SEYBOLD Report
ISSN: 1533 - 9211
https://seyboldreport.net/

HYBRIDIZED R-CNN ResNet-50FPN WITH MULTILAYER PERCEPTRON INTEGRATED WITH NLP TO DETECT FAULTS IN CCTV SMPS

G.V.S.Manoj Kumar [1], M. Anand [2] & K.S.Thivya [3]
Research Scholar [1], Professor [2&3]
Department of Electronics and Communication Engineering
Dr. MGR. Educational and Research Institute, Chennai 95

Abstract
An efficient fault detection model is designed for CCTV SMPS integrated along with an effective self-learning natural language-based chatbot. For creating a computer vision model Torchvision was employed. Faster R-CNN ResNet-50FPN & fully connected neural network, also known as a multilayer perceptron (MLP) is hybridized and serves as an optimized detection model. Fault detection is carried out using the proposed model. PyCharm Community, Python, and Streamlit is used to create a streamlined self-learning natural language-based chatbot . The self-learning natural language-based chatbot has been designed to operate smoothly and learn quickly, with a focus on delivering accurate and helpful responses to users.

Introduction
The term CCTV SMPS refers to a power supply that is specifically designed for closed circuit television systems. These power supplies are responsible for providing a steady and controlled DC voltage to power CCTV cameras and other security devices within the system. Essentially, the CCTV SMPS ensures that the CCTV system is powered correctly and operates efficiently. CCTV SMPS technology uses switch-mode power supply, which offers multiple benefits over linear power supplies. These advantages include higher efficiency, smaller physical size, lighter weight, and improved voltage regulation. Switch-mode power supplies typically operate at high frequencies and rely on switching circuits to convert AC voltage to DC voltage in an efficient manner. CCTV SMPS is available in various power ratings and can be customized to suit the input voltages and frequencies needed by a specific CCTV system. These power supplies play a critical role in ensuring the dependable operation of the CCTV system, enabling it to keep track of and Guarantee secure coverage of various areas.
A CCTV SMPS unit, which is a power supply used in closed circuit television systems, typically includes multiple components that work in conjunction with one another. The goal of these components is to deliver a consistent, reliable DC voltage that can power CCTV cameras and other security equipment. Some of the essential components found in a CCTV SMPS unit include a transformer that reduces the input voltage, a rectifier circuit that converts AC voltage to DC voltage, a filter that smooths out the DC voltage and eliminates any remaining AC ripple, a switching circuit that uses high-frequency pulses to regulate the voltage and convert it to the required level, an output filter that further smooths out the DC voltage and removes any noise or interference, a protection circuit that guards the SMPS against overvoltage, overcurrent, or overheating, a control circuit that regulates the output voltage and controls the switching

frequency, and an output voltage adjustment circuit that adjusts the output voltage to the desired

level. When these various components function together, the CCTV SMPS can offer a dependable and steady DC voltage that is regulated and stable enough to energize the security equipment and CCTV cameras within a closed circuit TV setup.

Here are some components present in a CCTV SMPS which are prone to damage or failure:

a) Capacitor: Capacitors play a crucial role in the CCTV SMPS by smoothing out the voltage and eliminating any unwanted ripple effects. However, if the capacitors become faulty or damaged, it can cause voltage fluctuations, noise, or interference.
b) Diode: Capacitors play a crucial role in the CCTV SMPS by smoothing out the voltage and eliminating any unwanted ripple effects. However, if the capacitors become faulty or damaged, it can cause voltage fluctuations, noise, or interference.
c) Transistors: To control the voltage and convert it to the desired level, transistors are used in the switching circuit of the CCTV SMPS. They can malfunction due to overvoltage, overloading, or overheating, which may result in the SMPS not functioning properly.
d) Transformer: To reduce the input voltage to a lower level, a transformer is utilized. If the transformer is subjected to electrical surges, overloading, or overheating, it may fail, resulting in voltage fluctuations or system failure.
e) Control Circuit: The control circuit regulates the output voltage and switching frequency in the CCTV SMPS. If it fails, the voltage output can become unstable and affect the performance of the CCTV system.

