International Journal of Scientific Research in Computer Science, Engineering and Information Technology

ISSN : 2456-3307

OPEN ACCESS

Available Online at : www.ijsrcseit.com

doi : https://doi.org/10.32628/IJSRCSEIT

Fire Detection Systems Using Feature Entropy Guided Neural Network

[1]S K. Ahmed Mohiddin, [2]I T V V S N S Pravallica, [3]K. Pujitha, [4]D. Nandini, [5]S. Preetham

[1]Associate Professor, [2,3,4,5] UG Student

Department of CSE, Sri Vasavi Institute of Engineering & Technology, Nandamuru, Andhra Pradesh, India

ARTICLEINFO

Article History:

Accepted: 10 April 2024

Published: 19 April 2024

Publication Issue

Volume 10, Issue 2

March-April-2024

Page Number

642-651

ABSTRACT

Fire detection from video has become possible and more feasible in prevention of fire disaster due to deep convolutional neural networks (CNNs) and embedded processing hardware. Artificial intelligence (AI) methods generally require more computational time and hardware with powerful graphical processing unit (GPU). In this paper, we propose cost-effective deep CNN architecture for fire detection from video with respect to computational performance of Jetson Nano from NVIDIA. In our paper we compare CNN networks (AlexNet and SqueezeNet) with our proposed CNN architecture. The proposed CNN architecture finds equilibrium between efficiency and accuracy for target system (Jetson Nano). We used CNNs which show high accuracy and low loss.

Keywords : Fire Detection, Convolutional Neural Network, Surveillance.

I. INTRODUCTION

Convolutional neural networks and deep learning were firstly used in late ’90s to recognize digits on check. However, the CNNs are being massively used during last 10 years. The prime reason is the surge in graphical processing units (GPUs) performance and their increasing affordability. GPUs were originally designed for gaming to perform countless millions of matrix operations per second. The GPUs are more optimized for neural networks than central processing units (CPUs) due to lots of matrix operations which are required for neural networks [1]. Similar expansion is happening with embedded processing hardware, where in recent years, we observe applications of CNNs in domains like event monitoring, authentication, autonomous driving, smart surveillance and more [2]. For this paper we have selected only fire and smoke detection, because fire and smoke events are one of the most common events, where early detection can avoid fire disasters and potentially reduce number of victims and costs. According to NFPA report Fire Loss only in the United States during 2018 were reported 1,318,500 fires which resulted in 3,655 civilian fire fatalities, 15,200 civilian fire injuries, and estimated $25.6 billion in direct property loss. On average, there was reported every 2 hours and 24 minutes fire victim and every 35 minutes fire injury. Home fires caused 2,720 deaths which is almost 74%. Fires were accounted for four of the

36,746,500 total calls where 8% were reported as false alarm [11]. Current state of the art smoke detectors are mostly based on ionization, photoelectric or laser technology. However, they are not very reliable in outdoor environment. The availability of open-source software helps to distribute implementation of CNNs rapidly. Below we present some differences between PyTorch, TensorFlow and Keras. All of the mentioned frameworks are widely used by machine learning community, because all of the mentioned frameworks are build on a C/C++ engine and offers Python API. The PyTorch contains a lot of pretrained neural network (NN) models and allows easy execution on GPU. The TensorFlow is slower than other frameworks and has no pretrained NN models.The Keras is a deep-learning API which uses TensorFlow and Theano and allows fast deployment. We used in our paper PyTorch due to faster execution.

II.RELATED WORK

The authors of paper [1] founded the benchmark capitalized on YOLOv4 protocol, developed a model of detection of fire using the Darknet deep learning framework. They employed a wide number of self-built fire datasets for training and testing in the experiment, including multi-scale. Our model outperformed others in fire detection, according to the findings. Based on Convolutional Neural Networks, this research developed a better YOLOv4 fire detection algorithm (CNN). They employ automated dataset of fire detection to increase model accuracy, a modified loss function to improve smallscale flame detection, and a combination of Soft- NMS for improving the suppressive impact of the box which is redundantly bounded and lower low rate of recalling.

