EEET ECOLOGICAL ENGINEERING & ENVIRONMENTAL TECHNOLOGY

Ecological Engineering & Environmental Technology 2024, 25(10), 236–257
https://doi.org/10.12912/27197050/191861
ISSN 2299–8993, License CC-BY 4.0

Received: 2024.07.27
Accepted: 2024.08.15
Published: 2024.09.01

Training Issues in Classifying Urban Flood Object Detection – A Deep Learning Study

Padmavathi Lambu[1], Rajesh Duvvuru[1*]

[1] School of Computer Science and Engineering, VIT-AP University, Guntur, India

* Corresponding author's email: dr.rajeshduvvuru@gmail.com

ABSTRACT

Climate change has had a significant impact on natural disasters, particularly floods, in recent times. Early warning systems play an important role in river flood prediction. However, cloudbursts trigger urban flash floods, causing significant disruption to humanity, property damage, and loss of life. In particular, the number of deaths from urban floods has increased in recent times, primarily due to a lack of information. Urban floods, in particular, inflict damage on assets such as vehicles, electric poles, and plants. In addition, flash floods in urban areas submerge roads, drainage, etc., leading to drowning and fatalities. Currently, there is a need to develop smart urban flood prediction and monitoring systems that disseminate instant flood information to rescue teams for a quick response. Currently, deep learning technologies play a significant role in object prediction, but their accuracy in predicting urban flood objects is relatively low. In deep learning algorithms, the training of networks, in conjunction with optimizers and epochs, plays a crucial role in achieving higher accuracies in object detection. The current article investigates the best deep learning training networks, optimizers, and epochs to train urban flood data objects that can achieve higher accuracy. This study considers two pre-trained models, XceptionNet and AlexNet, and three optimizers, including SGDM, ADAM, and RMSProp, to train the urban flood dataset, ensuring balance. We evaluate each training network's performance with the optimizer by tuning epochs and hyper-parameters as constants. Specifically, applying XceptionNet to the SGDM optimizer resulted in an accuracy of 97.47%. The results show that XceptionNet outperforms AlexNet in terms of performance and is recommended for flood object classification. Currently, the study solely focuses on two pre-trained networks and achived 97.47% accuracy; however, it has the potential to evaluate other deep convolutional neural networks, potentially achieving 100% data object training.

Keywords: deep learning, CNN algorithms, flood detection, xceptionnet, alexnet.

INTRODUCTION

Natural disasters caused by climate change, such as earthquakes, tsunamis, floods, wildfires, and other communicable diseases [Seidel et al., 2024; Rao et al., 2024; Prathyusha et al., 2024], have become very common in recent times. Whether natural or human-caused disasters, result in a high fatality rate and significant financial losses. Flooding stands out as a prominent natural disaster, often triggered by rapid environmental and climatic changes. In recent days, floods that have occurred in many places all across the world have resulted in high scores of deaths and have also broken various records for rainfall that have been in a few places for a very long time. Floods are unpredictable due to several meteorological and environmental conditions, making them challenging to forecast. Although floods are inevitable natural occurrences, it is essential to implement preventive measures in order to limit loss of life and mitigate economic harm [Wenchao et al., 2021; Bentivoglio et al., 2022; Faruq et al., 2020]. India is one of the world's countries experiencing severe urban and rural floods. India has identified approximately 12% of its land as a flooding zone, covering 40 million hectares. Andhra Pradesh, Uttar Pradesh, Haryana, Punjab, Bihar, and Gujarat are especially vulnerable states. In particular, the state of Andhra Pradesh is

experiencing severe urban flooding due to heavy rainfall, as shown in Figure 1. Figure 1 shows a gradual increase in rainfall from 2014 to 2020. The severity of rainfall is increasing day by day in all 13 districts of Andhra Pradesh, as shown in Figure 2. Out of the 13 districts, West Godavari, East Godavari, Guntur, and Chittoor districts experienced high rainfall in Srikakulam district, while the remaining 8 districts (Nellore and Vijayawada) experienced normal rainfall. As per the Central Water Commission (CWC) Reports 2020, the Godavari River, which flows through Andhra Pradesh, has witnessed floods 34 times between 1953 and 2019. Flooding remains a significant concern for the state, impacting both people and infrastructure [Bentivoglio et al., 2022; Sankaranarayanan et al., 2020]. Currently, cloud bursts cause urban flash floods, resulting in severe damage to the public and infrastructure. Global warming has led to a drastic increase in urban and flash floods, as evidenced by the floods in Chennai in 2015, Hyderabad in 2020, and Bengaluru in 2022, among others. Recent global warming trends predict a significant increase in urban flooding, which could potentially trigger severe disasters and impact 60 percent of the country's population. Despite the increasing frequency of disasters causing numerous fatalities, the effectiveness of early warning systems remains limited in our current smart digital era [Xinxiang et al., 2021].

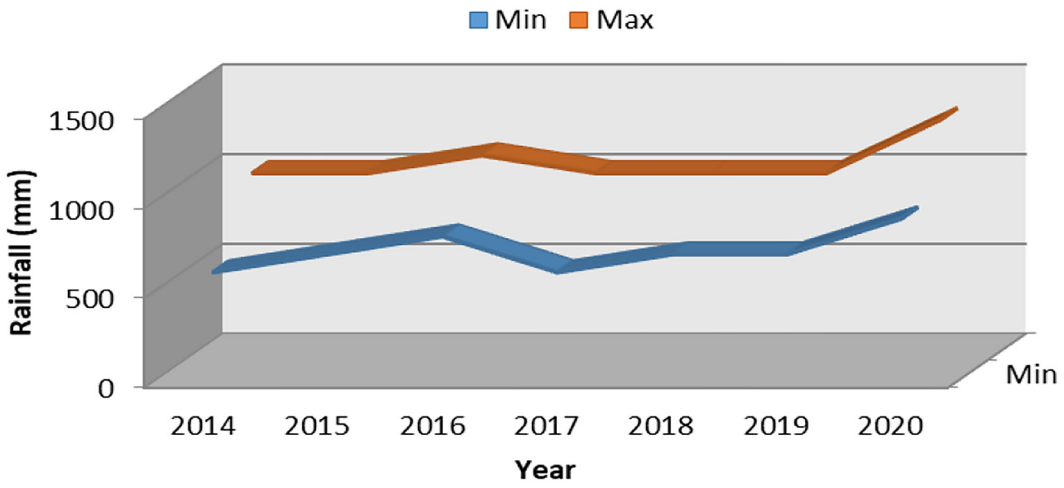

Figure 1. Year wise minimum and maximum rainfall in Andhra Pradesh state during 2014–2020 (Source: Central Water Commission, Govt. of AP, India)

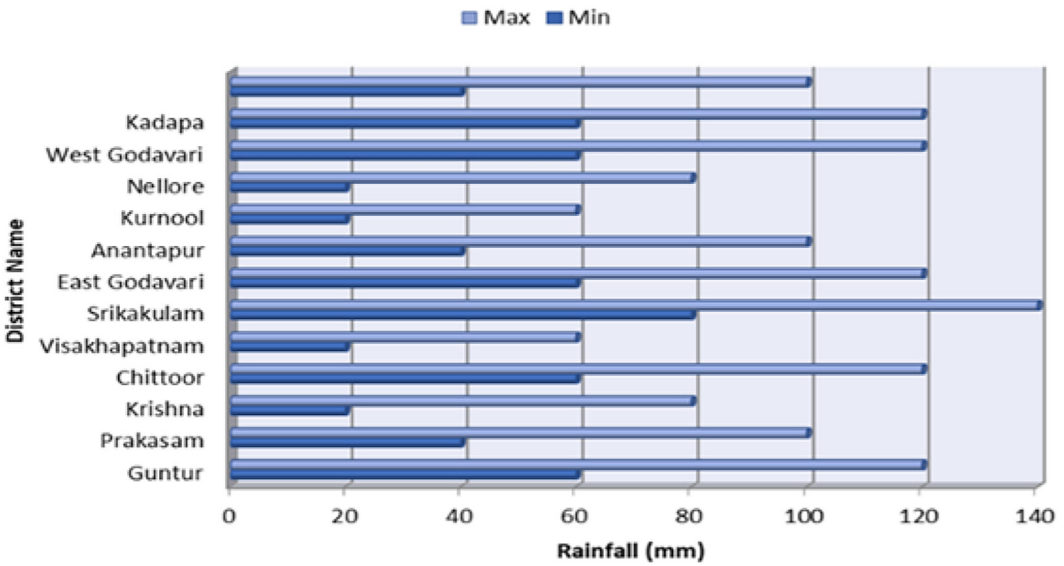

Figure 2. Year wise district minimum and maximum rainfall of Andhra Pradesh state during 2014–2020. (Source: Central Water Commission, Govt. of AP, India)

A six-year-old girl named Athidi drowned in an urban flood incident in Visakhapatnam Smart City due to heavy rain, and authorities reported her death five days later [The News Minute, 2015]. Another flood event trapped three AP State Road Transport Corporation buses carrying passengers in flood waters near Ramapuram in Rajampet mandal. Despite the fire services rescuing most of the passengers and APSRTC staff, they found 12 washed-away persons dead at various places in Rajampet mandal of Kadapa district [The News Minute, 2021].

The flooding of Kurnool town in Andhra Pradesh, India, in October 2009 illustrates how several factors combined to cause one of the worst floods in 100 years. More than 30 feet of water submerged several areas, and it took over three days for the water to fully recede from the town, affecting 4,70.000 people [Ramachandraiah, 2011]. These types of incidents are not unique to Andhra Pradesh; the rest of the states are also experiencing similar urban flood effects. An incident in Bangalore city highlights the deficiency in the early warning system for urban floods [Deccan Herland, 2023]. On Sunday evening, the car carrying a software engineer from Andhra Pradesh drowned in the storm water at an underpass near KR Circle in Bengaluru. We identified the deceased as Bhanurekha, a native of Vijayawada. Many such incidents are happening all over the globe due to heavy rains and flash floods. At present, there is a technical challenge in developing an Urban Flood (UF) early warning system. While rescue teams like the National Disaster Response Force (NDRF) are well-trained and equipped to respond swiftly, the dissemination of location information and early warnings remain two critical tasks for effective UF rescue operations (Eric et al., 2020). At present, there is a need for early identification of urban flood regions. The researchers have addressed the conditions of pre-flood and post-flood areas using SAR (Synthetic Aperture Radar) images to support rescue operations across coastal regions [Shreekumar et al., 2021]. Researchers have recommended diverse solutions through deep learning algorithms using a variety of SAR datasets to detect floods [Eric et al., 2020].