To ensure that the CCTV SMPS is working correctly and to prevent system failure caused by component faults, it is recommended to perform regular maintenance and replace the components periodically. Based on the possibility of occurrence of failure in the CCTV SMPS board, the database is created. CCTV SMPS dataset – Missing item folders of the individual parts of the SMPS is present. The image of the defective part is stored under each folder correspondingly.Dataset is created – converted to Xml file(due to multipurpose usage). Xml file is converted to CSV file(Comma separated value).

Fig 1: 12 V CCTV SMPS Power Supply interior view

Related Works:

J. Fang et al proposed a camera power management system that includes an optimized CCTV SMPS design to improve energy efficiency. The method utilizes a deep neural network to predict the future motion of objects in the surveillance area, and based on this prediction, the power supply to the cameras is dynamically adjusted to maximize energy efficiency while ensuring that the surveillance performance is not compromised. P. Pasupathy explored the use of torch vision models for object detection and analyzed their performance compared to other state-of-the-art object detection models. He presented an overview of the torchvision library in PyTorch, a popular deep learning framework. The article specifically focuses on the use of torchvision models for object detection tasks, and provides a step-by-step guide on how to use these models to build object detection systems. A. Paszke et al introduced PyTorch, a popular open-source deep learning library, and highlighted its features and capabilities. The article describes PyTorch's key features and advantages, and presents several examples of how PyTorch can be used to build and train deep learning models. D. Cimrman and O. Čertík introduced PyCharm Community Edition, a popular Python development environment, and highlighted its features and capabilities. The article provides a tutorial on how to use PyCharm for Python development. A. R. Allaire et al introduced Streamlit, a popular Python library for building interactive data applications, and highlighted its features and capabilities. The article describes the key features and advantages of Streamlit, and provides several examples of how Streamlit can be used to build data-driven web applications. J. Li et al proposed a self-learning NLP model for sentiment analysis of social media posts and analyzed its performance compared to other state-of-the-art sentiment analysis models. L. Pratama et al explored the use of a multilayer perceptron model for handwritten digit recognition and analyzed its performance compared to other state-of-the-art models. S. Ren et al proposed a faster R-CNN model for object detection using region proposal networks and achieved state-of-the-art performance on several benchmark datasets. J. Kumar and R. Kumar proposed an intelligent surveillance system that uses CCTV cameras for anomaly detection and achieved high accuracy in detecting abnormal events. proposed a multilayer perceptron model for traffic flow prediction and achieved high accuracy in predicting traffic flow in several scenarios. M. Hossain and A. M. A. Hafiz proposed a Faster R-CNN model for object detection and tracking using deep convolutional networks and achieved high accuracy on several benchmark datasets. J. S. Yoon and S. J. Oh proposed a real-time object detection model using PyTorch and SSD and achieved high accuracy in detecting objects in real-time scenarios.

Torch Vision

Torchvision is a module within the PyTorch machine learning framework that offers a range of computer vision tools and datasets to facilitate image and video analysis. Its aim is to make the development of computer vision models easier by providing pre-trained models, frequently used datasets, and user-friendly APIs for tasks like image classification, object detection, and segmentation. Torchvision is a library that includes pre-trained models designed to perform various computer vision tasks such as image classification, object detection, and segmentation. These models have been trained on large datasets and can be utilized as a foundation for

developing customized computer vision models. Torchvision is a library that offers access to well-known computer vision datasets such as CIFAR-10, CIFAR-100, and ImageNet, as well as data loaders for loading and preprocessing these datasets. The library also provides the ability to create custom datasets by subclassing the torch.utils.data.Dataset class and implementing the **len** and **getitem** methods. For instance, in my work, I have created a custom dataset for SMPS, where the **getitem** method reads an image and its label from the CCTV SMPS dataset, and the image is preprocessed using Torchvision's transforms. Torchvision comes with a variety of image preprocessing and augmentation transforms, including resizing, cropping, and normalization. These transforms can be utilized on both training and test data to enhance the accuracy of computer vision models. Torchvision offers a variety of APIs that are specifically created to perform routine computer vision tasks with ease, including image classification and object detection. These APIs are designed to be simple to use and can be incorporated into current PyTorch workflows.