The authors of paper [2] studied some problems about the detection of fire in forest areas, detection of fire in early moments, and erroneous detection. They applied traditional detection using objects approaches for determining the fire in the forests for the first time. Real-time properties are being increased by SSD, greater accuracy of detecting the fire, and the potential to detect fires earlier. To reduce the number of false alarms, they create a system using smoke and fire, and used newly introduced adjustments. Meanwhile, they tweak the tiny-yolo structure of YOLO and suggest a new structure. The experiments show that tinyyolo-voc1is improving the rate of accuracy of detection of fire. The document was really useful to safety of forests as well as in monitoring.

The authors of paper [3] proposed a support vector machine-based picture fire detection technique. To begin, the inter-frame difference approach is used to detect the motion region, which is then considered the fire suspected area. The probable fire region is then re-sampled using the scene categorization algorithm. The texture and color moment features of the fire suspected area are then extracted. Finally, the trained machine learning model support vector machine is used to input these eigen values as eigenvectors for fire and non-fire classification and recognition. The experimental findings revealed that the suggested approach in this work can eliminate the drawback of artificially establishing the flame characteristics threshold and has higher accuracy than a traditional flame identification algorithm. However, there are some phenomena that have been overlooked or misinterpreted. This is what the algorithm performs.

The authors of paper [4] used a new approach of fire detection that uses gas leak concentration to anticipate explosions and fires, which was previously known as a fire predictor and a fire appearance detector. Fire detection is one of the smart home's features. The early detection of a fire in the house is a critical step in preventing a large- scale fire and saving numerous items. The fire predictor simply displays the gas leak concentration and sounds an alarm. The classification of fire detectors is done using a fuzzy approach. The data can be sent to MFC from the output simulation system, but the MFC reader cannot understand it in real time.

The authors of paper [5] studied on the hardware of Raspberry Pi circumstances and the Raspberry

Pisoftware, they were able to compensate to deficiencies of standard detectors of fire for improving the dependability of fire deep uses paper YOLO v3 is a modest system of identifying the local videos for shipping fires that uses a lightweight direct regression detection technique. The RPI Fire is gaining high perfection rate, recall rate of video testing and fire simulation, able to match the requirement of ship detection of fire.

The authors of paper [9] proposes a fire recognition approach based on the centroid variety of consecutive frame images, which is based on the flickering characteristic of flame. This method has the advantages of simplicity, speed, and versatility when compared to the standard flame recognition method. It is capable of removing a wide range of non-fire interference sources. This type of fire detection technology, which uses a standard camera, has a lot of practical utility.

The authors of paper [10] proposed the detail information films of the fires in forests in varied situations was used for assessing an aerial-based forest fire identifying approach. To improve the detection rate, the moving properties are first identified, later adjusted by scale to highlight burning region. Second, smoke is retrieved using our proposed technique. In the real use of forest fire detection, our system demonstrates its robustness with a large accuracy rate. using optical flow for computing vectors of motion, then clustering DBSCAN which represents existing models motion patterns. By similarity and criterions of entropy classes resulting matched to motion pattern models. This helps in detecting the motions with some irregularities in traffic.

III.PROPOSED SYSTEM

We conducted experiments from different perspectives using images and videos from different sources and compared the results across different NNs. The final network was deployed on Jetson Nano from Nvidia, therefore the aim from the beginning was to design lightweight NN for embedded applications. We used architecture of Convolutional Neural Network (CNN) which shows good results on image classification.

Proposed system focused on First well known CNN was the LeNet introduced by LeCun in 1990 Since then the CNNs are used to solve different problems including action recognition image classification and more Benefit of CNNs is to learn rich features from raw data. The CNNs consists of convolution layers, pool layers and fully connected layers as illustrated. The AlexNet architecture consists of multiple layers which are described in table I below where resolution of input image is 227x227 pixels and output is number of classes in classification and accuracy is high.

We have used almost all of the same libraries which are used in normal ML/DL problems like pandas, numpy, matplotlib, sklearn etc. But here We want to highlight one important libraries.

1. Keras: This is one of the library which is used to code deep learning models. In its back-end it uses Tensorflow.

2. Tensorflow:This is used for deep lerning framework techniques are using.

CNN is on of deep learning frameworks which is inspired byliving creatures and their mechanism of visual perception. Firstwell known CNN was the LeNet introduced by LeCun in 1990[4]. Since then the CNNs are used to solve different problemsincluding action recognition [5], image classification and more [6]. Benefit of CNNs is to learn rich features from raw data. The CNNs consists of convolution layers, pool layers and fully connected layers as illustrated in figure 1. For our experiment we selected AlexNet and SqueezeNet and compared it to our CNN architecture. Subsections below describes each CNN.

