and fully connected layers to maximise feature extraction from the input images

(Jogin et al., 2018). The model started with the first convolutional layer, which comprised 32 filters of size 3x3. The rectified linear unit (ReLU) activation function was applied to introduce non-linearity as used in many deep learning algorithms. Subsequently, a max pooling layer with a pool size of 2x2 was added to downsample the feature maps. Next, the second convolutional layer was incorporated into the architecture. It consisted of 64 filters with a size of 3x3 and utilized the ReLU activation function. Like the previous layer, a max pooling layer was added to perform downsampling.

The third convolutional layer, composed of 128 filters with a size of 3x3, followed suit. Again, the ReLU activation function was applied, and a max pooling layer was added for downsampling. To prepare for classification, the feature maps were flattened, converting them into a one-dimensional vector. This step ensured compatibility with the subsequent fully connected layers. The fully connected layers were introduced to perform higher-level feature extraction and classification. The first fully connected layer consisted of 256 neurons, with the ReLU activation function applied. Dropout regularization with a rate of 0.2 was incorporated to mitigate overfitting. Following that, another fully connected layer with 128 neurons and ReLU activation was added, accompanied by dropout regularization with the same rate of 0.2. Finally, the output layer was constructed with two neurons, utilizing the softmax activation function. This configuration allowed the model to predict the probabilities of the two classes, indicating the presence or absence of the target object.

Train the Model

To start the training process, the model was compiled by specifying the optimizer, loss function, and evaluation metric. The optimizer, in this case, was set to 'adam,' which is a popular optimization algorithm used to adjust the model's internal parameters during training. Its goal is to find the optimal values that minimize the loss function (Hassan et al., 2023). The loss function was set to 'binary_crossentropy,' which is commonly used for binary classification problems. It measures the dissimilarity between the predicted outputs and the true labels, providing a measure of how well the model is performing (Kong & Cheng, 2021). The chosen evaluation metric was 'accuracy,' which indicates the proportion of correctly classified samples in relation to the total number of samples. It serves as a measure of how accurately the model predicts the correct class labels.

To prevent overfitting, the code included the Early Stopping technique. Early stopping is a strategy that monitors the validation loss during training and stops the training process when the loss stops improving or starts to deteriorate (Pasquadibisceglie et al., 2019). The EarlyStopping callback from TensorFlow's library was utilized to implement this technique. It was configured with specific parameters to control its behaviour. The monitor parameter was set to 'val_loss,' indicating that the validation loss would be monitored. The mode parameter was set to 'auto,' allowing the callback to automatically determine whether the validation loss should be minimized or maximized based on the training configuration.

A min_delta value of 0 was specified, meaning that even a very small improvement in the validation loss would be considered as an improvement. The patience parameter was set to 2,

1
2
3 which means that if there is no improvement in the validation loss for two consecutive epochs,
4 the training process would be stopped. Finally, the `restore_best_weights` parameter was set to
5 True. This ensured that the model's weights were restored to the configuration that achieved
6 the best performance on the validation set. By doing so, the model retained the optimal
7 parameters, which helped prevent overfitting.
8
9

10
11 The model was trained using the `fit` method. The training data, represented by `Train_IMG_Set`,
12 was provided as input. Additionally, the validation data, represented by `Validation_IMG_Set`,
13 was specified using the `validation_data` parameter. During training, the `EarlyStopping` callback
14 was included by passing it as an element of a list to the `callbacks` parameter. This enabled the
15 monitoring of the validation loss and the application of the early stopping mechanism. The
16 training process was conducted over 100 epochs, as specified by the `epochs` parameter. The
17 `verbose` parameter was set to 2, which enabled detailed logging output during training,
18 providing visibility into the progress of the training process.
19
20
21

22 Evaluate Model

23 The `evaluate` method was applied to the `Model_Three` object, with the test data `Test_IMG_Set`
24 as the input. The evaluation process assessed the performance of the algorithm on the test
25 dataset. The model made predictions on the test images, and the evaluation metrics, including
26 the loss and accuracy, were calculated. After the evaluation was completed, the results were
27 stored in the `Model_Results_Three` variable.
28
29
30

31 Test Model on Unseen Data

32 The final step involved testing the model on the "unseen data" or "test data." This dataset
33 consisted of samples that were not used during the model's training phase. The purpose of using
34 unseen data was to evaluate the model's ability to correctly predict outcomes on previously
35 unseen samples, thus reflecting its real-world performance. The testing process involved
36 feeding the unseen data into the trained model and obtaining predictions for each sample. These
37 predictions are then compared to the ground truth or actual labels of the test data to assess the
38 model's accuracy and performance. This comparison provided insights into how well the model
39 could generalize its learned patterns and make accurate predictions on new, unseen samples.
40
41
42
43

44 **While accuracy is essential, computational efficiency is equally critical in real-time fire**
45 **detection systems. The model's 100-epoch training process and use of early stopping**
46 **aimed to optimize performance while reducing unnecessary computation. However, real-**
47 **time deployment may require further model compression or pruning to balance accuracy**
48 **with latency. Future improvements could explore quantization techniques to ensure**
49 **timely predictions without significantly compromising detection precision in resource-**
50 **constrained environments.**
51
52
53
54

55 RESULTS AND DISCUSSION

56 During the training phase, the model's performance was monitored using the training loss and
57 accuracy metrics. In the initial epoch, the loss was observed to be 0.7445, indicating a relatively
58 high error rate. The accuracy achieved on the training set was 0.6130, suggesting that the model
59
60

1
2
3 correctly classified approximately 61.30% of the training samples. However, on the validation
4 set, the model's performance was not satisfactory, with a higher loss of 0.9304 and an accuracy
5 of 0.5000, indicating random guessing. As the training progressed to subsequent epochs, the
6 model showed signs of improvement. The loss gradually decreased, indicating a reduction in
7 the prediction error. The accuracy on the training set increased as well, indicating better
8 classification performance. For example, in epoch 16, the model achieved a loss of 0.3359 and
9 an accuracy of 0.8312 on the training set (Figure 1).
10
11
12

13
14 These results suggest that the model improved in capturing the underlying patterns and features
15 related to fire detection. Based on this dataset, the model struggled to achieve satisfactory
16 performance on the validation set initially, with a validation accuracy of 0.5000 in the first few
17 epochs. However, as the training progressed, the validation accuracy improved and reached up
18 to 0.9000 in epoch 14. This improvement suggests that the model learned to capture more
19 relevant features and generalize better to unseen fire instances. The training process took a
20 considerable amount of time, with each epoch varying in duration, ranging from 95 to 108
21 seconds. The patience parameter set to 2 made the training stop at epoch 16 since there was no
22 improvement in the validation loss for two consecutive epochs. Figure 2 shows the progression
23 of the training loss and validation loss across different epochs.
24
25
26

27 **Model Evaluation**

28
29 After the training process, the trained CNN model was evaluated on a separate test dataset to
30 assess its performance on unseen data. The evaluation metrics used were the loss and accuracy.
31 The evaluation of the model yielded a loss value of 0.5452 and an accuracy of 0.73 (Figure 3).
32 The loss value represents the discrepancy between the predicted outputs of the model and the
33 true labels of the test data. A lower loss value indicates a better alignment between the predicted
34 and actual values. In this case, the obtained loss value of 0.5452 suggests that the model's
35 predictions were reasonably close to the true labels on average. The accuracy metric represents
36 the proportion of correctly classified samples out of the total number of samples in the test
37 dataset. In this evaluation, the model achieved an accuracy of 0.73, indicating that it correctly
38 classified approximately 73% of the test. Based on these evaluation results, it can be inferred
39 that the trained model performed reasonably well on the test dataset. The loss value of 0.5452
40 indicates that the model's predictions were relatively accurate, and the achieved accuracy of
41 0.73 suggests that the model was able to classify most of the test samples correctly.
42
43
44
45
46

47 **Classification Report**

48 The classification report in Table 3 provided a detailed analysis of the model's performance for
49 each class in the test dataset. It included metrics such as precision, recall, F1-score, and support.
50 For Class 0, the precision was 0.85, indicating that out of all the samples predicted as Class 0,
51 85% were classified correctly. The recall, also known as sensitivity or true positive rate, was
52 0.65, indicating that the model identified 65% of the true Class 0 samples. The F1-score, which
53 was the harmonic mean of precision and recall, was 0.73. The F1-score provided a balanced
54 measure of the model's accuracy in terms of both precision and recall. The support referred to
55 the number of samples in Class 0 in the test dataset, which were 34.
56
57
58
59
60

For Class 1, the precision was 0.37, indicating that only 37% of the samples predicted as Class 1 were classified correctly. The recall was 0.64, meaning that the model identified 64% of the true Class 1 samples. The F1-score was 0.47, indicating a relatively lower accuracy for Class 1 predictions. The support for Class 1 was 11, representing the number of samples in that class. The accuracy of the model was calculated as 0.64, meaning that it correctly classified approximately 64% of the total test samples. In terms of the macro average, which calculated the average performance across all classes, the precision was 0.61, the recall was 0.64, and the F1-score was 0.60. The weighted average considered the support for each class.