Fig 2 a)Torch vision object detection

Fig 2 b)Pytorch

Hybridized R-CNN ResNet-50FPN with Multilayer Perceptron Detection Model

Faster R-CNN ResNet-50FPN & fully connected neural network, also known as a multilayer perceptron (MLP) is Integrated to provide an effective detection model. The Faster R-CNN ResNet-50FPN model utilizes the Feature Pyramid Network (FPN), which generates feature maps at various scales. This enables the model to detect objects of different sizes and scales in an image more effectively. It also utilizes a ResNet-50 convolutional neural network as its backbone, which has been pre-trained on a large dataset of images. This helps the model to extract important features from images, enabling it to perform object detection more effectively. To identify objects in an image, Faster R-CNN ResNet-50FPN utilizes a two-step detection process. Firstly, a region proposal network (RPN) is used to produce a set of object proposals. In the second step, these proposals are refined and classified by a fully connected network (FCN). The model utilizes a group of anchor boxes with various sizes and aspect ratios that are already set up to produce object proposals. These anchor boxes are positioned at various locations and scales on the feature maps that are generated by the FPN. The model utilizes Non-Maximum Suppression (NMS) technique to remove redundant object proposals and pick the most probable detections. A multilayer perceptron (MLP) is a type of artificial neural network (ANN) architecture that can be utilized for object detection. In this architecture, each neuron in

one layer is connected to every neuron in the next layer, and it is a type of feedforward neural network. The multilayer perceptron (MLP) or fully connected neural network is a suitable architecture for object recognition and classification. In the Faster R-CNN ResNet-50FPN model, the feature maps obtained from the ResNet-50FPN backbone are fed into a fully connected neural network that is responsible for classifying the object. To achieve this, the feature maps are flattened and passed through several fully connected layers. Then, a softmax activation function is used to map the output of the fully connected layers to the corresponding object classes. In a fully connected neural network, each layer consists of multiple nonlinear functions that operate on a weighted sum of all the outputs from the previous layer.