Urban floods will have a significant impact, leading to the drowning of humans and other objects such as vehicles, road networks, electric poles, fallen trees, and manholes. In nations like India, the lack of urban flood water-level alert systems contributes to the drowning of numerous people, leading to fatalities; in some cases, the bodies remain unidentified and untraceable for months. In particular, poor solid and liquid waste management during post-urban floods contributes to the spread of communicable diseases. Currently, rescue teams are encountering a technological challenge in obtaining immediate information about urban floods, a crucial step in expediting rescue operations to save lives and valuable assets. Currently, there is a need to develop smart urban flood prediction and monitoring systems that disseminate instant flood information to rescue teams for a quick response. Currently, river-based flood prediction tasks have advanced significantly, incorporating factors such as flood velocity, water levels, and rainfall, and utilizing the Internet of-Things (IoT) and artificial intelligence (AI) to inform the public about flood events. Currently, AI-based deep learning technologies play a significant role in object prediction, but their accuracy in predicting urban flood objects is relatively low. In deep learning algorithms, the training of networks, in conjunction with optimizers and epochs, plays a crucial role in achieving higher accuracies in object detection.

At present, there is a need to develop smart artificial intelligence technologies that function intelligently, capable of alerting citizens and local administrations to initiate immediate rescue operations. Detecting objects in flood water poses a significant challenge, particularly during urban floods. The urban flood object detection accuracy poses a significant challenge for researchers and practitioners, as achieving higher detection accuracies primarily depends on the object's training accuracy. However, researchers have conducted limited pre-training studies to assess the performance of urban flood object detections. In general, if the training accuracy is high, it may result in excellent test accuracy. Despite the network training model's reputation for object detection, the epoch rate at which the data undergoes optimal training remains unknown. The current work focuses on incorporating deep learning technologies to identify and monitor flood objects in urban areas, where the detection accuracy of multi-class flood objects is poor [Anil et al., 2022; Yazeed et al., 2022]. Generally, object detection accuracy depends on training accuracy, but researchers are often unsure how to set hyperparameters such as learning rate, epochs, optimizers, etc. to achieve 100% accuracy.

Object detection accuracy is dependent on training quality. Researchers often struggle to optimize hyperparameters like learning rate, epochs, optimizers, etc., for perfect accuracy. The choice of training network also depends on data characteristics and the number of classes, aiming for higher accuracy. The current investigation adopted two popular pre-trained models, namely AlexNet and XceptionNet, to attain higher training accuracy on a novel custom flood object dataset. We collected 110 flood images from various regions of Andhra Pradesh state without segmentation for this analysis. Incorporating a segmentation block into the main procedure can enhance classification accuracy while simultaneously reducing processing time. We run each of the training networks separately, keeping all other hyperparameters (like learning rate, etc.) the same. We change optimisers like SGDM, ADAM, and RMSProp from 20 to 100 epochs. We look at how well AlexNet and XceptionNet worked and find that XceptionNet did better with SDGM, ADAM, and RMSPROP, getting a 96.47% accuracy rate in training classification. However, in each epoch, the success rate of SGDM is higher than that of ADAM and RMSProp. In the present work, the Xception Net pre-trained model shows good training accuracy of 97.67% when classes are limited and less complex. However, when trained on flood datasets with three types of flood images, both AlexNet and Xception Net pre-trained models get between 23% and 50% accuracy, with extra help [Gaffinet et al., 2023; Vanama et al., 2020]. The current study solved important training problems in balanced flood object datasets by finding the best deep learning network, optimizers, and epoch rate to improve training accuracy. The present study is limited to verifying the best training accuracy on popular deep learning models such as XceptionNet and AlexNet, whereas very little research has been performed on Urban Flood Object data. This work aims to find the optimal hyperparameters to attain higher accuracies on XceptionNet and AlexNet using flood datasets [Mohammadtaghi et al., 2021]. The analysis of the dataset training results suggests that XceptionNet is the optimal network for applying detector algorithms.

The current research solely focuses on training problems with flood object datasets, and additionally, it must undergo accuracy tests using detector algorithms such as YOLO, MobileNet, etc. The current study discovered that the present XceptionNet is preferable to training the flood object dataset if they are associated with water images. This technical analysis is beneficial to the National Disaster Response Force (NDRF), which currently has limited facilities for dealing with Urban Floods (UF). Implementing a system to share Location Information and Early Warnings periodically would allow for early detection of urban floods, potentially reducing economic losses and fatalities [Anil et al., 2022].The rest of the article is organized as follows: Section II begins by reviewing relevant literature and technique to set the stage for our investigation. Section III details our novel approach and strategy. Section IV explains the steps taken to achieve our goals, providing a clear research plan. Section V analyses the recommended technique, its efficacy, and its consequences, and presents our findings in depth. Section VI Concludes by providing a concise overview of our primary discoveries and proposing potential avenues for future investigation.

RELATED WORKS

This study examines various ways utilizing deep learning algorithms to predict urban floods [Hashi et al., 2021]. The Esfandiari et al (2020) presented an advanced approach to flood detection and classification, addressing limitations in previous studies that relied on single image types like SAR (Synthetic Aperture Radar) [Esfandiari et al., 2020]. CNN observe multi-class scenarios along with water flow levels to differentiate emergency of flood events on the visible impacts of flooded regions. At the event, a Flood picture was put on a CNN design and tested on a part of the Custom Dataset from Google. The proposed Convolutional Neural Network (CNN) architecture outperforms pre-trained VGG models in terms of precision and retraining time on standard PCs [Maurya et al., 2021].

To enhance the dataset without additional data collection, rotational changes were applied as an augmentation technique. Despite these advancements, the resulting 96.67% accuracy is considered insufficient for flood detection applications, underscoring the critical need for extremely high precision in emergency response scenarios [Ghasemi et al., 2020]. Ahmed et al. developed and deployed a flood segmentation system named DeepLab, which use a deep learning algorithm to precisely and efficiently identify and differentiate the presence and magnitude of floods using visual data. The neural

network underwent training using a vast assortment of satellite photos, which were augmented with ground truth labels denoting the existence of flooded regions. During the inference process, the DeepLabv3 model, which has been trained, is utilized to predict the probability of each pixel in new satellite photos being part of a flooded area [Ahmed et al., 2023]. Braveen et al. proposed a unique Internet of Things-based IDI Sense app has been presented to monitor and transmit real-time parameters concerning dams and weather conditions. Initially, a combination of rainfall sensors, waterfall sensors, flow sensors, and ultrasonic sensors are utilized to accurately measure the quantity of water, the extent of vacant space in the dam, and the velocity of water flow within the dam. The data gathered by sensors positioned at several locations on the dam is transmitted to an Arduino device that is connected to the Internet. The spiking neural network (SNN) is utilized to forecast rainfall by analysing historical data from the meteorology dataset [Braveen et al., 2023]. However, the pursuit of even higher accuracy remains a priority for reliable real-world deployment, highlighting potential areas for future research such as incorporating additional data sources, exploring more advanced model architectures, and further refining data augmentation techniques.

METHODOLOGY

The study utilizes the analysis process depicted in Figure 3. The recommended approach consists of six main steps: Data Collection, Preprocessing, Learning Flood Data, DCNN, Classifier, and Test Flood Data. It also recommends the highest training accuracy for the flood object dataset. The methodology outlines the consecutive process of two convolutional network topologies – AlexNet and XceptionNet. Using input sizes of 227 × 227 pixels for AlexNet and 299 × 299

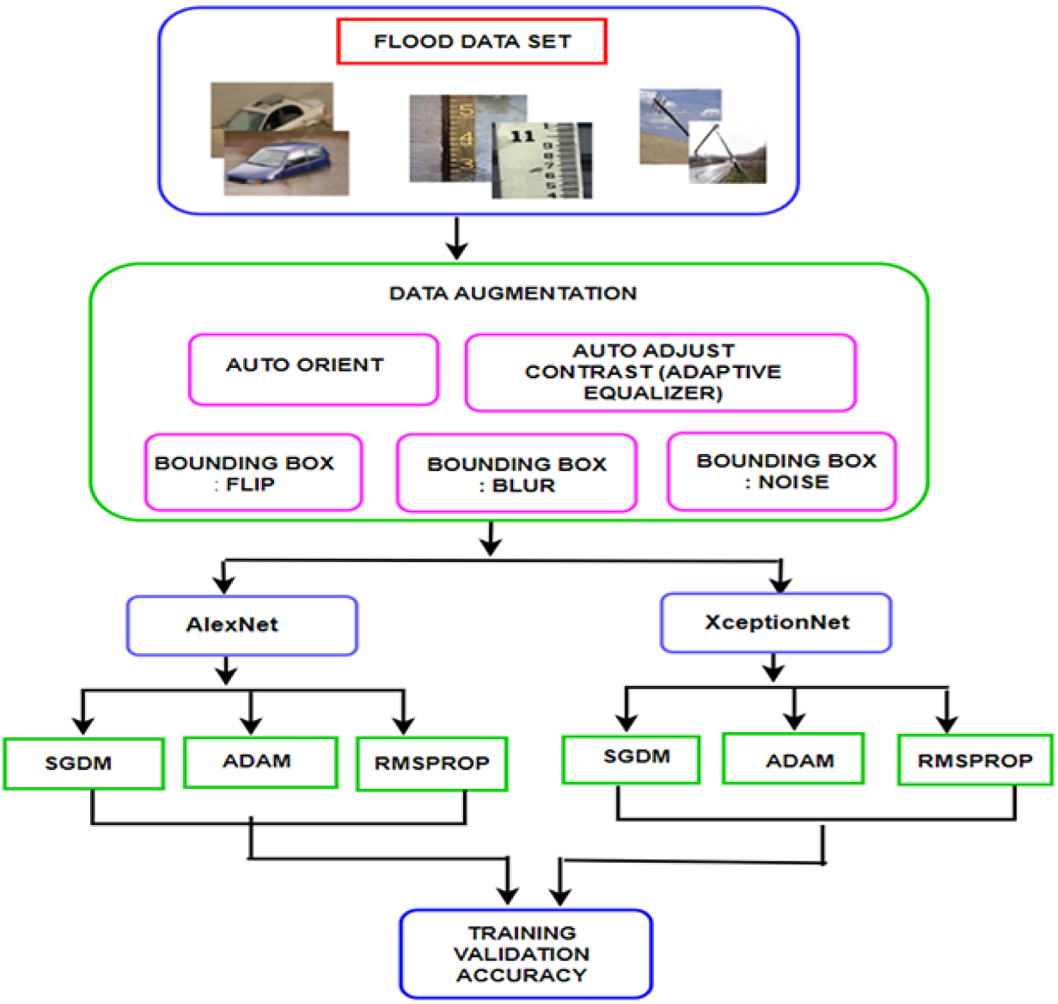

Figure 3. Methodology for data training on custom flood object dataset

pixels for Xception Net, the study put into practice a custom three-class flood dataset collected from Google. This dataset is applied to both models, experimenting with various training options including different solvers or optimizers, epoch numbers, and training parameters such as learning rate, constant etc. The use of a custom Google-sourced dataset and the exploration of different training parameters highlight the efforts to find the most accurate flood image classification.