It calculated the average performance weighted by the number of samples in each class. In this case, the weighted average precision was 0.73, the recall was 0.64, and the F1-score was 0.67. Based on this classification report, it can be inferred that the model had better performance in Class 0, as indicated by higher precision, recall, and F1-score compared to Class 1. The model achieved an overall accuracy of 0.64, suggesting that it was able to classify most of the samples correctly. However, it is important to note that the lower precision, recall, and F1-score for Class 1 indicated that the model had more difficulty in accurately identifying samples belonging to that class samples.

Confusion Matrix

Table 4 shows the confusion matrix, which provided a summary of the model's predictions compared to the actual labels for the test dataset. It consisted of four cells representing the number of samples for each combination of predicted and actual classes.

In the case of the confusion matrix presented:

- There were 22 samples that were predicted as positive (Class 1) and were positive (Actual Positive). This indicates that the model correctly identified 22 positive samples.
- There were 12 samples that were predicted as positive but were negative (Actual Negative). This suggests that the model had 12 false positive predictions.
- There were 4 samples that were predicted as negative (Class 0) but were positive. These were false negative predictions.
- Finally, there were 7 samples that were predicted as negative and were negative. These were true negative predictions.

In this case, the model had a higher number of false positives (12) compared to false negatives (4). This implies that the model tended to incorrectly classify some negative samples as positive. However, it also correctly identified most of the negative samples as negative.

Model Performance on Unseen Data

The model's performance on unseen data showed that it was able to correctly predict most fire images by returning label 0, as seen in Figure 4. This indicates that the model had learned important features and patterns associated with fire images during training. However, the model exhibited incorrect predictions on new non-fire images. It incorrectly classified some non-fire images as fire (label 0). This suggests that the model had difficulty distinguishing between fire and non-fire images that it had not encountered during training. It might not have

1
2
3 learned sufficient representations of non-fire images, leading to misclassifications in this
4 category.
5
6

7 **CONCLUSION AND RECOMMENDATIONS**

8 Throughout the research process, several key steps were undertaken. Data collection involved
9 gathering a diverse set of fire images specific to Nigerian commercial complexes, ensuring the
10 dataset's relevance to the local context. The RCNN model was developed and then trained on
11 this dataset, because of its ability to detect and classify objects within an image. The model's
12 performance was evaluated using both training and validation datasets, measuring metrics such
13 as accuracy, loss, precision, recall, and F1-score. Furthermore, the model's performance was
14 assessed on unseen data to evaluate its real-world applicability.
15
16
17
18

19 The analysis of the model's performance on unseen data revealed its effectiveness in correctly
20 predicting fire images with label 0, indicating its potential for early fire detection. This
21 capability can significantly contribute to the timely response to fire incidents, thereby
22 minimizing potential damage and ensuring public safety. However, it is worth noting that the
23 model exhibited limitations in correctly classifying new non-fire images, suggesting the need
24 for further improvement and fine-tuning. This highlights the importance of continuously
25 refining the model to improve its ability to accurately differentiate between fire and non-fire
26 instances.
27
28
29
30

31 The application of RCNN (Region-based Convolutional Neural Network) to train a fire dataset
32 in Nigeria has yielded significant findings and advancements in fire detection and management.
33 The developed RCNN model has demonstrated promising performance in accurately
34 identifying fire images, contributing to the improvement of fire monitoring and prevention
35 capabilities in Nigerian commercial complexes. This research has provided valuable insights
36 into the challenges and opportunities in fire detection using computer vision techniques in the
37 specific context of Nigerian commercial complexes.
38
39
40

41 Based on the findings and research process, the following recommendations are proposed for
42 future work in Nigerian commercial complexes:
43
44

- 45 1. **Dataset Enhancement:** Continuously expand and enrich the fire dataset by incorporating
46 more diverse images that capture different fire scenarios and environmental conditions
47 specific to Nigerian commercial complexes. This may include variations in lighting
48 conditions, angles, and types of fire incidents. By increasing the dataset's size and
49 diversity, the model can learn to generalize better and improve its accuracy in real-
50 world scenarios.
51
52
53
- 54 2. **Non-Fire Image Incorporation:** Pay special attention to including a comprehensive
55 collection of non-fire images that represent various objects and scenes commonly found
56 in commercial complexes. This will enable the model to better differentiate between
57 fire and non-fire instances, reducing false predictions on non-fire images. Incorporating
58
59
60

a wide range of non-fire instances, such as people, furniture, and equipment, will improve the model's ability to discern fire-related features accurately.

3. Fine-Tuning and Transfer Learning: Explore fine-tuning techniques and take advantage of pre-trained models on large-scale datasets to improve the performance of the RCNN model. Pre-trained models, such as those trained on general object recognition tasks, can provide a valuable starting point. By transferring knowledge from these models to the specific fire dataset, the model can benefit from the learned features and improve its accuracy, particularly in cases with limited training data.
4. **Ethical issues: Beyond technical performance, this research raises important ethical considerations. The deployment of fire detection systems using computer vision in Nigerian commercial complexes must ensure data privacy, particularly when CCTV footage is involved. It is essential to implement strict data protection measures and obtain informed consent where applicable. Addressing these ethical aspects is vital to ensure that such technological solutions are trustworthy, equitable, and socially responsible.**
5. Collaboration and Knowledge Exchange: Foster collaboration among researchers, industry experts, and stakeholders to exchange knowledge and expertise in the field of fire detection and management. Collaborative efforts can lead to the development of more robust and context-specific solutions tailored to Nigerian commercial complexes. This can include sharing datasets, algorithms, and best practices, as well as conducting joint research projects and workshops to promote knowledge transfer and collaboration.

Conclusively, this research has demonstrated the potential of RCNN for fire detection in Nigerian commercial complexes. By building on these findings and following the proposed recommendations, future work can advance fire monitoring and prevention capabilities, foster indigenous expertise in computer vision and deep learning, and position Nigeria at the forefront of technological advancements in fire management. The insights gained from this research contribute to the growing body of knowledge in fire detection and highlight the importance of taking advantage of advanced computer vision techniques for better safety and security in Nigerian commercial complexes.

REFERENCES

- Abeku, T., Michael, D., Akpan-Nsoh, I., Udejah, G., Ogugbuaja, C., Oyewole, R., Godwin, A., & Agboluaje, R. (2021). Losses to market fires hit N41.54 billion in two years. <https://guardian.ng/news/losses-to-market-fires-hit-n41-54-billion-in-two-years/>
- Arlovic, M., Hrzic, F., Patel, M., Bednarz, T., & Balen, J. (2025). Evaluation of synthetic data impact on fire segmentation models performance. *Scientific reports*, 15(1), 1-14.