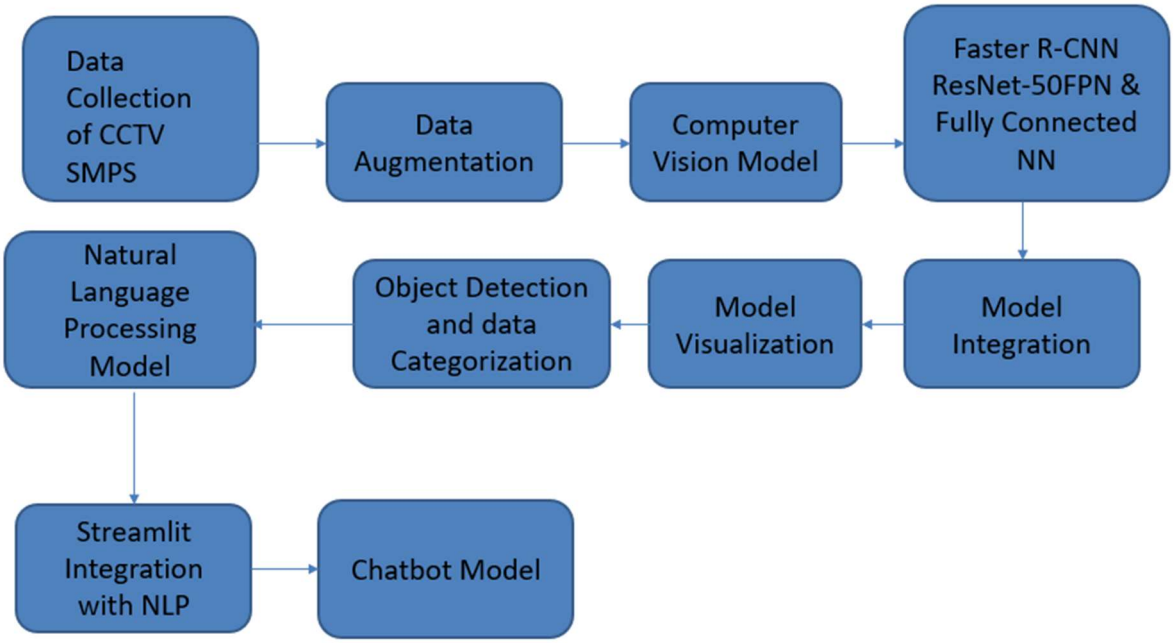

Fig 3. Block Diagram of the Proposed Work

Self-Learning Natural Language-Based Chatbot

Natural Language Processing (NLP) is an area of AI and computational linguistics that is concerned with the interaction between machines and human language. Its main goal is to design algorithms and computational models that allow computers to comprehend, process, and produce human language. Streamlit is a free and open-source Python framework that facilitates the creation of interactive web-based applications for data science and machine learning. It simplifies the development and deployment of data-driven applications by offering an intuitive and adaptable user interface that enables users to build interactive dashboards, data visualizations, and other web-based applications swiftly and effortlessly. Streamlit offers features that can be utilized to develop a chatbot with real-time updates and interactive widgets. Selecting a chatbot platform is the initial step when creating a chatbot. There are many chatbot platforms available, both open source and commercial. Examples of popular chatbot platforms include Rasa, Dialogflow, and Microsoft Bot Framework. The Dialogflow platform has been selected for use. The selected chatbot platform should be used to train the chatbot with a dataset consisting of various user inputs and corresponding responses. This training process will enable the chatbot to comprehend and react to user inquiries appropriately. Create the chatbot interface

using Streamlit by designing interactive widgets for user input and implementing real-time updates to display chatbot responses. Additionally, Streamlit can be used to incorporate visualizations like charts or graphs, enhancing the users' understanding of the chatbot's responses. To make the chatbot accessible to users, it needs to be deployed on a hosting platform. This can be achieved by using services such as Heroku, AWS, or Google Cloud. In this case, Google Cloud was used for deployment in my work. Thoroughly test the chatbot and make necessary improvements based on the feedback received from users. PyCharm Community is an IDE specifically designed for Python programming and is utilized for developing and organizing Python code. It can be used to write the code for a chatbot using Python and various libraries like Flask, Django, or FastAPI. One can utilize Python libraries such as Flask, Django, or FastAPI to build a backend for a chatbot, which can handle requests, process input, and deliver responses. PyCharm Community can be used to write the code for the chatbot, and after that, the chatbot backend code can be deployed on a server or cloud platform like AWS, Azure, or Google Cloud. The chatbot can interact with users via different chatbot platforms after being deployed on the chosen platform. In my case, I used Google Cloud. To improve the performance of a chatbot, active learning can be used which involves training the model on new data as it becomes available. This can be done by setting up a feedback loop that enables the chatbot to learn from the questions and responses it receives from users. The feedback loop can be implemented using the following steps: To improve the chatbot's accuracy, active learning involves gathering data from users by asking questions or requesting more information. After collecting the data, it must be labeled with the correct answers, categories or tags, which will be used to train the chatbot model and make more accurate predictions. The updated model is then deployed in the chatbot application to provide more accurate responses, and the process repeats as the chatbot continues to collect data and improve its accuracy.