MATHEMATICAL NOTATIONS AND ASSUMPTIONS

As specified earlier section, the XceptionNet and AlexNet is used for the study. The mathematical representation of XecptionNet and AlexNet is represented in equation 1. The flood detection classification is performed on the feature map's function. The feature map F(z) is the product of Input (M) and Kernels (W). The feature classification function is designated in Equation 1. Where, 'S' represents the list of flood data samples that range from -∞ to ∞ and Q, M, N represents Feature maps, Input and Kernels respectively.

$$F(z) = [q]\,(M \times N) = \sum_{s=-\infty}^{s=\infty} N\,[q + S]\,M[S] \quad (1)$$

Flood object data preparation

The Flood team compiles the bespoke Flood object dataset without any processing. The Flood team compiled it from both fieldwork and online sources. Figure 3 illustrates that the compiled dataset consists of approximately one hundred photos, each associated with three distinct categories. Images in the collection showcase key locations that can significantly impact metropolitan zones worldwide, including India. We save each downloaded photograph in a unique JPG file, utilizing the RGB color space by default. Similarly, class 1 consists of submerged automobiles, while class 2 comprises water markers indicating the water depth in the submerged areas. Finally, class 3 encompasses the collection of fallen electric poles.

This is because severe storms typically impact them swiftly, and most floods and cyclones are considered natural disasters. Additionally, electricity is the primary cause of severe shocks, which have the potential to result in fatalities. Therefore, the current detection of fallen electric poles plays a crucial role in preventing life-threatening shocks.

Dataset pre-processing

Pre-processing facilitates the elimination of unwanted distortions and boosts particular characteristics that are essential for the intended purpose. We utilize the subsequent pre-processing methodologies throughout the dataset preparation:

1. Data profiling: the unprocessed Flood object dataset was subjected to profiling stages including analysis of size distribution, shuffling, brightness distribution, and color distribution. The Matlab simulation software is utilized to analyze Flood datasets, with a focus on image quality metrics like brightness, aspect ratios, convexity, color distribution, fine details, resolution and segmentation of flood affected images [Rauschmayr et al., 2022].
2. Data cleaning: we eliminated erroneous, corrupted, improperly formatted, redundant, or unfinished photos of flood objects impacted from the dataset. A total of 139 photographs depicting 'Car' were received, and 29 images were removed from the collection. Similarly, we captured 126 images and discarded 16 of them; most of them are duplicates, while only a few are hazy. Class 3, like the previous class, has 142 photographs labeled as 'Watermarkers'. However, 32 of these photos were removed since they had low resolution and brightness. We remove 77 anomalous data points from the original dataset of 407, resulting in a remaining set of 330 images to restore the balance of the dataset [Angloher et al., 2023].
3. Balanced dataset: the classification of balanced datasets yields higher accuracy and lower bias in comparison to imbalanced datasets. For the current research, a selection of 330 high-quality flood photographs was made. These images were then resized to a resolution of 720 × 720 pixels in each category. Subsequently, these photos undergo enhancement (Chen et al., 2021).
4. Augmentation: each flood class image undergoes an augmentation operation that involves Contrast (Adaptive Equalization), Bounding Boxes, Flip (Horizontal (180O) and Vertical (90O)), Blur, and Noise. This procedure is repeated eight times, resulting in a total of 2640 augmented images [Ignatowicz et al., 2024]. The dataset is partitioned into three groups, with 70% allocated for training, 20% for testing, and validation 10%.

Figure 4. Sample urban flood object dataset class (A–B): car; (C–D): water-markers; (E–F): electricity poles

To accelerate the training process and ensure more realistic model testing calculations, we resized the image dataset. Specifically, images were downsized to 227 × 227 pixels for AlexNet and 299 × 299 pixels for XceptionNet, as illustrated in Figure 5.

ARCHITECTURE OF DEEP CONVOLUTION NEURAL NETWORK

Convolutional neural networks (CNNs) are a prominent deep learning architecture requires thorough training across multiple layers. They have gained widespread adoption in computer vision tasks due to their remarkable effectiveness. CNNs excel at processing complex visual inputs, transforming chaotic image data into meaningful classification results. This approach has been refined to provide a structured framework for tackling various visual recognition challenges. According to Figure 6, there are 3 layers: Kernels, featured map and fully connected layers where each layer has its own functional behaviour. Fully connected layers transform the output by compressing it, encoding spatial information into the channel dimension through reshaping.

A kernel, also called a filter, is a small matrix of learnable parameters used in the convolution operation. It detects specific patterns or features in the input data. Feature maps are the outputs of convolutional layers, representing distinct features from the input. This process transforms images into numerical features.

XceptionNet architecture

The architecture is built entirely on depth wise separable convolution layers, representing a novel approach in convolutional neural networks. This design was introduced to address the computational costs associated with traditional convolution operations. The key innovation of XceptionNet lies in utilization of two-stage convolution process: (i) depth wise convolution and (ii) pointwise convolution are used to combine information

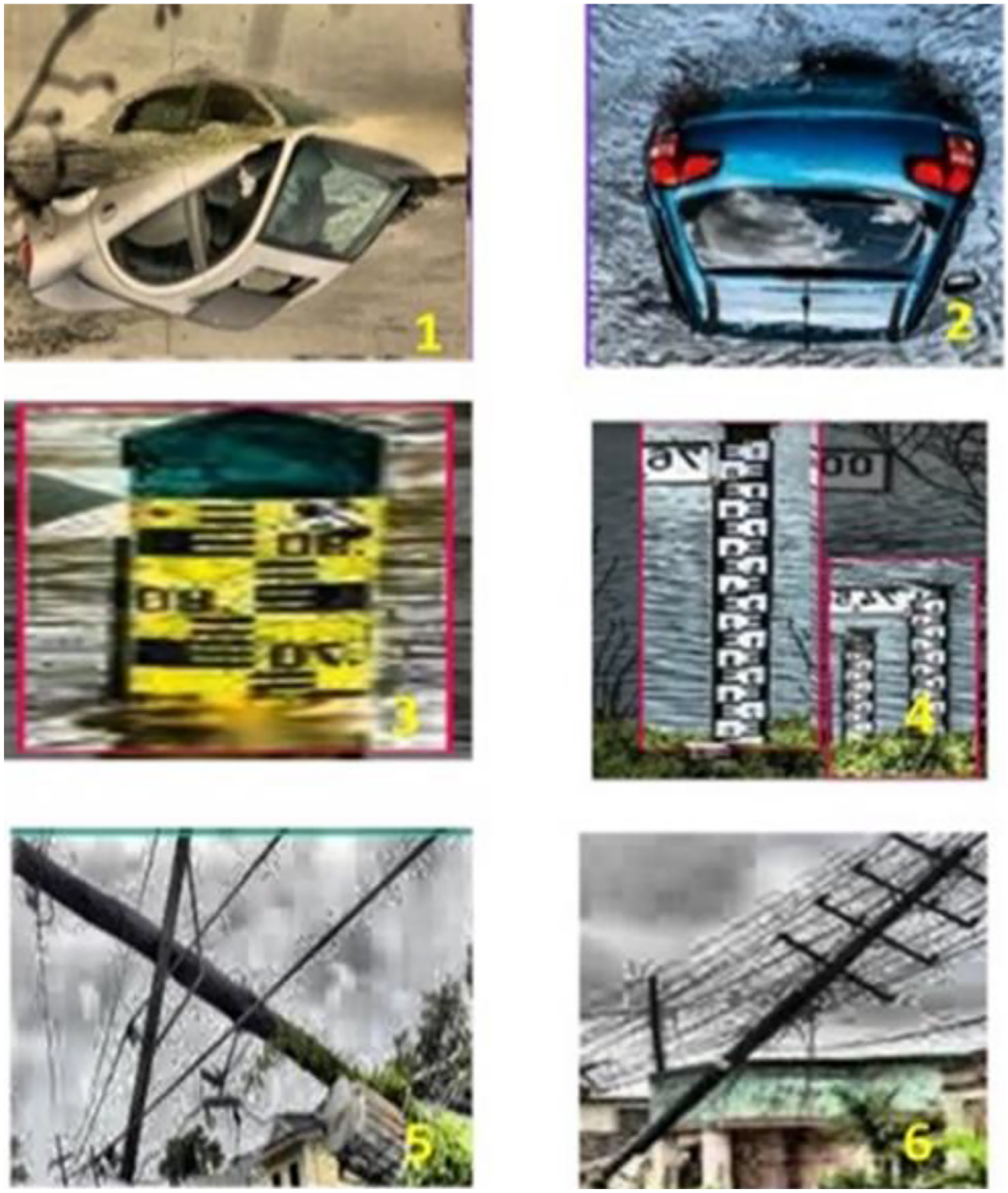

Figure 5. Annotated (pre-processed) urban flood object dataset class (1–2): car; (3–4): water-markers; (5–6): electricity poles