- 1
2
3 Balawejder, M. (2022). Overfitting in Deep Learning. Towards Data Science.
4 [https://towardsdatascience.com/overfitting-in-deep-learning-what-is-it-and-how-to-](https://towardsdatascience.com/overfitting-in-deep-learning-what-is-it-and-how-to-combat-it-9760d25ad05b)
5 [combat-it-9760d25ad05b](https://towardsdatascience.com/overfitting-in-deep-learning-what-is-it-and-how-to-combat-it-9760d25ad05b)
6
7
8 Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K., & Ghayvat, H.
9 (2021). CNN variants for computer vision: History, architecture, application, challenges and
10 future scope. *Electronics*, 10(20), 2470.
11
12 Chen, S., Bao, H., Zeng, X., & Yang, Y. (2003). A fire detecting method based on multi-sensor
13 data fusion. *SMC'03 Conference Proceedings. 2003 IEEE International Conference on*
14 *Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat.*
15 *No. 03CH37483)*, 4, 3775–3780.
16
17
18 Chime, V. (2021). Fire guts Next Cash and Carry supermarket in Abuja.
19 <https://www.thecable.ng/fire-guts-next-cash-and-carry-supermarket-in-abuja>
20
21
22 Das, T. (2022). Google Colab: Everything you Need to Know. Geekflare.
23 <https://geekflare.com/google-colab/>
24
25
26 DeVries, T., & Taylor, G. W. (2017). Improved regularization of convolutional neural
27 networks with cutout. *ArXiv Preprint ArXiv:1708.04552*.
28
29
30 Enoh, M. A., Okeke, U. C., & Narinua, N. Y. (2021). Identification and modelling of forest
31 fire severity and risk zones in the Cross–Niger transition forest with remotely sensed
32 satellite data. *The Egyptian Journal of Remote Sensing and Space Science*, 24(3), 879-887.
33
34
35 EPS Security. (2019). Five essential fire statistics for business owners.
36 [https://www.epssecurity.com/news/business-security/five-essential-fire-statistics-for-](https://www.epssecurity.com/news/business-security/five-essential-fire-statistics-for-business-owners/)
37 [business-owners/](https://www.epssecurity.com/news/business-security/five-essential-fire-statistics-for-business-owners/)
38
39
40 Funtua, D. I. A., Yargamji, D. G. I., & Hussaini, K. (2025). Systematic Review of AI-Powered
41 Surveillance Systems and Stem Education Initiatives for Mitigating Banditry in Nigeria.
42 *Yobe Journal of Educational Studies (Yojes)*, 87.
43
44
45 Gandhi, R. (2018). R-CNN, Fast R-CNN, Faster R-CNN, YOLO - Object Detection
46 Algorithms. Towards Data Science. [https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-](https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e)
47 [r-cnn-yolo-object-detection-algorithms-36d53571365e](https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e)
48
49
50 Gunawaardena, A. E., Ruwanthika, R. M. M., & Jayasekara, A. (2016). Computer vision-based
51 fire alarming system. 2016 Moratuwa Engineering Research Conference (MERCon), 325–
52 330.
53
54
55 Gupta, H. P., & Mishra, R. (2024). Utilizing transfer learning and pre-trained models for
56 effective forest fire detection: A case study of uttarakhand. *arXiv preprint*
57 *arXiv:2410.06743*.
58
59
60 Hassan, E., Shams, M. Y., Hikal, N. A., & Elmougy, S. (2023). The effect of choosing
optimizer algorithms to improve computer vision tasks: a comparative study. *Multimedia*
Tools and Applications, 82(11), 16591–16633.

- Iriawan, N., Pravitasari, A. A., Nuraini, U. S., Nirmalasari, N. I., Azmi, T., Nasrudin, M., ... & Ferriastuti, W. (2024). YOLO-UNet Architecture for Detecting and Segmenting the Localized MRI Brain Tumor Image. *Applied Computational Intelligence and Soft Computing*, 2024(1), 3819801.
- Jogin, M., Madhulika, M. S., Divya, G. D., Meghana, R. K., & Apoorva, S. (2018). Feature extraction using convolution neural networks (CNN) and deep learning. 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2319–2323.
- Khan, A., Sohail, A., Zahoor, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53, 5455–5516.
- Kong, L., & Cheng, J. (2021). Based on improved deep convolutional neural network model pneumonia image classification. *PloS One*, 16(11), e0258804.
- Kumar, A., Khare, R., Sankat, S., & Madhavi, P. (2023). Fire safety assessment for older adults in high-rise residential buildings in India: A comprehensive study. *International journal of building pathology and adaptation*, 41(3), 625-646.
- Lawal, I. M., & Ajayi, J. M. A. (2019). The role of Islamic social finance towards alleviating the humanitarian crisis in North-East Nigeria. *Jurnal Perspektif Pembiayaan Dan Pembangunan Daerah*, 6(5), 545–558.
- Lawrence, S., Giles, C. L., & Tsoi, A. C. (1997). Lessons in neural network training: Overfitting may be harder than expected. *Aaai/Iaai*, 540–545.
- Lee, Y. H., & Kim, Y. (2020). Comparison of CNN and YOLO for Object Detection. *Journal of the semiconductor & display technology*, 19(1), 85-92.
- Li, P., & Zhao, W. (2020). Image fire detection algorithms based on convolutional neural networks. *Case Studies in Thermal Engineering*, 19, 100625.
- Liu, X., Deng, Z., & Yang, Y. (2019). Recent progress in semantic image segmentation. *Artificial Intelligence Review*, 52, 1089–1106.
- Lule, E., Mikeka, C., Ngenzi, A., & Mukanyiligira, D. (2020). Design of an IoT-based fuzzy approximation prediction model for early fire detection to aid public safety and control in the local urban markets. *Symmetry*, 12(9), 1391.**
- Mseddi, W. S., Ghali, R., Jmal, M., & Attia, R. (2021, August). Fire detection and segmentation using YOLOv5 and U-net. In *2021 29th European Signal Processing Conference (EUSIPCO)* (pp. 741-745). IEEE.
- Muraina, I. (2022). Ideal dataset splitting ratios in machine learning algorithms: general concerns for data scientists and data analysts. 7th International Mardin Artuklu Scientific Research Conference, 496–504.

- 1
2
3 Murray, M., & Watson, P. K. (2019). Adoption of natural disaster preparedness and risk
4 reduction measures by business organisations in Small Island Developing States-A
5 Caribbean case study. *International Journal of Disaster Risk Reduction*, 39, 101115.
6
7
8 Mwedzi, N. A., Nwulu, N. I., & Gbadamosi, S. L. (2019). Machine learning applications for
9 fire detection in a residential building. 2019 IEEE 6th International Conference on
10 Engineering Technologies and Applied Sciences (ICETAS), 1–4.
11
12 **Ogunbadewa, E. Y. (2015). Identifying active fire in southwestern Nigeria with MODIS**
13 **data and geographical information systems. *Geodesy and Cartography*, 41(2), 81-91.**
14
15
16 **Ohanu, I. B., & Egesimba, U. S. (2024). Fire detection and suppression in gas-plants using**
17 **automatic sensors among electrical/electronic technology education graduates in Nigeria.**
18 ***Energy Exploration & Exploitation*, 42(6), 2092-2107.**
19
20
21 **Ojomah, B. C., Umeh, C. L., Adum, N. N., Odoh, N. C., & Onyekwere, C. E. (2025,**
22 **March). Impact of Artificial Intelligence in Sustainable Forest Management in Nigeria.**
23 ***In e-Proceedings of the Faculty of Agriculture International Conference (pp. 240-245).***
24
25
26 Oladokun, V. O., & Emmanuel, C. G. (2014). Urban market fire disasters management in
27 Nigeria: A damage minimization based fuzzy logic model approach. *International Journal*
28 *of Computer Applications*, 106(17).
29
30
31 Oladokun, V. O., & Isohola, F. A. (2010). A risk analysis model for fire disasters in
32 commercial complexes in Nigeria.
33
34 Pasquadibisceglie, V., Appice, A., Castellano, G., & Malerba, D. (2019). Using convolutional
35 neural networks for predictive process analytics. 2019 International Conference on Process
36 Mining (ICPM), 129–136.
37
38 Punch Newspapers. (2021). CCTV: Footage Of 9-Year-Old Suspect Setting Prince Ebeano
39 Supermarket, Abuja On Fire. <https://www.youtube.com/watch?v=BYchb11XcKI>
40
41 Raj, R., Nagaraj, S. S., Ritesh, S., Thushar, T. A., & Aparanji, V. M. (2021, September). Fruit
42 classification comparison based on cnn and yolo. In *IOP Conference Series: Materials*
43 *Science and Engineering* (Vol. 1187, No. 1, p. 012031). IOP Publishing.
44
45
46 Rice, L., Wong, E., & Kolter, Z. (2020). Overfitting in adversarially robust deep learning.
47 *International Conference on Machine Learning*, 8093–8104.
48
49
50 Sharma, A. (2020). Google Colab for Machine Learning and Deep Learning.
51 [https://www.analyticsvidhya.com/blog/2020/03/google-colab-machine-learning-deep-](https://www.analyticsvidhya.com/blog/2020/03/google-colab-machine-learning-deep-learning/)
52 [learning/](https://www.analyticsvidhya.com/blog/2020/03/google-colab-machine-learning-deep-learning/)
53
54
55 Shi, L., Long, F., Lin, C., & Zhao, Y. (2017). Video-based fire detection with saliency detection
56 and convolutional neural networks. *Advances in Neural Networks-ISNN 2017: 14th*
57 *International Symposium, ISNN 2017, Sapporo, Hakodate, and Muroran, Hokkaido, Japan,*
58 *June 21–26, 2017, Proceedings, Part II 14, 299–309.*
59
60