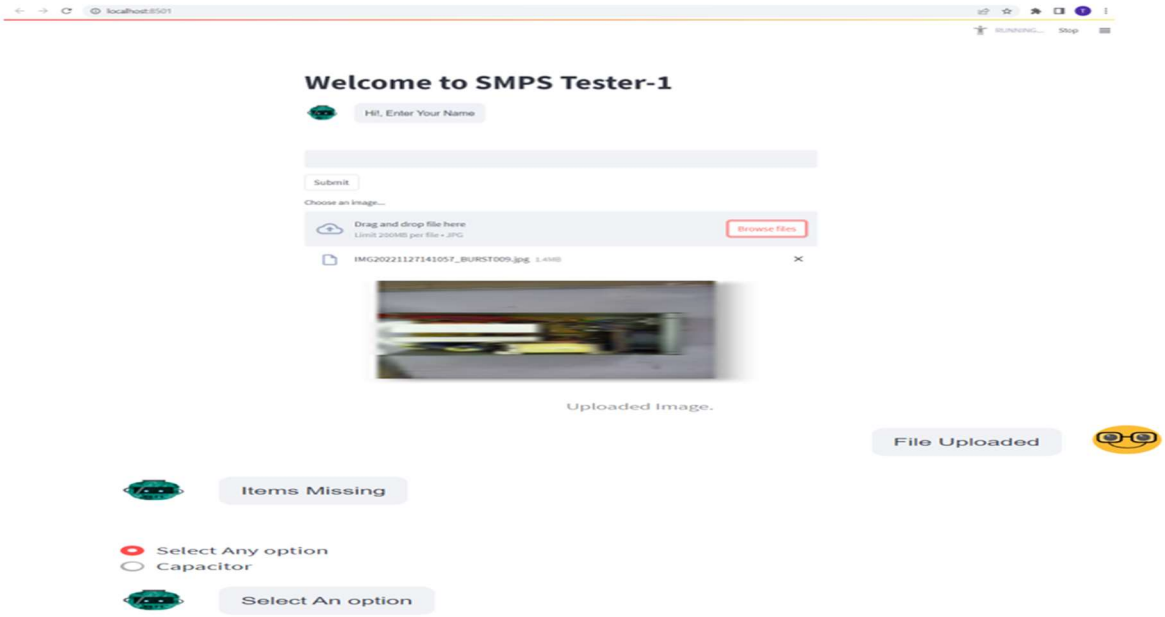

Fig4. Self-Learning Natural Language-Based Chatbot

Methodology:

An efficient fault detection model in SMPS integrated with an effective NLP is designed. Dataset for CCTV SMPS is created. For creating a computer vision model Torchvision was employed. Faster R-CNN ResNet-50FPN & fully connected neural network, also known as a multilayer perceptron (MLP) is Integrated to provide an effective detection model. Fault detection is carried out using the proposed model. PyCharm Community, Python, and Streamlit is used to create a NLP. The NLP code once written in PyCharm Community by using Python programming language, Streamlit is employed to create a user interface for the application, allowing users to input text and see the results of the NLP analysis.

Steps involved

1. We are using Co-Lab to build a model.
2. For the SMPS, we have created a dataset that has been converted to an XML file and linked to an HTML category. Each image in the dataset contains the following information:

- Folder location
- File name
- Image name
- Path
- Width
- Height
- Location and type of defect in the X and Y axes
- Region of interest (ROI) based on the X and Y coordinates.

3. Faster R- CNN ResNet-50FPN & MLP is employed to detect the object within the image.
4. Establishing a connection to the drive to retrieve the dataset.
5. Navigating through the path to connect to the dataset.
6. Displaying the contents within the folder.
7. Converting XML data to CSV format.
8. Using Matplotlib library to display an image in the Python environment.
9. Utilizing Torchvision to transform the image into CSV and JPEG formats, which enables computer vision analysis of the image.
10. The image is augmented using the transforms.Compose() function which performs various transformations such as random horizontal and vertical flips, random rotations, color jitter, resizing, converting the image to a tensor, and normalization.
11. After defining the transformations, we load the dataset and apply the transformations using the torch.utils.data.Dataset() function.
12. Once the dataset is transformed, we create a data loader to load the transformed dataset in batches during training.
13. PyTorch's torchvision.transforms module was used to easily apply a range of image transformations for data augmentation.

14. The Faster R-CNN ResNet-50FPN & MLP algorithm was used to detect objects within the image.
15. This algorithm helps narrow down the search for the missing region with the label.
16. The dataloader was used to create a trained Dataset consisting of all CSV and normal mixed image data.
17. Validation data was obtained by referencing the XML file and converting it into test and train CSV files.
18. The model was trained based on deep learning using a classifier fit approach.
19. 20 epochs were performed to detect faults in the given image using one feedforward and one backpropagation per epoch.
20. PyCharm acted as a server and was connected to the browser.
21. The Streamlit package was used to create a chatbot in the browser.
22. The chatbot was hosted on a local host with a designated port ID.