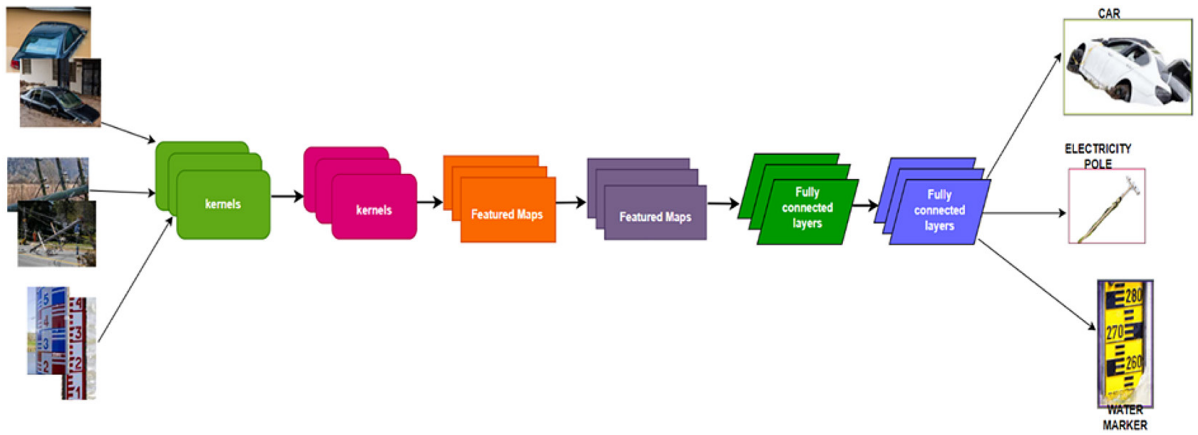

Figure 6. Architecture of deep convolution neural network to classify flood objects

across all channels. By decomposing the convolution process into these two distinct steps, XceptionNet achieves greater computational efficiency while maintaining or even improving performance in various deep learning tasks, particularly in image processing and computer vision applications.

ReLU Layer

The ReLU (Rectified linear unit) layer is the most prevalent in the XceptionNet model, appearing 35 times throughout the network. It performs a thresholding operation on each input element, zeroing out any negative values. In this implementation, the layer's attributes - max_value, threshold value, and negative_slope - are all non-negative.

MaxPooling 2D Layer

The XceptionNet training architecture incorporates four MaxPooling 2D layers, which are essential for reducing the spatial dimensions of feature maps. These layers perform max pooling, a process that extracts the highest value from each window of a feature map, thereby decreasing computational load and mitigating overfitting. Each MaxPooling 2D layer is characterized by three key parameters: Pool Size, Strides, and Padding. The Pool Size, specified as an integer or a tuple of 2 integers, determines the dimensions of the pooling window. For example, a pool size of (3, 3) means the layer will select the maximum value from a 3x3 window.

Strides are a tuple of two integers, an integer, or None that determine the distance at which the pooling window moves for each stride. In this specific XceptionNet implementation, strides are set to (2, 2). Lastly, Padding adjusts the input boundaries, and in this network, it is set to (0, 0, 0, 0), indicating no padding is applied.

Input layer

The XceptionNet typically accepts input images of 299 × 299 pixels. This input layer consists of artificial neurons that receive the initial RGB image data, which is then processed by subsequent layers in the neural network.

Grouped convolution 2D layer

The Grouped Convolution 2D layer is the predominant layer in the XceptionNet deep learning model, occurring 34 times throughout the network. In this layer, the input channels are split into groups by a 2-D grouped convolutional layer, which then uses sliding convolutional filters. The grouped convolution layers are separated into two channels, referred to as Depth-wise Separable Convolution. This procedure involves applying a convolution filter to each input channel individually. Then, a point-wise convolution is used to combine the result of the depth-wise convolution with 1 × 1 convolutions in a linear manner.

Addition layer

The 'Addition layer' performs element-wise addition of inputs from several neural network layers. When constructing the layer, it is crucial to explicitly provide the desired amount of inputs. The layer's inputs are designated with specific names, totaling N inputs. All inputs to an addition layer possess identical size. The 'Addition layer' is the most frequently occurring layer in the XceptionNet deep learning model, appearing 12 times throughout the network

Output layer

Generally, the Output Layer has 1000 neurons (for the 1000 classes in the ImageNet dataset) with a SoftMax activation. In this case, the classification layer forms the designated output layer. The size of the input layer contains three flood image classes for classification: drowning vehicles, fallen poles, and water marker reading. The output size is two, and cross-entropy is used as the loss function.

SoftMax layer

The SoftMax function is utilized as an activation function in the XceptionNet deep learning model. The function normalizes the result by applying a transformation that converts it into a probability distribution, using the axis supplied. The function accepts an integer or a list of numbers as input to determine the axis, and the output preserves the same shape as the input.

Convolution 2D layer

The convolution layer forms the core of the XceptionNet deep learning model, appearing 40 times throughout the network architecture. This layer processes input data through three distinct phases: the Entry Flow, Middle Flow, and Exit Flow. The Entry Flow begins with three convolution layers, each employing 32 filters of size 3 × 3. Following this, the Middle Flow consists of 8 blocks, each containing a depth-wise separable convolution layer with 3 × 3 filters and a max pooling

layer. The network concludes with the Exit Flow, which incorporates two depth-wise separable convolution layers (also with 3 × 3 filters) and a global average pooling layer. Each Convolution 2D Layer is characterized by specific properties: a filter size of (3, 3), 64 filters, a stride of (2, 2), and a scale factor of (1, 1). The padding value is set to '0', with weights dimensioned as 3 × 3 × 3 × 64 and bias as 1 × 1 × 64. Additional weight parameters such as WeightLearnRateFactor, WeightL2Factor, and BiasLearnRateFactor are all set to 1, while BiasL2Factor is 0. The network utilizes the glorot method for weight initialization and zeros for bias initialization. This intricate structure allows the XceptionNet to effectively process and learn from complex image data, making it a powerful tool in deep learning applications.

Fully connected layer

The XceptionNet's training architecture includes two fully connected layers, each containing 4096 neurons with ReLU activation. The final layer automatically determines the input size and sets the output size to 2. Several parameters, including WeightLearnFactor, WeightL2Factor, and BiasLearnRateFactor, are set to 1, while BiasL2Factor is set to 0. The weights are initialized using the Glorot method, and biases are initialized to zeros.

Batch normalization

The Batch Normalization layer is the predominant layer in the XceptionNet deep learning model, occurring 40 times within the network. A Batch Normalization layer performs a normalization process on a small batch of data, treating each channel's observations independently. To expedite the training of the convolutional neural network and reduce its vulnerability to network initialization, it is recommended to include batch. The XceptionNet consists of 170 layers, including 22.8 million parameters. As shown in Table 1. The default input image size for the model is 299 × 299 × 3. Subsequently, it undergoes a continuous reduction process, gradually decreasing from 149 × 149 × 32. Finally, the classification output is compressed to 1× 1 s × 1000 c convolutions.

AlexNet architecture

AlexNet is a ground breaking convolutional neural network (CNN) widely recognized for its exceptional performance in image recognition and classification tasks. Among the components that make up Alex Net are five convolution layers, three max-pooling layers, two normalized layers, two fully linked layers, and one SoftMax layer. A convolution filter and a non-linear activation function known as "ReLU" are the components that make up each individual convolution layer. The pooling layers carry out the max-pooling function, and the presence of completely connected layers fixes the input size.

Convolution 2D layer

The AlexNet deep learning model mainly depends on convolution layers, incorporating a total

Table 1. Prototype of 170 Layers of XceptionNet Deep Convolution Networks

No.	Name	Type	Activations	Learnable properties
1	Input_1 299 × 299 × 3 images With rescale symmetric normalization	Image input	299(s) × 299(s) × 3(c) × 1(B)	-
2	Block1_conv1 32 3 × 3 × 3 convolution with stride [2 2] and padding [0 0 0 0]	Convolution	149(s) × 19(s) × 32(c) × 1B	Weights (3 × 3 × 3 × 32) Bias (1 × 1 × 32)
3	Block1_conv1_bn Batch Normalization with 32 channels	Batch normalization	149(s) × 19(s) × 32(c) × 1B	Offset (1 × 1 × 32) Scale (1 × 1 × 32)
4	Block1_conv1_act ReLU	ReLU	149(s) × 19(s) × 32(c) × 1B	-
5	Block1_conv2 64 3 × 3 × 32 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	147(s) × 147(s) × 64(c) × 1B	Weights (3 × 3 × 32 × 64) Bias (1 × 1 × 64)
|	|	|	|	|
168	Predictions 1000 fully connected layers	Fully connected	1(s) × 1(s) × 1000 (c) × 1B	Weights (1000 × 2048) Bias (1000 ×1)
169	Prediction_ SoftMax	SoftMax	1(s) × 1(s) × 1000 (c) × 1B	-
170	Classification Layer _Predictions	Classification output	1(s) × 1(s) × 1000 (c) × 1B	-

of 40 instances in its design. The network is organized with five separate convolutional layers, each tailored with appropriate parameters to analyze and extract characteristics from the input data. The initial convolutional layer utilizes 96 filters measuring 11 × 11, with a stride of 4 and ReLU activation. Next, the second layer is implemented with 256 filters of dimensions 5 × 5, a stride of 1, and ReLU activation. Both the third and fourth layers employ 384 filters of dimensions 3 × 3, with a stride of 1 and ReLU activation. The last convolutional layer consists of 256 filters, each having a size of 3 × 3 and a stride of 1. The Convolution 2D Layer is defined by specific properties, such as a filter size of (3, 3), 64 filters, a stride of (2, 2), and a dilation factor of (1, 1). The padding value is set to zero, with weights dimensioned as 3 by 3 by 3 by 64 and bias as 1 by 1 by 64. The weight parameters WeightLearnRateFactor, Weight-2Factor, and BiasLearnRateFactor are assigned a value of 1, while the BiasL2Factor is set to 0. The network utilizes the glorot approach for weight initialization and initializes biases with zeros. The complex arrangement of convolutional layers of AlexNet allows it to efficiently analyze sophisticated visual data, making it a formidable tool for jobs involving picture recognition and classification.

Cross channel normalization

A layer called 'channel-wise local response normalization' performs normalizing on each channel individually. This layer conducts local response normalization on each channel individually and allows the user to specify the size of the 'WindowChannelSize' parameter. The 'Cross Channel Normalization' layer used only two times in Alex Net.