- 1
2
3 Wang, M., Yue, P., Jiang, L., Yu, D., Tuo, T., & Li, J. (2024). An open flame and smoke
4 detection dataset for deep learning in remote sensing based fire detection. *Geo-spatial*
5 *Information Science*, 1-16.
6
7
8 Wu, H., Wu, D., & Zhao, J. (2019). An intelligent fire detection approach through cameras
9 based on computer vision methods. *Process Safety and Environmental Protection*, 127, 245–
10 256.
11
12 Yahaya, H. (2022, October 20). Security: FCTA Directs Installation Of CCTV In Public Places.
13 Daily Trust. [https://dailytrust.com/security-fcta-directs-installation-of-cctv-in-public-](https://dailytrust.com/security-fcta-directs-installation-of-cctv-in-public-places/)
14 [places/](https://dailytrust.com/security-fcta-directs-installation-of-cctv-in-public-places/)
15
16
17 Zhang, J., Zhu, H., Wang, P., & Ling, X. (2021). ATT squeeze U-Net: A lightweight network
18 for forest fire detection and recognition. *Ieee Access*, 9, 10858-10870.
19
20
21 Zhang, L., Wang, M., Ding, Y., & Bu, X. (2023). MS-FRCNN: A Multi-Scale Faster RCNN
22 Model for Small Target Forest Fire Detection. *Forests*, 14(3), 616.
23
24 Zhang, Q., Xu, J., Xu, L., & Guo, H. (2016). Deep convolutional neural networks for forest
25 fire detection. 2016 International Forum on Management, Education and Information
26 Technology Application, 568–575.
27
28
29 Zhou, X. L., Yu, F. X., Wen, Y. C., Lu, Z. M., & Song, G. H. (2010). Early fire detection based
30 on flame contours in video. *Information Technology Journal* , 9(5), 899–908.
31 <https://docsdrive.com/pdfs/ansinet/itj/2010/899-908.pdf>
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

Appendix 1 Image Augmentation code for search term 2 (Flip)

```
1
2
3
4
5
6   Import os
7
8   Import cv2
9
10
11   import shutil
12
13   def flip_image(image):
14
15       flipped_image = cv2.flip(image, 1) # Flip horizontally
16
17       return flipped_image
18
19
20
21
22
23
24   def transform_images(input_folder, output_folder):
25
26       # Create the output folder if it doesn't exist
27
28       if not os.path.exists(output_folder):
29
30           os.makedirs(output_folder)
31
32
33
34
35
36       # Loop through each file in the input folder
37
38       for filename in os.listdir(input_folder):
39
40           # Check if the file is an image
41
42           if filename.endswith(".jpg") or filename.endswith(".jpeg") or filename.endswith(".png"):
43
44               # Read the image
45
46               image_path = os.path.join(input_folder, filename)
47
48               image = cv2.imread(image_path)
49
50
51               # Apply the flip transformation to the image
52
53               transformed_image = flip_image(image)
54
55
56
57
58
59
60
```

```
1
2
3     # Generate a new filename for the transformed image
4
5     transformed_filename = f"flipped_{filename}"
6
7
8
9
10
11     # Save the transformed image to the output folder
12
13     transformed_path = os.path.join(output_folder, transformed_filename)
14
15
16     cv2.imwrite(transformed_path, transformed_image)
17
18
19
20
21     print(f"Flipped {filename} and saved as {transformed_filename}")
22
23
24
25
```

```
26 # Usage example
27
28
29 input_folder = "C:/Users/HP/Desktop/dataset/new 2"
30
31 output_folder = "C:/Users/HP/Desktop/dataset/output"
32
33 transform_images(input_folder, output_folder)
34
35
36
37
38
```

Appendix 2 Image Augmentation for Search Term 3 (Rotate and Zoom)

```
41 import os
42
43 import cv2
44
45 def rotate_image(image, angle):
46
47     height, width = image.shape[:2]
48
49     rotation_matrix = cv2.getRotationMatrix2D((width/2, height/2), angle, 1.0)
50
51     rotated_image = cv2.warpAffine(image, rotation_matrix, (width, height))
52
53
54     return rotated_image
55
56
57
58
59
60
```

```
1
2
3 def zoom_image(image, scale):
4
5     height, width = image.shape[:2]
6
7     zoomed_image = cv2.resize(image, (int(width * scale), int(height * scale)))
8
9
10
11     return zoomed_image
12
13
14
15
16 def transform_images(input_folder, output_folder):
17
18     # Create the output folder if it doesn't exist
19
20
21     if not os.path.exists(output_folder):
22
23         os.makedirs(output_folder)
24
25
26
27
28
29     # Loop through each file in the input folder
30
31     for filename in os.listdir(input_folder):
32
33         # Check if the file is an image
34
35
36         if filename.endswith(".jpg") or filename.endswith(".jpeg") or filename.endswith(".png"):
37
38
39             # Read the image
40
41             image_path = os.path.join(input_folder, filename)
42
43             image = cv2.imread(image_path)
44
45
46             # Zoom the image
47
48             zoomed_image = zoom_image(image, 1.2) # Zoom by 1.2 times
49
50
51
52
53
54             # Generate a new filename for the zoomed image
55
56             zoomed_filename = f"zoomed_{filename}"
57
58
59
60
```

```
1
2
3     # Save the zoomed image to the output folder
4
5     zoomed_path = os.path.join(output_folder, zoomed_filename)
6
7     cv2.imwrite(zoomed_path, zoomed_image)
8
9
10
11
12
13     # Rotate the image
14
15     rotated_image = rotate_image(image, 30) # Rotate by 30 degrees
16
17
18
19
20
21     # Generate a new filename for the rotated image
22
23     rotated_filename = f"rotated_{filename}"
24
25
26
27
28
29     # Save the rotated image to the output folder
30
31     rotated_path = os.path.join(output_folder, rotated_filename)
32
33     cv2.imwrite(rotated_path, rotated_image)
34
35
36
37
38
39     print(f"Transformed {filename} and saved as {zoomed_filename} and
40 {rotated_filename}")
41
42
43 # Usage example
44
45 input_folder = "C:/Users/HP/Desktop/dataset/new 3"
46
47 output_folder = "C:/Users/HP/Desktop/dataset/output"
48
49 transform_images(input_folder, output_folder)
50
51
52
53
54
55
```