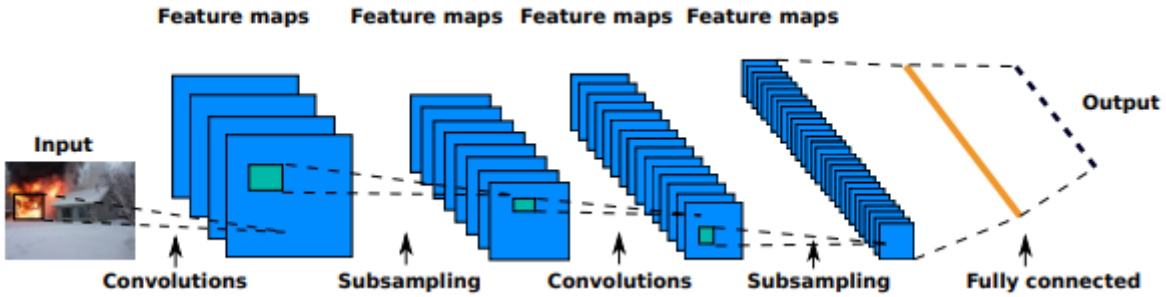

Fig 1 : Architecture of typical CNN

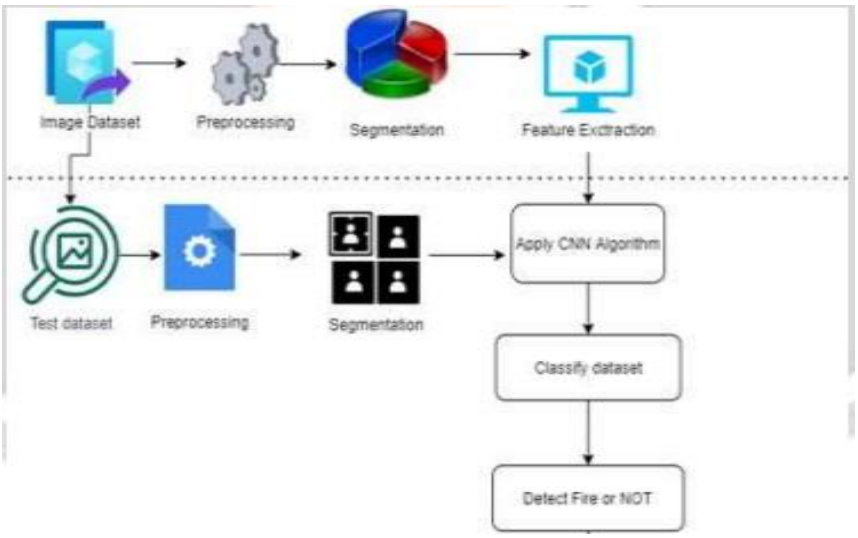

Fig 2 : Detailed Flow of Architecture

In the above figure, we get complete information of how image capture to the how image detected.

1. Preprocessing: In this method we collect all raw data from capturing image frame and perform operation on it is called preprocessing. For example: Training algorithm i.e. CNN on photo which capture by it is result in not good classification results.

2. Feature Extraction: A CNN is a neural network that extracts image attributes froman input image and classifies them using another neural network. The feature extraction network works with the input image. The neural network makes use of the feature signals extracted.

3. Segmentation: CNN (CNN feature) is an application of region- based approaches in the real world. On the basis of the object detection results, it does semantic segmentation. R-CNN employs selective search to extract a large number of object ideas before computing CNN characteristics for each one.