Fully connected layer

The AlexNet architecture includes three fully connected layers:Two layers with 4096 neurons each, using ReLU activation and An output layer with 1000 neurons (matching ImageNet classes) and SoftMax activation.The input size is automatically determined, while the output size is set to three. Weight-related parameters (WeightLearn-Factor, WeightL2Factor, BiasLearnRateFactor) are set to 1, with BiasL2Factor at 0. Weights are initialized using the Glorot method, and biases are initialized to zeros.

Grouped convolution 2D layer

In the AlexNet deep learning model, the Grouped Convolution 2D layer is the most frequently used layer, appearing 34 times throughout the network. This layer divides the input channels into groups and applies sliding convolutional filters to each group. Grouped convolutional layers are ideal for channel-wise separable convolutions

Output layer

Classification Output Layer has1000 neurons (for the 1000 classes in the ImageNet dataset) with a SoftMax activation. The output layer is a feature map forms the designated size of input image with the filters applied. As opposed to the fact that the output size is two and cross-entropy is utilized as a loss function.

Input layer

The input layer of the neural network accepts an RGB image with dimensions of 227 × 227 pixels. This layer consists of artificial neurons that receive the initial image data, which is then processed by subsequent layers in the network.

ReLU layer

The AlexNet deep learning model features the ReLU (Rectified Linear Unit) layer seven times throughout its architecture. This layer applies a simple threshold operation: it sets any input value below zero to zero, while leaving positive values unchanged. In AlexNet's implementation, the ReLU layer's properties – max_value, threshold value, and negative_slope – are all non-negative. This configuration helps prevent vanishing gradients, enabling more efficient learning in the network.

Max pooling 2D layer

The Alex Net training network consists of four Max Pooling 2D layers. The Max Pooling 2D layer typically comprises three components: Pool Size, Strides, and Padding. When training, specify an integer or a tuple of three integers to determine the window size for calculating the maximum value. The coordinates (3, 3) will determine the maximum value within a pooling window of size 3 × 3. The possible values for the variable are an Integer, which is a tuple of 2 integers, or None. Values of strides. The pooling stride is set to (2, 2) and the padding is set to (0, 0, 0, 0), which determines the movement of the pooling window for each pooling step.

SoftMax layer

The AlexNet model uses the SoftMax function once as an activation layer. This function normalizes its input along a specified axis, producing a probability distribution while maintaining the input's shape. It's typically applied to the final layer for multi-class classification tasks.

Dropout layer

The 'dropout layer' is a layer that applies a random opportunity rate of 0.5 to the input objects and sets them to zero. This layer is a syntax that can be used to specify the non-obligatory Name and opportunity belongings by way of using a name-price pair and any of the arguments outlined inside the preceding syntaxes. In the deep getting to know model that Alex Net uses, the dropout layer is the maximum distinguished layer, and it appears times across the community. The Alex Net consists of 25 layers. As shown in Table 2. The default input image size for the model is 277 × 277 × 3 RGM image. Subsequently, it undergoes a continuous reduction process, gradually decreasing from 55 × 55 × 96. Finally, the classification output is compressed to 1 × 1 s × 1000 c convolutions.

EXPERIMENTATION

The Experimentation of Urban Flood Detection Dataset Training on both 'Alex Net' and 'XceptionNet' pre-trained deep neural networks that have been proposed as excellent models for identifying and classifying flood images. These models are highly regarded in the field of deep learning for their exceptional accuracy in classifying the 1000 natural images of ImageNet. For training purposes, a data set of 100 images is constructed to train the classifier on flood images. The functionality of both network designer architectures is illustrated in Figures 7 and 8.

Flood object dataset classification

The Flood Image dataset trains 3 primary classes of pictures for the cause of identity and classification of detection. The schooling statistics consists of 110 photographs for each class, specifically Cars, Electricity-Poles, and Watermarker. The photo shows a screenshot of the flood facts set used for training. Figure 8 displays two types of training consequences: Training_Validation_Accuracy and Training_Data_Loss functions. The training dataset employs multi-layer deep convolutional networks including of 5 pooling layers, 10 convolutional layers, and 10 ReLU-convolutional layers. The system utilizes three widely-used training optimizer algorithms: Root Mean Square Propagation (RMSProp), Adaptive Moment Estimation (ADAM) optimizer and Stochastic Gradient Descent with Momentum (SGDM) optimizer. We conducted an experiment

Table 2. Prototype of 25 Layers of AlexNet Deep Convolution Networks

No.	Name	Type	Activations	Learnable properties
1	Data 227 × 227 × 3 images with 'zero center' normalization	Image Input	227(s) × 227(s) × 3(c) × 1(B)	-
2	conv1 96 11 × 11 × 3 convolution with stride [4 4] and padding [0 0 0 0]	Convolution	55(s) × 55(s) × 96(c) × 1(B)	Weights (11 × 11 × 3 × 96) Bias (1 × 1 × 96)
3	ReLU	ReLU	55(s) × 55(s) × 96 (c) × 1(B)	-
4	Norm1 Cross channel normalization with 5 channels	Cross channel normalization.	55(s) × 55(s) × 96 (c) × 1(B)	-
5	Pool1 3 × 3 max pooling with sti de [2 2] and padding[0 0 0 0]	Max polling	27(s) × 27(s) × 96(c) × 1(B)	1 × 1 × 64
|	|	|	|	|
23	fc8 1000 fully connected layers	Fully connected	1(s) × 1(s) × 1000(c) × 1(B)	Weighs: 1000 × 40 Bias: 1000 × 1
24	Prob softmax	Softmax	1(s) × 1(s) × 1000(c) × 1(B)	-
25	Output Crossentropyex with 'Tench' and 999 other classes	Classification output	1(s) × 1(s) × 1000(c) × 1(B)	-

about the number of epochs varied between three algorithms to optimize accuracy for flood dataset.

All the other training parameters were fixed: we used longest sequence length typically 100, an initial learning rate of.01 which was halved after each epoch if no improvement in validation loss was seen for last three epochs, minibatch size of 128 and zero padding on right. Additional constants were L2 regularization = 0.01, drop factor of learning rate = 0.1 applied for each (10) epochs, decay factor for squared gradient = .9 (epsilon was set to - le-08). We used the L2 norm technique for gradient thresholding which calculates the absolute distance of coordinates from vector space origin. This approach allowed us to systematically assess the impact of epoch variation on model performance while controlling for other variables.

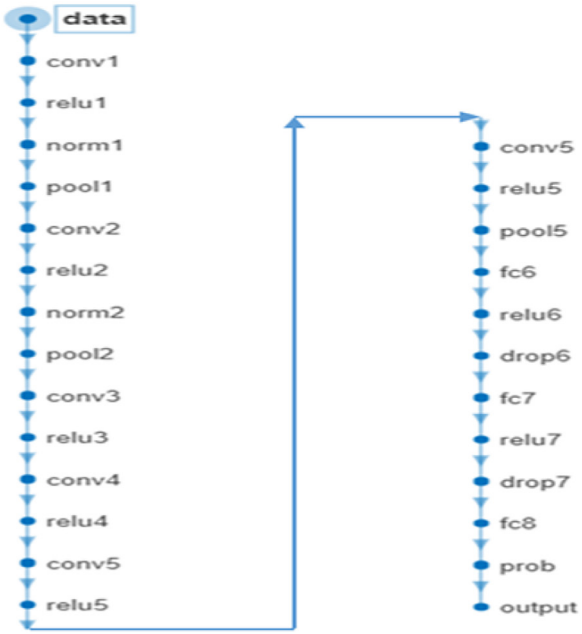

Figure 8. Training layered architecture of XceptionNet

Hyper parameter tuning

The Hyperparameters include the Learning rate, Weight decay, Momentum, and Batch size. The learning rate determines how quickly the network adjusts its weights during training, while weight decay controls the regularization of the network. Momentum helps the network avoid local minima during training, and batch size determines the number of samples used in each training iteration. The experimentation was done