Appendix 3 Image Augmentation for Search Term 5 (Stretch)

```
56
57
58 import os
59
60
```

```
1
2
3 import cv2
4
5
6 def stretch_image(image, scale_x, scale_y):
7
8     height, width = image.shape[:2]
9
10
11     stretched_image = cv2.resize(image, (int(width * scale_x), int(height * scale_y)))
12
13     return stretched_image
14
15
16
17
18 def stretch_images(input_folder, output_folder, scale_x, scale_y):
19
20     # Create the output folder if it doesn't exist
21
22
23     if not os.path.exists(output_folder):
24
25         os.makedirs(output_folder)
26
27
28
29
30
31     # Loop through each file in the input folder
32
33
34     for filename in os.listdir(input_folder):
35
36         # Check if the file is an image
37
38
39         if filename.endswith(".jpg") or filename.endswith(".jpeg") or filename.endswith(".png"):
40
41             # Read the image
42
43
44             image_path = os.path.join(input_folder, filename)
45
46
47             image = cv2.imread(image_path)
48
49
50
51
52             # Stretch the image
53
54
55             stretched_image = stretch_image(image, scale_x, scale_y)
56
57
58
59
60             # Generate a new filename for the stretched image
```

```
1
2
3     stretched_filename = f"stretched_{filename}"
4
5
6
7
```

```
8     # Save the stretched image to the output folder
9
```

```
10    stretched_path = os.path.join(output_folder, stretched_filename)
11
12
```

```
13    cv2.imwrite(stretched_path, stretched_image)
14
15
```

```
16
17
18    print(f"Stretched {filename} and saved as {stretched_filename}")
19
20
21
22
```

```
23
24 # Usage example
25
```

```
26 input_folder = "C:/Users/HP/Desktop/dataset/new 5"
27
28
```

```
29 output_folder = "C:/Users/HP/Desktop/dataset/output"
30
31
```

```
32 scale_x = 1.5 # Stretch factor for the x-axis
33
34
```

```
35 scale_y = 0.8 # Stretch factor for the y-axis
36
37
```

```
38 stretch_images(input_folder, output_folder, scale_x, scale_y)
39
40
41
```

42 **Appendix 4 Image Augmentation for Search Term 7 (Affine)**

```
43 Import os
44
```

```
45 Import cv2
46
47
```

```
48 Import numpy as np
49
50
```

```
51 def affine_transform_image(image, M):
52
```

```
53     transformed_image = cv2.warpAffine(image, M, (image.shape[1], image.shape[0]))
54
55
```

```
56     return transformed_image
57
58
59
60
```

```
1
2
3 def affine_transform_images(input_folder, output_folder, transformation_matrix):
4
5
6     # Create the output folder if it doesn't exist
7
8     if not os.path.exists(output_folder):
9
10        os.makedirs(output_folder)
11
12
13
14
15
16     # Loop through each file in the input folder
17
18     for filename in os.listdir(input_folder):
19
20
21        # Check if the file is an image
22
23
24        if filename.endswith(".jpg") or filename.endswith(".jpeg") or filename.endswith(".png"):
25
26            # Read the image
27
28            image_path = os.path.join(input_folder, filename)
29
30            image = cv2.imread(image_path)
31
32
33
34
35
36            # Apply the affine transformation to the image
37
38            transformed_image = affine_transform_image(image, transformation_matrix)
39
40
41
42
43
44            # Generate a new filename for the transformed image
45
46            transformed_filename = f"transformed_{filename}"
47
48
49
50
51            # Save the transformed image to the output folder
52
53            transformed_path = os.path.join(output_folder, transformed_filename)
54
55            cv2.imwrite(transformed_path, transformed_image)
56
57
58
59
60
```

```
1
2
3         print(f"Transformed {filename} and saved as {transformed_filename}")
4
5
6
7
8
9
```

```
10 # Usage example
11
```

```
12 input_folder = "C:/Users/HP/Desktop/dataset/new 6"
13
```

```
14 output_folder = "C:/Users/HP/Desktop/dataset/output"
15
```

```
16
17
18 # Define the affine transformation matrix
19
```

```
20
21 transformation_matrix = np.float32([[1.2, 0.3, 50], [0.2, 1.3, -20]])
22
```

```
23 affine_transform_images(input_folder, output_folder, transformation_matrix)
24
25
26
27
28
```

29 **Appendix 5 Image Augmentation applied for Search Term 8 (Crop)**

```
30 Import os
31
```

```
32
33 Import cv2
34
```

```
35
36 def crop_image(image, x, y, width, height):
37
```

```
38     cropped_image = image[y:y+height, x:x+width]
39
```

```
40     return cropped_image
41
42
43
44
45
```

```
46 def crop_images(input_folder, output_folder, x, y, width, height):
47
```

```
48     # Create the output folder if it doesn't exist
49
```

```
50
51     if not os.path.exists(output_folder):
52
```

```
53         os.makedirs(output_folder)
54
55
56
57
58
```

```
59     # Loop through each file in the input folder
60
```

```
1
2
3     for filename in os.listdir(input_folder):
4
5         # Check if the file is an image
6
7         if filename.endswith(".jpg") or filename.endswith(".jpeg") or filename.endswith(".png"):
8
9             # Read the image
10
11             image_path = os.path.join(input_folder, filename)
12
13             image = cv2.imread(image_path)
14
15             # Crop the image
16
17             cropped_image = crop_image(image, x, y, width, height)
18
19
20
21
22
23
24
25
26             # Generate a new filename for the cropped image
27
28             cropped_filename = f'cropped_{filename}'
29
30
31
32
33
34             # Save the cropped image to the output folder
35
36             cropped_path = os.path.join(output_folder, cropped_filename)
37
38             cv2.imwrite(cropped_path, cropped_image)
39
40
41
42
43
44             print(f"Cropped {filename} and saved as {cropped_filename}")
45
46
47
48
49 # Usage example
50
51 input_folder = "C:/Users/HP/Desktop/dataset/new 7"
52
53
54 output_folder = "C:/Users/HP/Desktop/dataset/output"
55
56
57
58
59
60 x = 50 # Starting x-coordinate for cropping
```

```
1
2
3     y = 100 # Starting y-coordinate for cropping
4
5
6     width = 250 # Width of the cropped region
7
8
9     height = 200 # Height of the cropped region
10
11
12
13
14     crop_images(input_folder, output_folder, x, y, width, height)
15
16
17
```