IV. RESULTS AND DISCUSSION

```
import tensorflow as tf
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator
import shutil
TRAINING_DIR = "tmp5"

training_datagen = ImageDataGenerator(rescale = 1./255,
                                      horizontal_flip=True,
                                      rotation_range=30,
                                      height_shift_range=0.2,
                                      fill_mode='nearest')

VALIDATION_DIR = "tmp6"
validation_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = training_datagen.flow_from_directory(
    TRAINING_DIR,
    target_size=(224,224),
    class_mode='categorical',
  batch_size = 64
)

```

Fig 3. Results screenshot

```
Found 980 images belonging to 2 classes.
Found 239 images belonging to 2 classes.
```

Fig 4. Results screenshot

```
from tensorflow.keras.optimizers import RMSprop,Adam
model = tf.keras.models.Sequential([
          tf.keras.layers.Conv2D(96, (11,11), strides=(4,4), activation='relu', input_shape=(224, 224, 3)),
          tf.keras.layers.MaxPooling2D(pool_size = (3,3), strides=(2,2)),
          tf.keras.layers.Conv2D(256, (5,5), activation='relu'),
          tf.keras.layers.MaxPooling2D(pool_size = (3,3), strides=(2,2)),
          tf.keras.layers.Conv2D(384, (5,5), activation='relu'),
          tf.keras.layers.MaxPooling2D(pool_size = (3,3), strides=(2,2)),
          tf.keras.layers.Flatten(),
          tf.keras.layers.Dropout(0.2),
          tf.keras.layers.Dense(2048, activation='relu'),
          tf.keras.layers.Dropout(0.25),
          tf.keras.layers.Dense(1024, activation='relu'),
          tf.keras.layers.Dropout(0.2),
          tf.keras.layers.Dense(2, activation='softmax')
])
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001),
              metrics=['acc'])
model.summary()
```

Fig 5. Results screenshot

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 54, 54, 96)        34944
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 26, 26, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 22, 22, 256)       614656
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 256)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 6, 6, 384)         2457984
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 384)         0
_________________________________________________________________
flatten (Flatten)            (None, 1536)              0
_________________________________________________________________
dropout (Dropout)            (None, 1536)              0
_________________________________________________________________
dense (Dense)                (None, 2048)              3147776
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              2098176
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 2050
=================================================================
```

Fig 6. Results screenshot

```
# class myCallback(tf.keras.callbacks.Callback):
#   def on_epoch_end(self, epoch, logs={}):
#     if(logs.get('val_acc')>=0.98):
#       print('\nReached ^98%')
#       self.model.stop_training = True
# callbacks = myCallback()

history = model.fit(
    train_generator,
    steps_per_epoch = 15,
    epochs = 50,
    validation_data = validation_generator,
    validation_steps = 15
    #callbacks=[callbacks]
)
```

```
Epoch 1/50
15/15 [==============================] - 41s 3s/step - loss: 0.5109 - acc: 0.7162 - val_loss: 0.3878 - val_acc: 0.8577
```

Fig 7. Results screenshot

```
# class myCallback(tf.keras.callbacks.Callback):
#   def on_epoch_end(self, epoch, logs={}):
#     if(logs.get('val_acc')>=0.98):
#       print('\nReached ^98%')
#       self.model.stop_training = True
# callbacks = myCallback()

history = model.fit(
    train_generator,
    steps_per_epoch = 15,
    epochs = 50,
    validation_data = validation_generator,
    validation_steps = 15
    #callbacks=[callbacks]
)
```

```
Epoch 1/50
15/15 [==============================] - 41s 3s/step - loss: 0.5109 - acc: 0.7162 - val_loss: 0.3878 - val_acc: 0.8577
```

Fig 8. Results screenshot

```
%matplotlib inline
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'g', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()
plt.show()

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'orange', label='Validation loss')
plt.title('Training and validation loss')