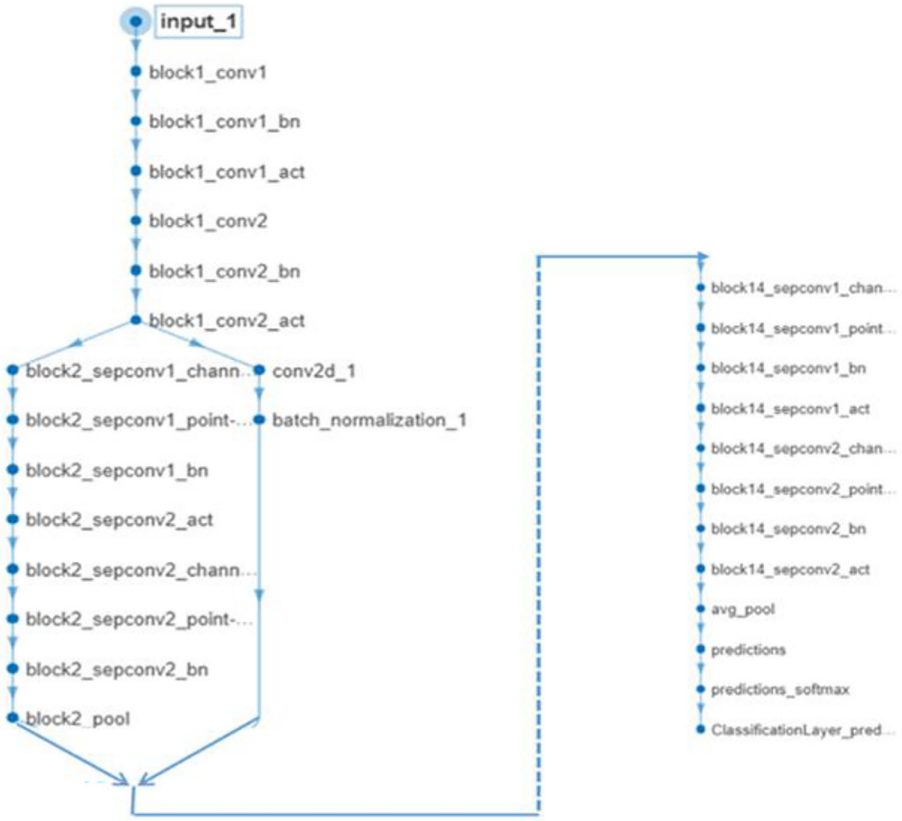

Figure 7. Training layered architecture of XceptionNet

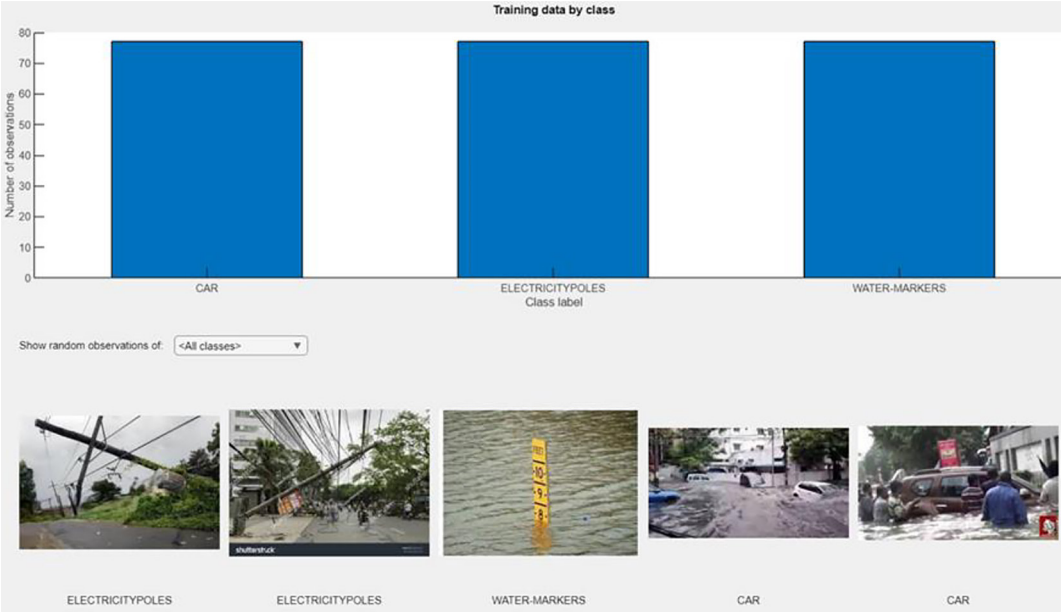

Figure 9. Flood object custom dataset with three classes (water-marker, electricity pole, and car)

based on turning the hyper parameters, which has shown greater impact on data sets. Key hyper parameters used:

- basic – determines how often the validation performance is evaluated during the training process. The "Basic" training options typically include the following parameters: Max Epochs: This sets the maximum number of training iterations. Mini-Batch Size: This specifies the number of training examples to include in each mini-batch used for computing gradients and updating weights. Validation Frequency: As discussed earlier, this parameter determines how often the validation performance is evaluated during the training process. Execution Environment: This option allows to specify the hardware resources and computational environment to be used for training the neural network. The Execution Environment options typically include the following choices. CPU: This is the default setting and is suitable for small to moderate-sized neural networks or when GPU resources are not available. GPU: This option allows you to utilize the computational power of a graphics processing unit (GPU) for training the neural network. Parallel Pool: This option enables parallel computing by distributing the training workload across multiple CPU cores or GPUs (if available). Cloud: This option allows you to offload the training process to cloud computing resources, such as MATLAB Cloud Services or third-party cloud platforms.
- solver – this is considered to specify the training algorithm or optimizer used for updating the network weights during the training process. This optimizer has different solver options available: SGDM (SGD with Momentum): This solver combines the SGD algorithm with a momentum term. The momentum term accumulates the gradients of previous iterations, helping to accelerate the training process and potentially escape local minima. RMSProp: This is an adaptive learning rate optimization algorithm It adapts the learning rate for each weight based on the magnitude of recent gradients, which can help mitigate the problem of vanishing or exploding gradients. ADAM: The Adam (Adaptive Moment Estimation) optimizer is one of the most popular and effective optimization algorithms for deep learning. It combines the ideas of momentum and RMSProp, adapting the learning rate for each weight based on both the moving average of the gradients and the moving average of the squared gradients. Initial Learn Rate: The learning rate determines the step size during the weight update process. It is generally recommended to start with a relatively small learning rate (0.001 or 0.01) and gradually increase it if the training process is progressing too slowly.
- advanced – this option is used to configure advanced parameters and settings for the neural network training process. Here are some of the common "Advanced" training options: L2

Regularization: This option enables to prevent overfitting (also known as weight decay). It adds a penalty term to the loss function, which shrinks the weights towards zero, encouraging the network to learn simpler and more generalizable representations. Gradient Threshold Method: Implements gradient clipping to prevent excessively large gradients, which can cause numerical instability or exploding gradients, Options include 'l2norm' (clip gradients by teir L2 norm) and 'absmax' (clip gradients by their absolute maximum value). Gradient Threshold: This parameter sets the threshold value used for clipping the gradients when using the selected gradient threshold method. Validation Patience: This helps prevent excessive training and overfitting once the validation performance plateaus or starts to degrade. Shuffle: This option determines whether the training data should be shuffled before each epoch during training.

- checkpoint path: this option allows you to specify a file path where checkpoint files will be saved during the training process with current state of the neural network, including the weights, biases, and other relevant information.
- sequence – this specifies how the training data should be organized and presented to the neural network during work out. Sequence Length: It represents the length of the input sequences using this option. The sequence length can be set to a fixed value or set to "longest" to automatically determine the maximum sequence length from the training data. Sequence Padding Direction: determines how the input sequences should be padded for different lengths. The available options are "left" (pad at the beginning of the sequence) or "right" (pad at the end of the sequence). Sequence Padding Value: When padding input sequences are set to a fixed length, this option specifies the value to be used for padding. The default padding value is typically 0, but you can set it to a different value depending on your data and problem requirement.
- checkpoint frequency – this setting determines how often the number of iterations (epochs or iterations) are saved during the training process in checkpoint files. Checkpoint Frequency Unit: This option specifies the unit of measurement for the checkpoint frequency. Learning Rate Schedule: This option allows you to specify a schedule for adjusting the learning rate during training. Common schedules include step decay (reducing the learning rate by a fixed factor at specific intervals), exponential decay (gradually reducing the learning rate over time), and piecewise learning rate (manually specifying different learning rates for different epochs). Learn Rate Drop Factor: It determines the multiplicative factor by which the learning rate is reduced at each step.

RESULTS

Performance analysis of XceptionNet

The training results, as detailed in Table 3, demonstrate the effectiveness of the XceptionNet architecture in achieving 96% accuracy on the training data using our specified parameters. This performance varied depending on the choice of optimizer algorithm and the number of epochs. Interestingly, we observed that this peak accuracy was reached using three different optimizers: ADAM, RMSPROP, and SGDM. In contrast, when applied to the ALEXNET architecture on the same Flood dataset, these optimizers struggled, managing only a maximum training accuracy of 50%. Further analysis revealed that with XceptionNet, the SGDM optimizer consistently achieved 96% accuracy across a range of epoch values (20, 40, 60, 80, and 100), while RMSPROP required 100 epochs to reach the same benchmark. The ADAM optimizer showed a unique pattern, with lower accuracy at both low and high epoch counts, peaking only in the mid-range. Figures 11 and 12 visually represent these findings for SGDM and ADAM optimizers. Notably, SGDM emerged as the top performer, reaching the 96% accuracy mark in just 20 epochs. This performance variability across different epoch ranges and optimizers is clearly illustrated in Figures 10–12, providing valuable insights into the optimization process for this particular flood dataset and model architecture.

Performance analysis of AlexNet

The training of AlexNet using the flood object dataset was unsuccessful. Taking into account the earlier factors, which change the training accuracy depending on the optimizer

method and epochs, the AlexNet architecture achieved 96% accuracy with the training data (Table 3). The training data set achieved an accuracy of 96% when simulating using the ADAM optimizer and SDGM. Due to AlexNet's 50% maximum training accuracy, the SGDM, RMSPROP, and ADAM optimizers yielded subpar results on the Flood dataset. At 20 Max_Epochs, though, both the SGDM and ADAM optimizers achieved 96% accuracy. Figures 13–15 display the outcomes of the education accuracy tests conducted on the SDGM, RMSPROP, and ADAM optimizers. With a training accuracy of 96% after 20 epochs, SDGM outperformed ADAM and RMSPROP. Figures 13–15 demonstrate that ADAM accuracy was lowest at lower and higher epochs and highest at middle-range epochs.

Performance comparison between XceptionNet and AlexNet

The training performance of XceptionNet on the flood object dataset is comparatively higher than that of AlexNet in all three optimizers (Figure 16). In particular, the performance of XceptionNet reaches its peak in SGDM, with ADAM and RMSProp following closely behind (Table 3).

All three optimizers performed poorly with AlexNet. In most cases, the resultant accuracy of AlexNet remains consistently stable and low, regardless of the number of epochs, while the performance of XceptionNet exhibits varying accuracies across the three optimizers. We have observed that the SDGM optimizer's training accuracy for both deep learning training architectures remains constant. However, when we vary the