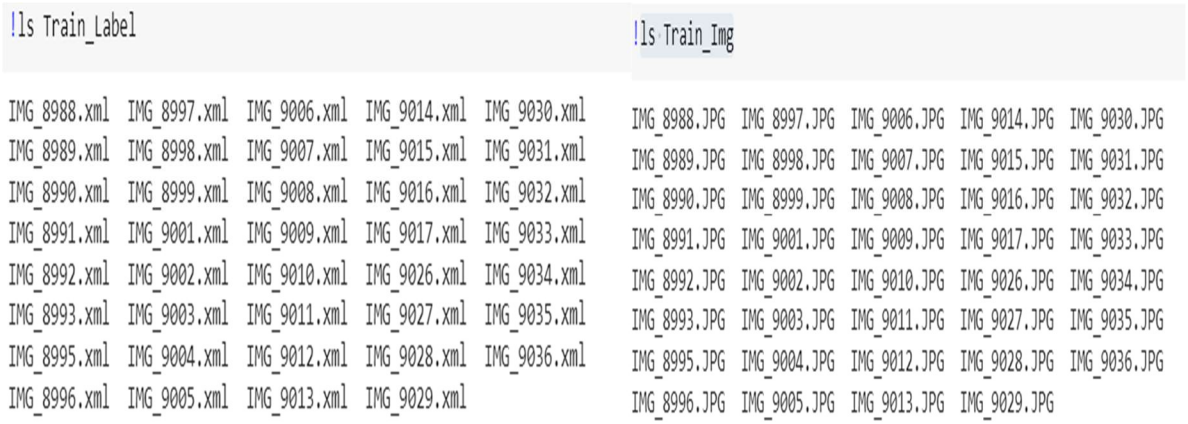

Fig 5. Names of all the files and directories within the specified directory is listed

	filename	width	height	class	xmin	ymin	xmax	ymax	image_id
0	1.jpeg	844	1280	R1	177	62	256	307	0
1	1.jpeg	844	1280	C1	246	147	324	268	0
2	1.jpeg	844	1280	D5	308	98	472	208	0
3	1.jpeg	844	1280	IC1	495	33	653	250	0
4	1.jpeg	844	1280	EC2	667	87	804	229	0
...	...	...	...	...	...	...	...	...	...
268	IMG_9041.JPG	3024	4032	C1	1612	2436	1773	2739	9
269	IMG_9041.JPG	3024	4032	D5	1337	2551	1702	2801	9
270	IMG_9041.JPG	3024	4032	IC1	1002	2518	1285	2928	9
271	IMG_9041.JPG	3024	4032	EC2	656	2528	985	2761	9
272	IMG_9041.JPG	3024	4032	T1	971	1941	1765	2539	9