Appendix 6 Full RCNN Code for Fire Detection

```
18
19
20 # PACKAGES AND LIBRARIES
21
22
23 #GENERAL
24
25 import pandas as pd
26
27
28 import numpy as np
29
30
31 import seaborn as sns
32
33
34 import matplotlib.pyplot as plt
35
36 #PATH PROCESS
37
38 import os
39
40
41 import os.path
42
43
44 from pathlib import Path
45
46
47 import glob
48
49 #IMAGE PROCESS
50
51 from PIL import Image
52
53
54 from keras.preprocessing import image
55
56
57 from keras.utils import load_img, img_to_array
58
59
60 from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
1
2
3 import cv2
4
5
6 from keras.applications.vgg16 import preprocess_input, decode_predictions
7
8 #SCALER & TRANSFORMATION
9
10
11 from sklearn.preprocessing import StandardScaler
12
13
14 from sklearn.preprocessing import MinMaxScaler
15
16
17 from keras.utils.np_utils import to_categorical
18
19
20 from sklearn.model_selection import train_test_split
21
22
23 from keras import regularizers
24
25
26 from sklearn.preprocessing import LabelEncoder
27
28 #ACCURACY CONTROL
29
30
31 from sklearn.metrics import confusion_matrix, accuracy_score, classification_report,
32 roc_auc_score, roc_curve
33
34
35 from sklearn.model_selection import GridSearchCV, cross_val_score
36
37
38 from sklearn.metrics import mean_squared_error, r2_score
39
40 #OPTIMIZER
41
42
43 from keras.optimizers import RMSprop, Adam, Optimizer, Optimizer
44
45 #MODEL LAYERS
46
47
48 from tensorflow.keras.models import Sequential
49
50
51 from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D,
52 BatchNormalization, MaxPooling2D, BatchNormalization, \
53 Permute, TimeDistributed, Bidirectional, GRU, SimpleRNN, LSTM,
54 GlobalAveragePooling2D, SeparableConv2D
55
56
57 from keras import models
58
59
60 from keras import layers
```

```
1
2
3 import tensorflow as tf
4
5
6 from keras.applications import VGG16,VGG19,inception_v3
7
8 from keras import backend as K
9
10
11 from keras.utils import plot_model
12
13 #SKLEARN CLASSIFIER
14
15
16 from xgboost import XGBClassifier, XGBRegressor
17
18 from lightgbm import LGBMClassifier, LGBMRegressor
19
20
21 from catboost import CatBoostClassifier, CatBoostRegressor
22
23
24 from sklearn.linear_model import LogisticRegression
25
26 from sklearn.naive_bayes import GaussianNB
27
28
29 from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
30
31 from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
32
33
34 from sklearn.ensemble import BaggingRegressor
35
36
37 from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
38
39 from sklearn.neural_network import MLPClassifier, MLPRegressor
40
41 from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
42
43
44 from sklearn.linear_model import LinearRegression
45
46
47 from sklearn.cross_decomposition import PLSRegression
48
49
50 from sklearn.linear_model import Ridge
51
52 from sklearn.linear_model import RidgeCV
53
54
55 from sklearn.linear_model import Lasso
56
57 from sklearn.linear_model import LassoCV
58
59 from sklearn.linear_model import ElasticNet
60
```

```
1
2
3 from sklearn.linear_model import ElasticNetCV
4
5
6 #IGNORING WARNINGS
7
8 from warnings import filterwarnings
9
10
11 filterwarnings("ignore",category=DeprecationWarning)
12
13 filterwarnings("ignore", category=FutureWarning)
14
15 filterwarnings("ignore", category=UserWarning)
16
17
18
19
20
21 # PATH & LABEL PROCESS
22
23 MAIN PATH
24
25 Fire_Dataset_Path = Path("/content/drive/My Drive/Colab Notebooks/Fire Datasets")
26
27
28
29
30
31 PATH PROCESS
32
33 JPG_Path = list(Fire_Dataset_Path.glob(r"*/*.jpg"))
34
35
36
37
38
39 LABEL PROCESS
40
41 JPG_Labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1],JPG_Path))
42
43
44
45
46 print("FIRE: ", JPG_Labels.count("fire"))
47
48
49 print("NO_FIRE: ", JPG_Labels.count("nofire"))
50
51
52
53
54 # TRANSFORMATION TO SERIES STRUCTURE
55
56 JPG_Path_Series = pd.Series(JPG_Path,name="JPG").astype(str)
57
58
59 JPG_Labels_Series = pd.Series(JPG_Labels,name="CATEGORY")
60
```

```
1
2
3 print(JPG_Path_Series)
4
5
6 print(JPG_Labels_Series)
7
8 JPG_Labels_Series.replace({"non_fire_images":"NO_FIRE","fire_images":"FIRE"},inplace
9 =True)
10
11
12 print(JPG_Labels_Series)
13
14
15
16
17
18 # TRANSFORMATION TO DATAFRAME STRUCTURE
19
20 Main_Train_Data = pd.concat([JPG_Path_Series,JPG_Labels_Series],axis=1)
21
22 print(Main_Train_Data.head(-1))
23
24
25
26
27
28 SHUFFLING
29
30 Main_Train_Data = Main_Train_Data.sample(frac=1).reset_index(drop=True)
31
32 print(Main_Train_Data.head(-1))
33
34
35
36
37
38 print(Main_Train_Data["JPG"][12])
39
40 print(Main_Train_Data["CATEGORY"][12])
41
42
43 print(Main_Train_Data["JPG"][200])
44
45 print(Main_Train_Data["CATEGORY"][200])
46
47
48 print(Main_Train_Data["JPG"][45])
49
50 print(Main_Train_Data["CATEGORY"][45])
51
52 print(Main_Train_Data["JPG"][100])
53
54 print(Main_Train_Data["CATEGORY"][100])
55
56 print(Main_Train_Data["CATEGORY"][100])
57
58 print(Main_Train_Data.head(-1))
59
60
```

```
1
2
3
4
5
6 # VISUALIZATION
7
8
9 plt.style.use("dark_background")
10
11
12
13
14 GENERAL
15
16 Main_Train_Data['CATEGORY'].value_counts().plot.pie(figsize=(5,5))
17
18
19 plt.show()
20
21
22
23
24 IMAGES
25
26 figure = plt.figure(figsize=(10,10))
27
28
29 x = cv2.imread(Main_Train_Data["JPG"][0])
30
31
32 plt.imshow(x)
33
34
35 plt.xlabel(x.shape)
36
37 plt.title(Main_Train_Data["CATEGORY"][0])
38
39
40
41
42 figure = plt.figure(figsize=(10,10))
43
44
45 x = cv2.imread(Main_Train_Data["JPG"][237])
46
47
48 plt.imshow(x)
49
50
51 plt.xlabel(x.shape)
52
53 plt.title(Main_Train_Data["CATEGORY"][237])
54
55
56
57 figure = plt.figure(figsize=(10,10))
58
59
60 x = cv2.imread(Main_Train_Data["JPG"][20])
```

```
1
2
3 plt.imshow(x)
4
5
6 plt.xlabel(x.shape)
7
8 plt.title(Main_Train_Data["CATEGORY"][20])
9
10
11
12
13
14 figure = plt.figure(figsize=(10,10))
15
16 x = cv2.imread(Main_Train_Data["JPG"][48])
17
18 plt.imshow(x)
19
20
21 plt.xlabel(x.shape)
22
23
24 plt.title(Main_Train_Data["CATEGORY"][48])
25
26
27
28
29 fig, axes = plt.subplots(nrows=5,
30
31                          ncols=5,
32
33                          figsize=(10,10),
34
35                          subplot_kw={"xticks":[],"yticks":[]})
36
37
38
39
40
41 for i,ax in enumerate(axes.flat):
42
43     ax.imshow(cv2.imread(Main_Train_Data["JPG"][i]))
44
45     ax.set_title(Main_Train_Data["CATEGORY"][i])
46
47
48
49 plt.tight_layout()
50
51
52 plt.show()
53
54
55
56
57 fig, axes = plt.subplots(nrows=5,
58
59                          ncols=5,
```

```
1
2
3         figsize=(10,10),
4
5         subplot_kw={"xticks":[],"yticks":{}})
6
7
8
9
10
11 for i,ax in enumerate(axes.flat):
12
13     x = cv2.imread(Main_Train_Data["JPG"][i])
14
15     x = cv2.cvtColor(x,cv2.COLOR_RGB2BGR)
16
17     ax.imshow(x)
18
19     ax.set_title(Main_Train_Data["CATEGORY"][i])
20
21
22
23 plt.tight_layout()
24
25
26 plt.show()
27
28
29
30
31 # DETERMINATION TRAIN AND TEST DATA
32
33 IMAGE GENERATOR
34
35 Train_Generator = ImageDataGenerator(rescale=1./255,
36
37
38         shear_range=0.3,
39
40
41         zoom_range=0.2,
42
43
44         brightness_range=[0.2,0.9],
45
46
47         rotation_range=30,
48
49         horizontal_flip=True,
50
51
52         vertical_flip=True,
53
54
55         fill_mode="nearest",
56
57         validation_split=0.05)
58
59
60
```

```
1
2
3 Test_Generator = ImageDataGenerator(rescale=1./255)
4
5
6
7
```

```
8 SPLITTING TRAIN AND TEST
9
```

```
10 Train_Data,Test_Data =
```

```
11 train_test_split(Main_Train_Data,train_size=0.9,random_state=42,shuffle=True)
12
```

```
13
14
15 print("TRAIN SHAPE: ",Train_Data.shape)
16
```

```
17 print("TEST SHAPE: ",Test_Data.shape)
18
```

```
19
20 print(Train_Data.head(-1))
21
```

```
22
23 print("----"*20)
24
```

```
25 print(Test_Data.head(-1))
26
```

```
27
28 print(Test_Data["CATEGORY"].value_counts())
29
30
31
32
```

```
33 encode = LabelEncoder()
34
35
36
37
```

```
38 For_Prediction_Class = encode.fit_transform(Test_Data["CATEGORY"])
39
40
41
42
```

```
43 #How Generator Applied Image Look Like
44
45
46
47
```

```
48 example_Image = Train_Data["PNG"][99]
49
```

```
50
51 Load_Image = load_img(example_Image,target_size=(200,200))
52
```

```
53 Array_Image = img_to_array(Load_Image)
54
```

```
55
56 Array_Image = Array_Image.reshape((1,) + Array_Image.shape)
57
58
59
60
```

```
1
2
3     i = 0
4
5
6     for batch in Train_Generator.flow(Array_Image,batch_size=1):
7
8         plt.figure(i)
9
10
11         IMG = plt.imshow(array_to_img(batch[0]))
12
13         i += 1
14
15         if i % 4 == 0:
16
17             break
18
19
20
21 plt.show()
22
23
24
25
```