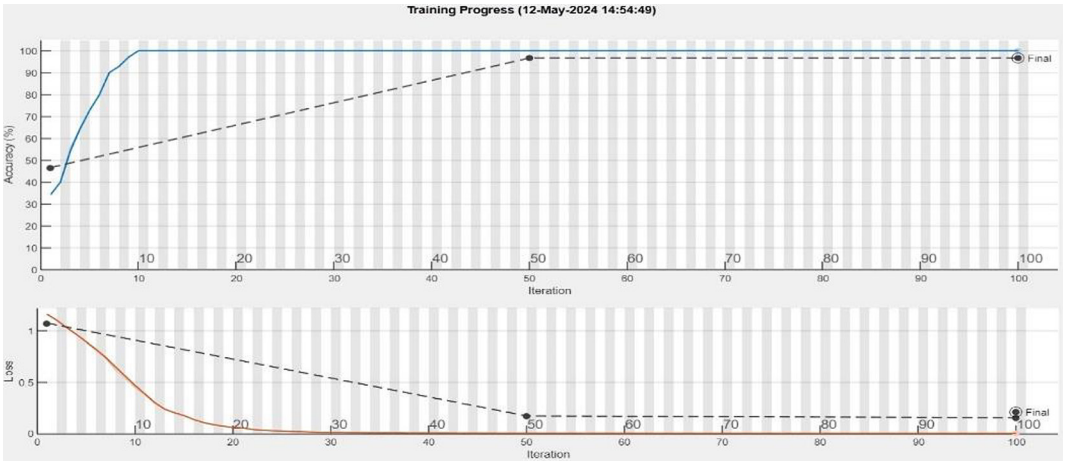

Figure 10. Loss and traning accuracy line plot of SGDM Optimizer on XceptionNet

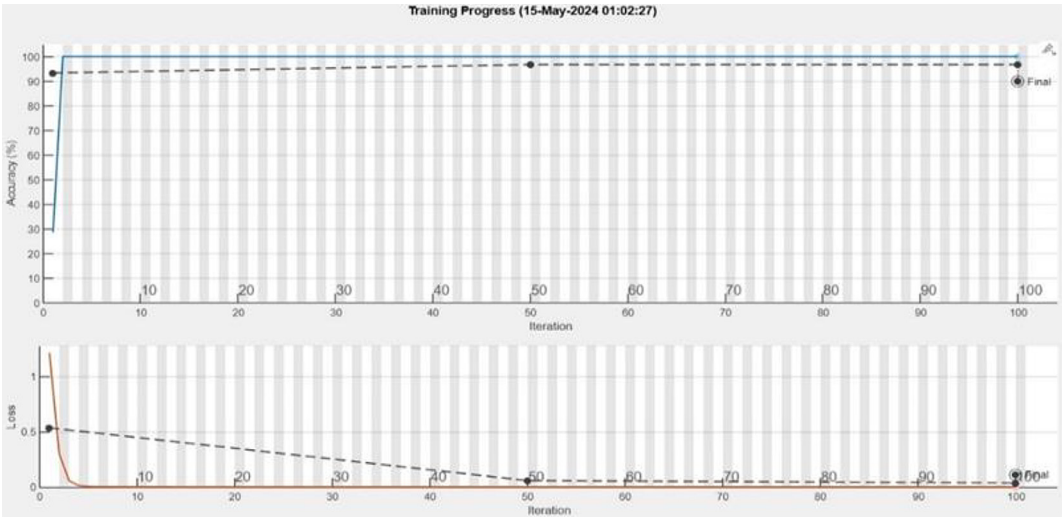

Figure 11. Loss and training accuracy line plot of ADAM Optimizer on XceptionNet

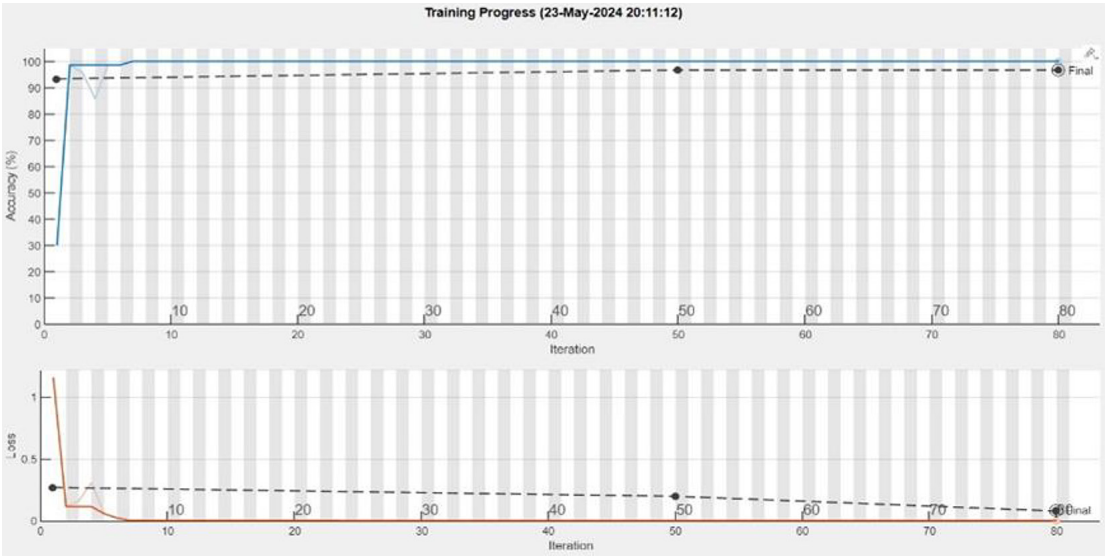

Figure 12. Loss and traning accuracy line plot of RMSPROP Optimizer on XceptionNet

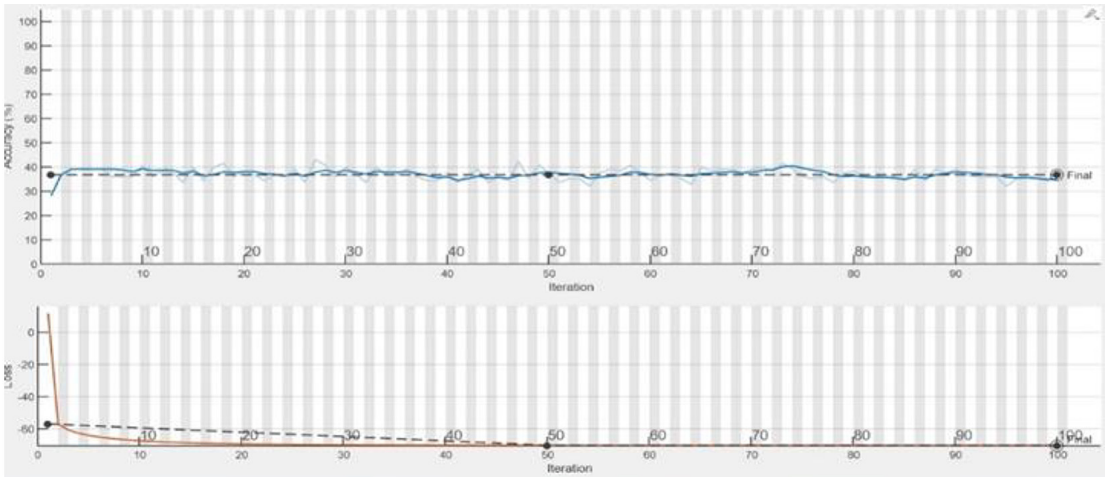

Figure 13. Loss and training accuracy line plot of SGDM Optimizer using AlexNet

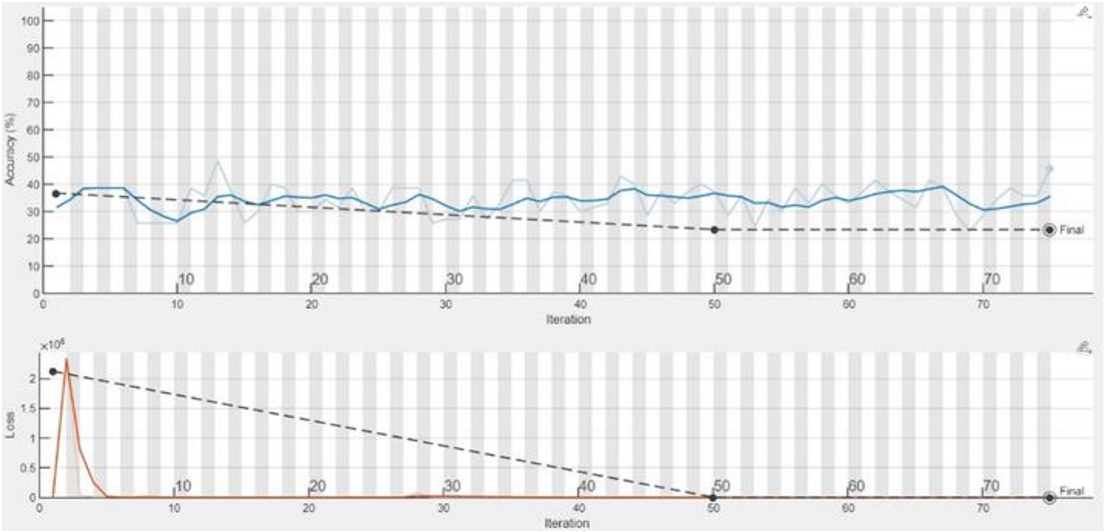

Figure 14. Loss and traning accuracy line plot of ADAM Optimizer using Alex Net

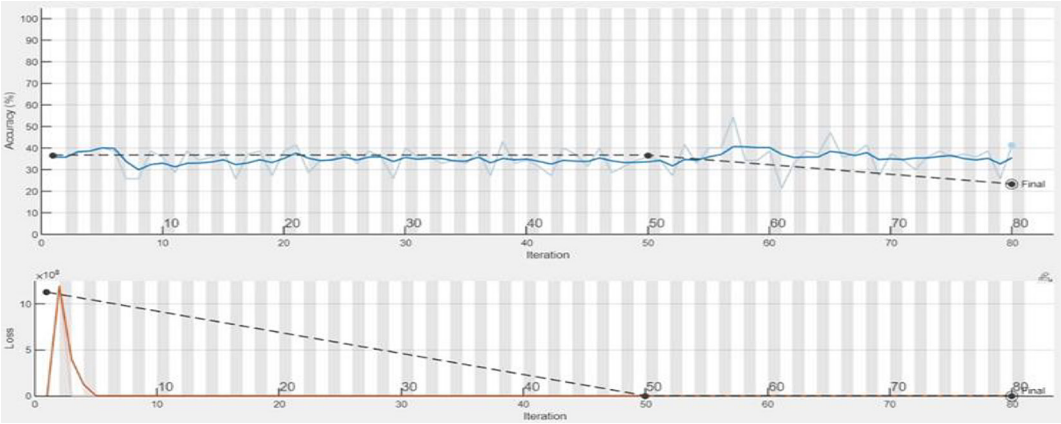

Figure 15. Loss and traning accuracy line plot of RMSPROP Optimizer using Alex Net

epochs from 20 to 100 while keeping other tuning parameters like learning rate and batch size constant, the performance of the ADAM and RMSPROP optimizers significantly changes. rate, batch size, etc. The AlexNet dataset's training is faster than that of XceptionNet, with XceptionNet requiring nearly five times more resources to train the current custom dataset. Even AlexNet is very fast in training datasets compared with XceptionNet, but the average dataset training performance is 50% lower when using SGDM, followed by ADAM with 46% and RMSPROP with 42%. XceptionNet showed a significant performance advantage over AlexNet due to the use of depth wise separable convolutions, which reduce the number of parameters and computational cost.