plt.legend(loc=0)
plt.figure()
plt.show()
```

Fig 9. Results screenshot

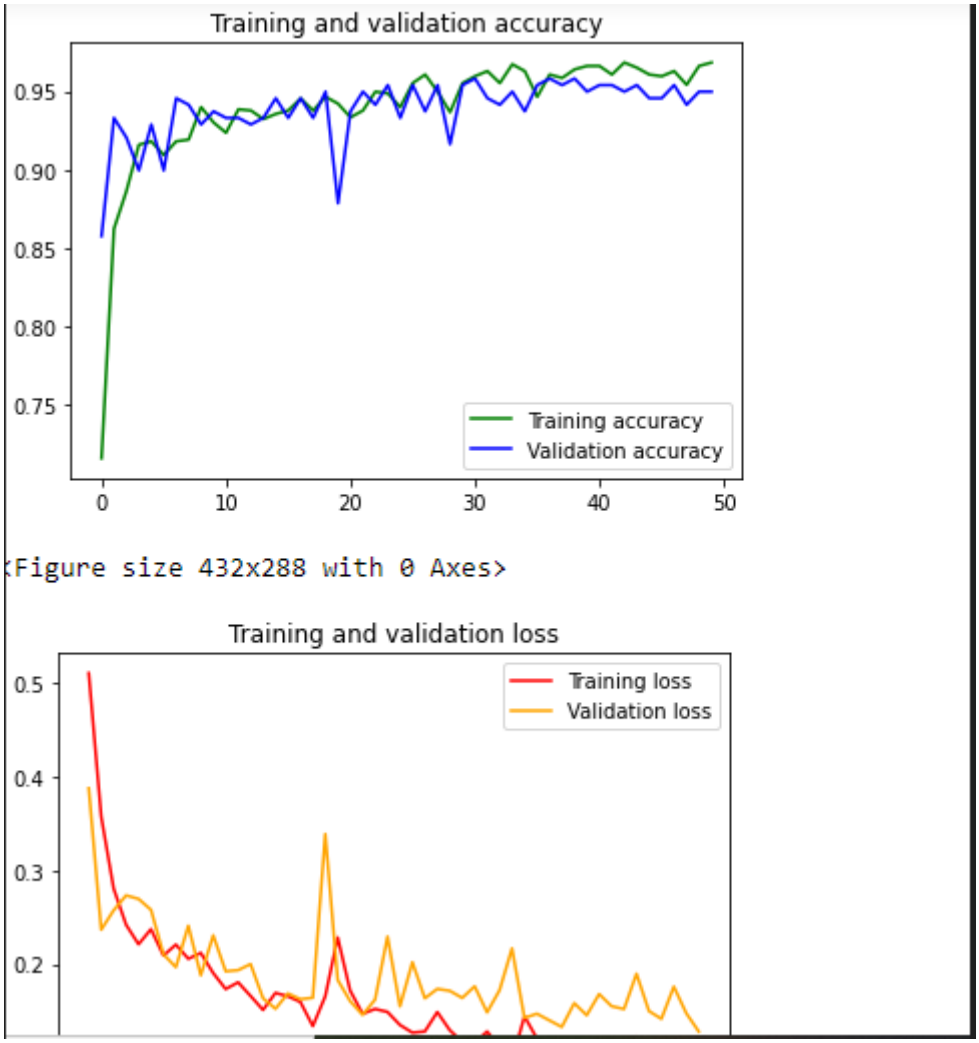

Fig 10. Results screenshot

```
1  img = image.load_img('/content/NITPDL02 - Deep Convolutional Neural Network for Fire Detection/NITPDI
2  x = image.img_to_array(img)
3  x = np.expand_dims(x, axis=0) /255
4  classes = model.predict_classes(x)
5  if classes[0]==1:
6      print("The image is Neutral ")
7  else:
8       print("The image is Fire ")
9  #print(np.argmax(classes[0])==0, max(classes[0]))