26 APPLYING GENERATOR AND TRANSFORMATION TO TENSOR

```
27
28
29 Train_IMG_Set = Train_Generator.flow_from_dataframe(dataframe=Train_Data,
30
31             x_col="JPG",
32
33             y_col="CATEGORY",
34
35             color_mode="rgb",
36
37             class_mode="categorical",
38
39             batch_size=32,
40
41             subset="training")
42
43
44
45
46
47
48
49
50 Validation_IMG_Set = Train_Generator.flow_from_dataframe(dataframe=Train_Data,
51
52             x_col="JPG",
53
54             y_col="CATEGORY",
55
56             color_mode="rgb",
57
58             class_mode="categorical",
59
60
```

```
1
2
3         batch_size=32,
4
5         subset="validation")
6
7
8
9
10
11 Test_IMG_Set = Test_Generator.flow_from_dataframe(dataframe=Test_Data,
12
13         x_col="JPG",
14
15         y_col="CATEGORY",
16
17         color_mode="rgb",
18
19         class_mode="categorical",
20
21         batch_size=32)
22
23
24
25
26
27
28
29 CHECKING
30
31 for data_batch,label_batch in Train_IMG_Set:
32
33     print("DATA SHAPE: ",data_batch.shape)
34
35     print("LABEL SHAPE: ",label_batch.shape)
36
37
38
39     break
40
41
42
43
44 from PIL import ImageFile
45
46
47
48
49 ImageFile.LOAD_TRUNCATED_IMAGES = True
50
51
52
53
54 from PIL import Image
55
56
57 from pathlib import Path
58
59
60
```

```
1
2
3 for file_path in Path('/content/drive/My Drive/Colab Notebooks/Fire
4 Datasets').glob('**/*.jpg'):
5
6
7     print('Check: %s' % file_path)
8
9
10
11
12     im = Image.open(file_path)
13
14
15     im.verify()
16
17
18
19
20     print('Done.')
```

```
21
22
23
24
25 for data_batch,label_batch in Test_IMG_Set:
26
27     print("DATA SHAPE: ",data_batch.shape)
28
29     print("LABEL SHAPE: ",label_batch.shape)
30
31     break
32
33
34
35
36
37
38 print("TRAIN: ")
39
40 print(Train_IMG_Set.class_indices)
41
42 print(Train_IMG_Set.classes[0:10])
43
44 print(Train_IMG_Set.image_shape)
45
46 print("---"*20)
47
48 print("VALIDATION: ")
49
50 print(Validation_IMG_Set.class_indices)
51
52 print(Validation_IMG_Set.classes[0:10])
53
54 print(Validation_IMG_Set.image_shape)
55
56
57
58
59
60
```

```
1
2
3 print("---"*20)
4
5
6 print("TEST: ")
7
8 print(Test_IMG_Set.class_indices)
9
10
11 print(Test_IMG_Set.classes[0:10])
12
13
14 print(Test_IMG_Set.image_shape)
15
16 # CNN-RCNN
17
18 # Define the model
19
20
21 Model_Three = Sequential()
22
23
24
25
26 # Convolutional layer 1
27
28 Model_Three.add(Conv2D(32, (3, 3), activation='relu'))
29
30 Model_Three.add(MaxPooling2D(pool_size=(2, 2)))
31
32
33
34
35
36 # Convolutional layer 2
37
38 Model_Three.add(Conv2D(64, (3, 3), activation='relu'))
39
40 Model_Three.add(MaxPooling2D(pool_size=(2, 2)))
41
42
43
44
45
46 # Convolutional layer 3
47
48 Model_Three.add(Conv2D(128, (3, 3), activation='relu'))
49
50 Model_Three.add(MaxPooling2D(pool_size=(2, 2)))
51
52
53
54
55
56 # Flatten the feature maps
57
58 Model_Three.add(Flatten())
59
60
```

```
1
2
3
4
5
6 # Fully connected layers
7
8 Model_Three.add(Dense(256, activation='relu'))
9
10
11 Model_Three.add(Dropout(0.2))
12
13
14 Model_Three.add(Dense(128, activation='relu'))
15
16
17 Model_Three.add(Dropout(0.2))
18
19 Model_Three.add(Dense(2, activation='softmax'))
20
21
22
23
24 # Compile the model
25
26 Model_Three.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
27
28
29
30
31 #Defining early stopping to prevent overfitting
32
33
34 early_stopping = tf.keras.callbacks.EarlyStopping(
35
36     monitor = 'val_loss',
37
38     mode = 'auto',
39
40     min_delta = 0,
41
42     patience = 2,
43
44     restore_best_weights = True
45
46
47
48
49 )
50
51
52
53
54 RCNN_Model = Model_Three.fit(Train_IMG_Set,
55
56     validation_data=Validation_IMG_Set,
57
58     callbacks= [early_stopping],
59
60
```

```
1
2
3         epochs=100, verbose=2)
4
5
6     print(Model_Three.summary())
7
8
9
10
11     plot_model(Model_Three,to_file="Model_Three.jpg",show_layer_names=True,show_dtype=
12     True,show_shapes=True)
13
14
15
16
17     Model_Results_Three = Model_Three.evaluate(Test_IMG_Set)
18
19
20     print("LOSS: " + "%.4f" % Model_Results_Three[0])
21
22
23     print("ACCURACY: " + "%.2f" % Model_Results_Three[1])
24
25
26
27
28     plt.plot(RCNN_Model.history["accuracy"])
29
30     plt.plot(RCNN_Model.history["val_accuracy"])
31
32
33     plt.ylabel("ACCURACY")
34
35
36     plt.xlabel("EPOCH")
37
38
39     plt.legend()
40
41     plt.show()
42
43
44
45
46     plt.plot(RCNN_Model.history["loss"])
47
48     plt.plot(RCNN_Model.history["val_loss"])
49
50
51     plt.ylabel("LOSS")
52
53
54     plt.legend()
55
56     plt.show()
57
58
59
60
```

```
1
2
3 plt.plot(RCNN_Model.history["val_accuracy"])
4
5
6 plt.plot(RCNN_Model.history["val_loss"])
7
8 plt.ylabel("ACCURACY-LOSS")
9
10
11 plt.legend()
12
13
14 plt.show()
15
16
17
18 Dict_Summary_Three = pd.DataFrame(RCNN_Model.history)
19
20
21 Dict_Summary_Three.plot()
22
23
24
25
26 Prediction_Three = Model_Three.predict(Test_IMG_Set)
27
28
29 Prediction_Three = Prediction_Three.argmax(axis=-1)
30
31
32
33
34 print(Prediction_Three)
35
36
37
38
39 predictions = Model_Three.predict(Test_IMG_Set)
40
41
42 Prediction_Class_Three = predictions.argmax(axis=-1)
43
44
45
46
47 print(Prediction_Class_Three)
48
49
50
51
52 fig, axes = plt.subplots(nrows=6,
53
54                          ncols=4,
55
56                          figsize=(20, 20),
57
58                          subplot_kw={'xticks': [], 'yticks': []})
```

```
1
2
3
4
5
6     for i, ax in enumerate(axes.flat):
7
8         ax.imshow(cv2.imread(Test_Data["JPG"].iloc[i]))
9
10        ax.set_title(f"TEST: {Test_Data.CATEGORY.iloc[i]}\n
11        PREDICTION: {Prediction_Three[i]}")
12
13
14
15    plt.tight_layout()
16
17
18    plt.show()
19
20
21
22
23    print(classification_report(For_Prediction_Class,Prediction_Three))
24
25
26
27
28    print(confusion_matrix(For_Prediction_Class,Prediction_Three))
29
30
31
32
33    PREDICTION FOR DIFFERENT SOURCE
34
35    * FIRE
36
37
38    image_path = Path("/content/drive/My Drive/Colab Notebooks/Fire Datasets/fire2.jpg")
39
40
41    img = load_img(image_path,target_size=(256,256))
42
43
44    x = img_to_array(img)
45
46    x = np.expand_dims(x,axis=0)
47
48
49
50
51    Diff_Pred = Model_Three.predict(x)
52
53
54    Diff_Pred = Diff_Pred.argmax(axis=-1)
55
56    print(Diff_Pred)
57
58
59
60
```