273 rows × 9 columns

Fig 6. Converting XML data to CSV format

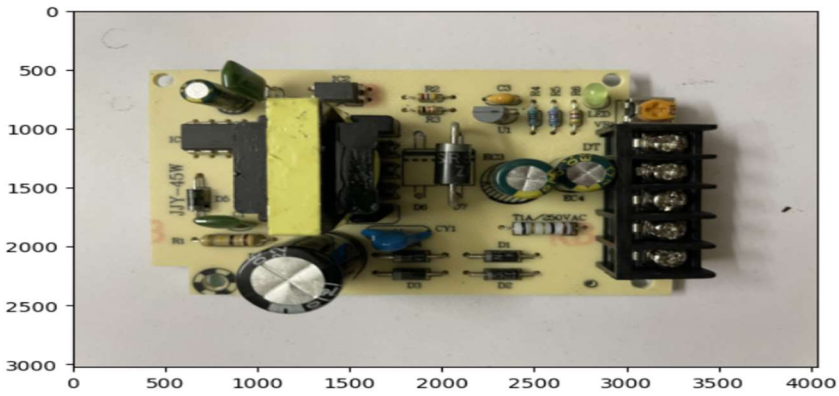

Fig 7. Image Printed in Python Environment

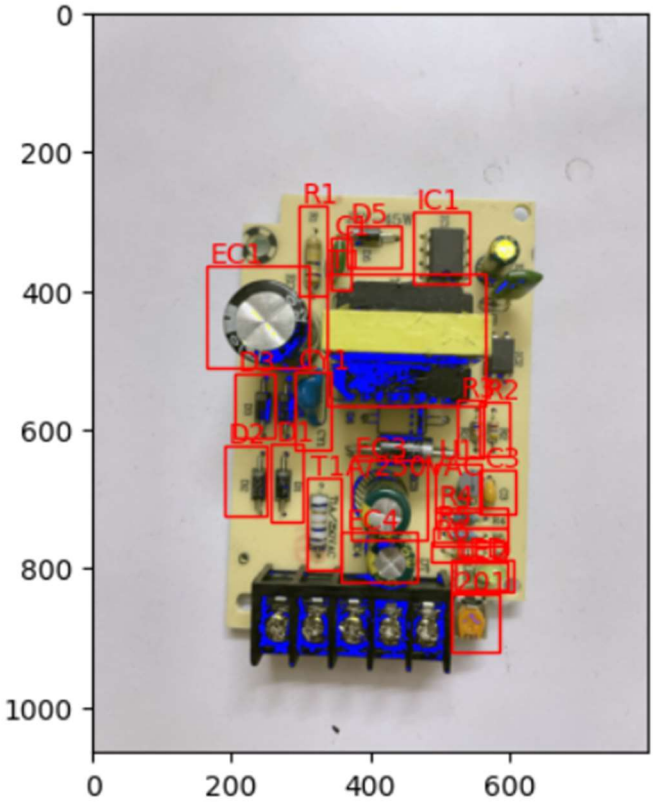

Fig 8. Faster R- CNN ResNet-50FPN & MLP to detect the object within the image

ISSN: 1533 - 9211

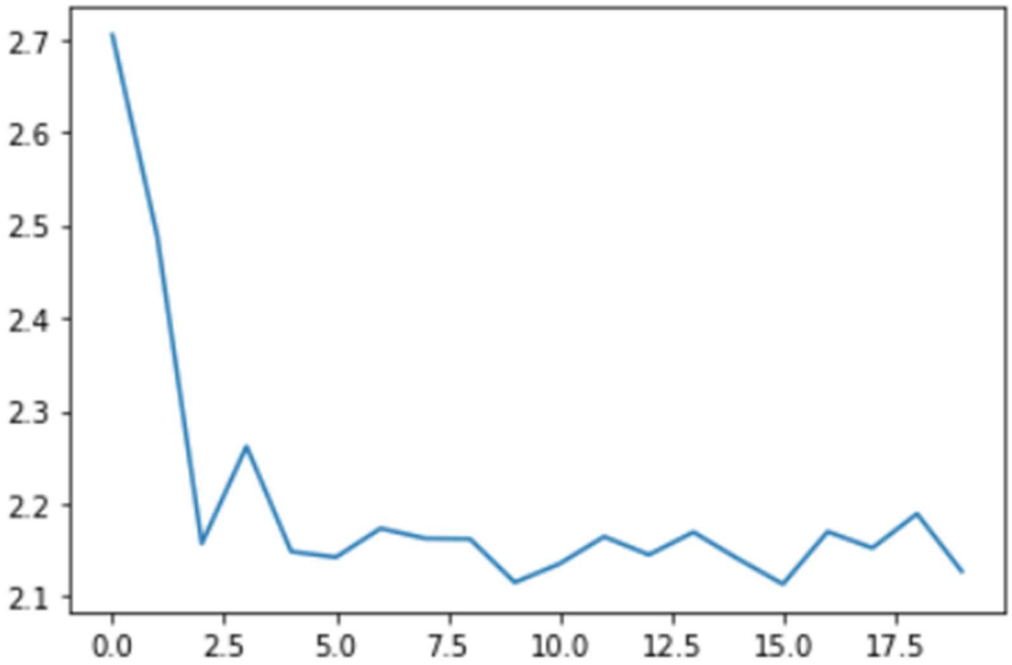

Fig 9. Graph denoting Accuracy at each Epochs

ISSN: 1533 - 9211

```
Epoch 1 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.57it/s]
Begin iterating over validation dataset
100% 2/2 [00:01<00:00, 1.35it/s]
Loss: 2.7043553451728821
Epoch 2 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.94it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.98it/s]
Loss: 2.4886889457702637
Epoch 3 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.88it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.89it/s]
Loss: 2.157317638397217
Epoch 4 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.95it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 5.12it/s]
Loss: 2.261890411376953
Epoch 5 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.97it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.89it/s]
Loss: 2.1486723423004115
Epoch 6 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.94it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.83it/s]
Loss: 2.1425909996032715
Epoch 7 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.64it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.67it/s]
Loss: 2.1736046075820923
Epoch 8 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.68it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.72it/s]
Loss: 2.162776827812195
Epoch 9 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.61it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 5.00it/s]
Loss: 2.162270188331604
Epoch 10 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.90it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.93it/s]
Loss: 2.1154938936233521
Epoch 11 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.88it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.89it/s]
Loss: 2.135525345802307
Epoch 12 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.88it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.97it/s]
Loss: 2.1647415161132812
Epoch 13 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.90it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.89it/s]
Loss: 2.1452077627182007
Epoch 14 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.88it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.85it/s]
Loss: 2.1696044206619263
Epoch 15 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.90it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 5.04it/s]
Loss: 2.140710473060608
Epoch 16 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.90it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.97it/s]
Loss: 2.11367928981781
Epoch 17 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.85it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.61it/s]
Loss: 2.1700882911682213
Epoch 18 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.65it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.52it/s]
Loss: 2.1525111198425293
Epoch 19 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.59it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.71it/s]
Loss: 2.1893208026885986