The image is Fire
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning: `mode
  warnings.warn('`model.predict_classes()` is deprecated and '
```

Fig 11. Results screenshot

V. CONCLUSION

In this paper we introduced our fire detection network and compared it to existing CNNs. The comparison of CNNs was Authorized licensed use limited to: University of Wollongong. performed on fire detection from image using two different datasets for training of CNNs. The goal was to design lightweight CNN model with high accuracy results. The fire detection network model after training achieved smallest size (683.1kB), which is less than the Squeezenet (5.0MB) or the

AlexNet (228.0MB) before bit-width compression. Measured accuracy of the fire detection network model was 79.66% on the dataset1 and 88.0% on the dataset2 using embedded hardware (Jetson Nano). On the dataset1 we measured accuracy for the AlexNet model (81.66% of accuracy) and the Squeezenet model (77.33%). On the dataset2 the AlexNet model achieved 88.0% and the SqueezeNet 92.5% of accuracy. In terms of execution time, our fire detection network model showed best execution time per image processing on both datasets. Measured time for thedataset1 for the Fire detection network achieved 0.052s per image, the Squeezenet model executed in 0.062s per image, and the AlexNet model in 0.081s. On the dataset2 the fire detection network model performed in 0.109s per image, the Squeezenet in 0.127s and the AlexNet in 0.140s. In this paper we have also demonstrated potential future steps of development in fire detection systems.

VI. FUTURE WORK

In this paper we have also demonstrated potential future steps of development in fire detection systems. Research described in this paper was partially supported by the Czech Ministry of Education in frame of National Sustainability Program under grant LO1401. For research, infrastructure of the SIX Center was used.

VII.REFERENCES

[1]. AAA lkhatib, “Smart and Low Cost Techniquefor Forest Fire Detection using Wireless Sensor Network,” Int. J. Comput. Appl., vol. 81, no. 11, pp. 12–18, 2013.

[2]. J. Zhang, W. Li, Z. Yin, S. Liu, and X. Guo,“Forest fire detection system based on wireless sensor network,” 2009 4th IEEE Conf. Ind. Electron. Appl. ICIEA 2009, pp. 520–523, 2009.

[3]. A. A. Alkhatib, “A review on forest firedetection techniques,” Int. J. Dis- trib. Sens. Netw.,vol. 2014, no. March, 2014.

[4]. P. Skorput, S. Mandzuka, and H. Vojvodic,“The use of Unmanned Aerial Ve- hicles for forest fire monitoring,” in 2016 International SymposiumELMAR, 2016, pp. 93–96.

[5]. F. Afghah, A. Razi, J. Chakareski, and J. Ashdown, Wildfire Monitoring in Remote Areasusing Autonomous Unmanned Aerial Vehicles. 2019.

[6]. Hanh Dang-Ngoc and Hieu Nguyen-Trung,“Evaluation of Forest Fire De- tection Model usingVideo captured by UAVs,” presented at the 2019 19th International Symposium on Communicationsand Information Technologies (ISCIT), 2019, pp. 513–518.

[7]. C. Kao and S. Chang, “An Intelligent Real- Time Fire-Detection Method Based on VideoProcessing,” IEEE 37th Annu. 2003 Int. CarnahanConf. OnSecurity Technol. 2003 Proc., 2003.

[8]. N. I. Binti Zaidi, N. A. A. Binti Lokman, M. R.Bin Daud, H. Achmad, and K. A. Chia, “Firerecognition using RGB and YCbCr color space,” ARPN J. Eng. Appl. Sci., vol. 10, no. 21,pp. 9786–9790, 2015.

[9]. C. E. Premal and S. S. Vinsley, “Image Processing Based Forest Fire Detection using YCbCr Colour Model,” Int. Conf. Circuit PowerComput. Technol. ICCPCT, vol. 2, pp. 87–95, 2014.

[10].C. Ha, U. Hwang, G. Jeon, J. Cho, and J.Jeong, “Vision-based fire detection algorithm usingoptical flow,” Proc. - 2012 6th Int. Conf. ComplexIntell. Softw. Intensive Syst. CISIS 2012, pp. 526– 530, 2012.

[11].K. Poobalan and S. Liew, “Fire Detection Algorithm Using Image Processing Techniques,”Proceeding 3rd Int. Conf. Artif. Intell. Comput. Sci., no. December, pp. 12–13, 2015.

[12].B. Pradhan, H. A. H. Al-Najjar, M. I. Sameen, I. Tsang, andA. M. Alamri, “Unseen land cover

classification from High Resolution orthophotos using integration of zero-shot learn ing and convolutional neural networks," Remote Sensing,vol. 12, no. 10, 2020.

[13].M. B. A. Gibril, B. Kalantar, R. Al-Ruzouq et al., "Mapping het erogeneous urban landscapes from the fusion of digital surfacemodel and unmanned aerial vehicle-based images using adap tive multiscale image segmentation and classification," RemoteSensing, vol. 12, no. 7, p. 1081, 2020.

[14].M. Hasanlou, R. Shah-Hosseini, S. T. Seydi, S. Karimzadeh, and M. Matsuoka, "Earthquake damage region detection bymultitemporal coherence map analysis of radar and multispec tral imagery," Remote Sensing, vol. 13, no. 6, p. 1195, 2021.

[15].M. Hasanlou and S. T. Seydi, "use of multispectral and hyper spectral satellite imagery for monitoring Waterbodies and Wetlands," Southern Iraq's Marshes, vol. 36, pp. 155–181,2021.