1
2
3 * NOT FIRE
4
5
6
7

8 image_path_Two = Path("/content/drive/My Drive/Colab Notebooks/Fire
9 Datasets/nofire3.jpg")
10
11

12 img_Two = load_img(image_path_Two, target_size=(256,256))
13
14

15 x_Two = img_to_array(img_Two)
16
17

18 x_Two = np.expand_dims(x_Two, axis=0)
19
20
21

22
23 Diff_Pred_Three = Model_Three.predict(x)
24
25

26 Diff_Pred_Three = Diff_Pred_Three.argmax(axis=-1)
27
28

29 print(Diff_Pred_Three)
30
31

Appendix 7 Glossary

- 32 1. **Affine Transformation:** A data augmentation technique used to geometrically
33 transform images while preserving points, straight lines, and planes.
- 34 2. **AI-powered surveillance:** Monitoring systems enhanced by artificial intelligence to
35 detect anomalies, track individuals, or analyse behaviour in real time, often used in
36 security and law enforcement contexts.
- 37 3. **API:** Application Programming Interface. A set of protocols and tools that allows
38 different software applications to communicate and interact with each other.
- 39 4. **ATT Squeeze:** Refers to a specific attention-based feature compression or refinement
40 technique (typically part of neural network architectures), where attention mechanisms
41 are used to highlight important features while reducing dimensionality.
- 42 5. **Augmentation:** A method of increasing the diversity of training data by applying
43 transformations such as rotation, flipping, cropping, and stretching to existing images.
- 44 6. **Binary Crossentropy:** A loss function used in binary classification tasks, measuring
45 the difference between predicted probabilities and actual class labels.
- 46 7. **CCTV:** Closed-Circuit Television. A system used for surveillance, where video
47 cameras transmit signals to a specific set of monitors for monitoring and security
48 purposes.
- 49 8. **Classification Report:** A detailed summary that includes precision, recall, F1-score,
50 and support for evaluating classification model performance.
- 51 9. **CO: Carbon monoxide.** A colourless, odourless, and toxic gas produced by burning
52 fossil fuels. In high concentrations, it can be harmful or fatal, necessitating careful
53 monitoring in enclosed or industrial environments.
54
55
56
57
58
59
60

10. **Convolutional Layer:** A type of neural network layer used to detect features in input images by applying a set of filters.
11. **Convolutional Neural Network (CNN):** A class of deep neural networks commonly used for analysing visual imagery. CNNs are particularly effective in image recognition and classification tasks due to their ability to automatically and adaptively learn spatial hierarchies of features.
12. **Crowd-sourced annotations:** Labels or tags for datasets collected from a large group of non-expert contributors, often via online platforms. This approach is cost-effective and enables rapid data labelling at scale.
13. **Early Stopping:** A regularization technique that halts training when the model's performance on a validation set ceases to improve, to prevent overfitting.
14. **Epoch:** One complete cycle through the training dataset during the training process of a neural network.
15. **False Negative (FN):** A type of prediction error where the model incorrectly predicts a negative result for a positive instance.
16. **False Positive (FP):** A type of prediction error where the model incorrectly predicts a positive result for a negative instance.
17. **Flattening:** The process of converting multi-dimensional feature maps into a one-dimensional vector before feeding into fully connected layers.
18. **Fully Connected Layer:** A layer where each neuron is connected to all neurons in the previous layer, typically used for classification in neural networks.
19. **Google Colab:** Google Colaboratory. A cloud-based platform provided by Google that allows users to write and execute Python code in a Jupyter notebook environment, particularly suited for machine learning and data analysis.
20. **IoT-based fuzzy models:** Intelligent systems that use fuzzy logic within Internet of Things (IoT) environments to manage uncertainty and approximate reasoning, often applied in automation.
21. **Max Pooling:** A downsampling technique used to reduce the spatial dimensions of feature maps while retaining important features.
22. **Min_delta:** A parameter in early stopping that defines the minimum change in validation loss to qualify as an improvement.
23. **Overfitting:** A modelling error in machine learning where a model learns noise or details in the training data too well, reducing its ability to generalize to new data.
24. **Patience Parameter:** A setting in early stopping that specifies how many epochs with no improvement to wait before stopping training.
25. **Precision:** The ratio of correctly predicted positive observations to total predicted positives.
26. **RCNNs:** Region-based Convolutional Neural Networks. These are a type of CNN designed for object detection, which identify regions of interest in an image and then classify the contents of those regions.
27. **Recall:** The ratio of correctly predicted positive observations to all actual positives.

- 1
 - 2
 - 3
 - 4
 - 5
 - 6
 - 7
 - 8
 - 9
 - 10
 - 11
 - 12
 - 13
 - 14
 - 15
 - 16
 - 17
 - 18
 - 19
 - 20
 - 21
 - 22
 - 23
 - 24
 - 25
 - 26
 - 27
 - 28
 - 29
 - 30
 - 31
 - 32
 - 33
 - 34
 - 35
 - 36
 - 37
 - 38
 - 39
 - 40
 - 41
 - 42
 - 43
 - 44
 - 45
 - 46
 - 47
 - 48
 - 49
 - 50
 - 51
 - 52
 - 53
 - 54
 - 55
 - 56
 - 57
 - 58
 - 59
 - 60
28. **ReLU:** Rectified Linear Unit. An activation function used in neural networks that outputs the input directly if it is positive; otherwise, it outputs zero. It is widely used due to its simplicity and efficiency in training deep neural networks.
29. **Remote sensing:** The process of collecting data about an object or area from a distance, typically using satellites, drones, or aircraft. Commonly used in environmental monitoring, agriculture, and mapping.
30. **Sensor-integrated AI systems:** AI models that directly interact with and process data from physical sensors (e.g., cameras, temperature, motion sensors) to make real-time decisions or analyses.
31. **SoftMax Activation:** A function used in the output layer of a classification neural network to normalize output into a probability distribution over predicted output class.
32. **Synthetic data:** Artificially generated data that mimics real-world data, used to train AI models when real data is scarce, expensive, or sensitive. It helps improve model performance and generalization.
33. **Test Dataset:** A subset of the dataset used exclusively to evaluate the final model's performance.
34. **Training Dataset:** The portion of the dataset used to train the model by adjusting weights based on loss calculations.
35. **Transfer Learning:** A technique in machine learning where a model developed for one task is reused as the starting point for a model on a second, related task. It saves time and computational resources.
36. **U-Net:** A convolutional neural network architecture commonly used for image segmentation, especially in medical imaging. It features a “U”-shaped structure with symmetrical encoding and decoding paths.
37. **Validation Dataset:** A subset of the training data used to evaluate model performance during training and tune hyperparameters.
38. **YOLO: You Only Look Once.** A real-time object detection algorithm that processes an entire image in a single pass through a neural network, enabling fast and accurate identification of objects.