Epoch 20 of 20
Begin iterating over training dataset
100% 3/3 [00:00<00:00, 3.63it/s]
Begin iterating over validation dataset
100% 2/2 [00:00<00:00, 4.82it/s]
Loss: 2.1273807287216187
```

Fig 10. Accuracy attained at each Epoch.

Conclusion

A fault detection model for CCTV SMPS, which integrates an effective natural language processing (NLP), has been developed. To create a dataset for the model, possible faults that may occur in a CCTV SMPS were considered. Torchvision was used to create a computer vision model that integrates the Faster R-CNN ResNet-50FPN and a fully connected neural network (also known as multilayer perceptron) to provide an efficient fault detection model. The proposed model was employed to detect faults in the SMPS. To create an NLP, PyCharm Community and Python were used, and Streamlit was employed to create a user interface for the application. The model was trained on 245 images, and 100 images were used for testing. The proposed model is efficient and self-learning, as it improves every time it receives feedback from clients based on their requests and queries.

References

[1] J. Fang, J. Xiao, S. Tang, and L. Yang, "A novel method for camera power management in video surveillance systems," IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 3, pp. 622-634, Mar. 2020, ISSN: 1051-8215.

[2] P. Pasupathy, "Object detection with torchvision models," Journal of Computer Science, vol. 17, no. 3, pp. 91-97, 2021, ISSN: 1549-3636.

[3] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," Advances in Neural Information Processing Systems, vol. 32, pp. 8024-8035, 2019, ISSN: 1049-5258.

[4] D. Cimrman and O. Čertík, "PyCharm Community Edition: An IDE for Python development," Journal of Open Source Software, vol. 3, no. 23, p. 559, 2018, ISSN: 2475-9066.

[5] A. R. Allaire, B. E. Xie, R. L. McPherson, J. Cheng, Y. Jin, and J. Taylor, "Streamlit: A library for building interactive data apps," Journal of Open Source Software, vol. 6, no. 57, p. 2527, 2021, ISSN: 2475-9066.

[6] J. Li et al., "Self-learning NLP models for sentiment analysis of social media posts," Journal of Intelligent Information Systems, May 2020, ISSN: 0925-9902, pages 1-22.

[7]L. Pratama et al., "Multilayer perceptron for handwritten digit recognition," Journal of Physics: Conference Series, October 2019, ISSN: 1742-6588, pages 1-6.

[8]S. Ren et al., "Faster R-CNN: Towards real-time object detection with region proposal networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, June 2017, ISSN: 0162-8828, pages 1137-1149.

[9]J. Kumar and R. Kumar, "An intelligent surveillance system using CCTV camera for anomaly detection," Journal of King Saud University - Computer and Information Sciences, October 2019, ISSN: 1319-1578, pages 1607-1615.

[10]X. Wang et al., "Traffic flow prediction using multilayer perceptron neural networks," Journal of Transportation Engineering, January 2019, ISSN: 0733-947X, pages 1-12.

[11]M. Hossain and A. M. A. Hafiz, "Object detection and tracking using Faster R-CNN with deep convolutional networks," Journal of Signal Processing Systems, April 2021, ISSN: 1939-8018, pages 1-15.

[12]J. S. Yoon and S. J. Oh, "Real-time object detection with PyTorch and SSD," Journal of Visual Communication and Image Representation, October 2020, ISSN: 1047-3203, pages 1-8.