The custom flood object dataset comprises 65421 annotated images classified into three distinct classes. This experiment revealed that AlexNet is not suitable for larger datasets, while XceptionNet could potentially consider training larger datasets. All three optimizers achieve a peak accuracy of 97.47% for XceptionNet, while the ADAM optimizer achieves the highest accuracy of 83.33% for AlexNet. The rest of the two optimizers recorded a very low accuracy that ranges from 23.33% to 40% for AlexNet. We also observe a significant improvement in accuracy in XceptionNet, but AlexNet's behavior is not stable, and it has begun to decline after reaching its peak accuracy. XceptionNet's shortfall lies in its larger computation time, making it unsuitable for smaller datasets due to execution time constraints.

Table 3. Training data performance of XceptionNet and AlexNet

Optimizer algorithm	Max_Epochs	XceptionNet	AlexNet
		Accuracy (%)	Accuracy (%)
SGDM	20	96.47	40
	40	96.47	40
	60	96.47	40
	80	96.47	40
	100	96.47	40
ADAM	20	63.33	36.67
	40	73.33	23.33
	60	80	40
	80	66.67	83.33
	100	96.47	36.67
RMSProp	20	40.25	40
	40	33.33	36.67
	60	43.33	40
	80	96.47	23.33
	100	96.47	36.67

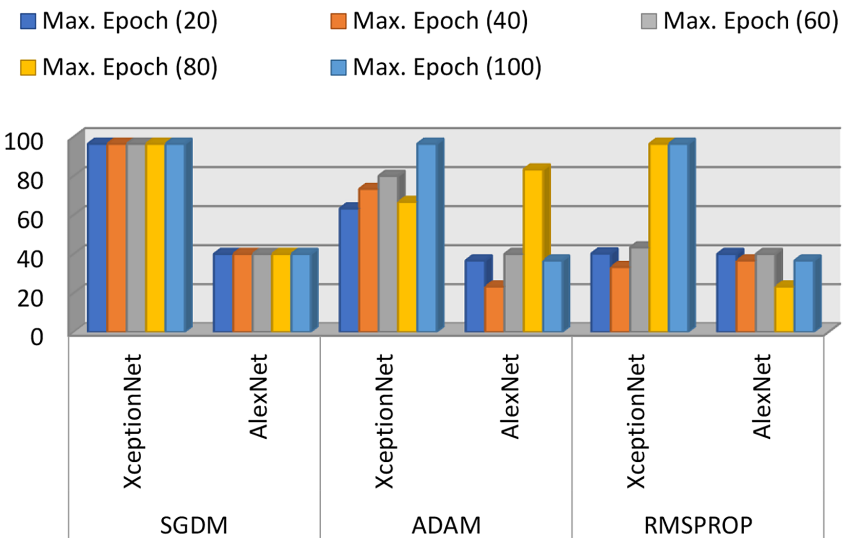

Figure 16. Training performance of SDGM, RMSPROP and ADAM Optimizer on AlexNet and XceptionNet

DISCUSSION

The current study compares the performance of Google Net, AlexNet, and Squeeze Net with earlier research, specifically the work of Khan et al. (2021) [Ignatowicz, et al.,2024]. They were able to attain accuracy rates of 94.99%, 94.61%, and 94.09%, respectively, by adjusting hyper-parameters with the help of electroencephalogram (EEG) data. The results of the detection analysis reveal that AlexNet surpasses both Google Net and SqueezeNet in terms of performance ability. On the other hand, they focused solely on the epochs and the learning rate, completely ignoring the importance of optimizers. In this particular study, the hyper-parameters are only fine-tuned, and researchers additionally investigate the impact that optimizers have on the accuracy of training [Kalantar et al.,2021]. Ullah et al. (2022) explicitly evaluated the performance of AlexNet, ResNet18, and Squeeze Net using a dataset that included 4333 photographs classified into eight different types of road cracks.

While conducting this experiment, the training and testing photos stayed the same throughout the entire epoch and iteration. The choice of an optimizer was not the primary focus of the investigation. ResNet 18 can only achieve an accuracy of 85.2%. We investigated the AlexNet and XceptionNet models in the suggested method, focusing primarily on addressing training issues. We were able to accomplish this by fine-tuning hyper-parameters, with a particular emphasis on optimizers such as ADAM, SGDM, and RMSProp [Kumar et al.,2023]. Ashhar et al. (2021) primarily focused their study on assessing the accuracy of various deep learning models, such as GoogleNet, SqueezeNet, DenseNet, ShuffleNet, and MobileNetV2, in categorizing lung cancers observed on a CT scan.

With the help of the GoogleNet model, they were able to achieve an accuracy of 94.53%. [Kumar et al., 2023]. The research that they conducted did not take into account AlexNet and XceptionNet, which largely focused on validation accuracy. In their study, Dahiya et al. (2022) focused on training accuracy and utilized the Plant Village dataset, which consisted of 20,640 photos representing 15 different classes and three different species: tomato, pepper, and potato. Using this dataset, they applied eight distinct deep learning architectures, which are as follows: AlexNet, GoogleNet, MobileNet, ResNet 18, ResNet 50, ResNet 101, ShuffleNet, and SqueezeNet. They utilized epochs, learning rate, small batch size, and optimizer as hyperparameters; however, they did not discuss the performance of XceptionNet.

They utilize a range of epochs from 30 to 50, applying only the ADAM and SGDM optimizers. The RMSProp optimizer is not utilized. However, the current work modifies the epochs from 20 to 100. GoogleNet is the only one of the eight deep learning architectures that displays improved performance when it comes to reliably detecting larger datasets. They can only work on a maximum of three epochs and support a maximum of two optimizers. The current study applied three different types of optimizers—SGDM, ADAM, and RMSProp—to six different epochs, ranging from twenty to one hundred. In addition to using vector

distance algorithms, we adjusted the batch size to 32 elements. To maintain simplicity, we limit the current analysis to three specific pre-trained models. On the other hand, it is feasible to further train the existing dataset by employing additional deep learning models such as EfficeintNet, AlexNet, VGG16, DarkNet, PANet, ShuffleNet, NasNet XceptionNet, MobileNet-v2, and a number of other models. Consequently, this will enable the evaluation of the efficiency of the provided outcomes. Researchers with balanced smaller datasets can benefit from the current study as it enables them to achieve higher accuracy by effectively adjusting the hyper-parameters. Most of the deep learning detection mechanisms is on medical and agriculture datasets. The novelty of this study is in the identification of objects harmed by floods in metropolitan areas using a proprietary dataset of flood-affected imagery. Utilizing flood-affected imagery files enables rescue personnel to promptly warn and execute rescue operations.

CONCLUSIONS

The current investigation has fulfilled the study's objectives. The study has identified the most effective deep-training network for successfully training flood object data. Additionally, the study has demonstrated to researchers that the SGDM optimiser is the most effective method for training flood object datasets, resulting in higher detection accuracies. The training accuracy results specifically indicate that lower epoch rates result in lower accuracies for the flood object dataset, and that a minimum of median epoch rates, such as 60 to 100, is necessary to achieve a higher learning rate. The study also recommends against using ADAM and RMSProp Optimizer for training flood object datasets. The overall observation leads to the conclusion that using XceptionNet in conjunction with the SGDM optimizer, with epochs ranging from 60 to 100, will result in higher training accuracy for flood object datasets. Furthermore, these studies assist researchers in maintaining recommended epochs for training flood object datasets, as choosing the appropriate optimizer can reduce training time and potentially achieve higher train and test accuracies. The current research serves as a useful tool for training flood objects that inundate during flash urban floods caused by cloud bursts, which are crucial for current climate change scenarios. The present research only concentrates on addressing training challenges associated with flood object datasets. Furthermore, it is imperative to subject the research to accuracy evaluations utilizing detector algorithms such as YOLO, MobileNet, and RCNN.

Acknowledgements

Authors acknowledge their respective institutions for providing support and facilities.

REERENCES

1. Agonafir C., Lakhankar T., Khanbilvardi R., Krakauer N., Radell D., Devineni N. 2023. A review of recent advances in urban flood research., Water Security, 19, 100141, https://doi.org/10.1016/j.wasec.2023.100141
2. Ahmed I., Ahmad M., Jeon, G. and Chehri A. 2023. An Internet-of-Things and AI-Powered Framework for Long-Term Flood Risk Evaluation. IEEE Internet of Things Journal. 11(3), 3812–3819. https://doi.org/10.1109/JIOT.2023.3308564
3. Alabbad Y., Demir I. 2022. Comprehensive flood vulnerability analysis in urban communities: Iowa case study, International Journal of Disaster Risk Reduction, 74, 102955. https://doi.org/10.1016/j.ijdrr.2022.102955
4. Angloher G., Banik S., Bartolot D., Benato G., Bento A., Bertolini A., Breier R., Bucci C., Burkhart J., Canonica L. and D'Addabbo A. 2023. Towards an automated data cleaning with deep learning in CRESST. The European Physical Journal Plus, 138(1), 1–11. https://doi.org/10.1140/epjp/s13360-023-03674-2
5. Anil H., Pawan B. 2022. a review on urban flood management techniques for smart city and future research, Springer Charm, International conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI 2022) proceedings, 3, 319–336. https://doi.org/10.1007/978-3-031-18497-0_23
6. Avand M., Moradi H.R., Ramazanzadeh Lasboyee M. 2021. Spatial Prediction of Future Flood Risk: An Approach to the Effects of Climate Change, Geosciences, 11(1), 25. https://doi.org/10.3390/geosciences11010025
7. Baghermanesh S.S., Jabari S., McGrath, H. 2021. Urban flood detection using sentinel1-a images. 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, 527–530. https://doi.org/10.1109/IGARSS47720.2021.9554283
8. Bagyaraj M., Senapathi V., Chung S.Y., Gopalakrishnan G., Xiao Y., Karthikeyan S., Nadiri A.A., Barzegar R. 2023. A geospatial approach for assessing urban flood risk zones in Chennai, Tamil Nadu, India. Environmental Science and Pollution

Research, 30(45), 100562–100575. https://doi.org/10.1007/s11356-023-29132-1

9. Bakhtiari V., Piadeh F., Behzadian K., Kapelan Z. 2023. Dealing with urban floods within a resilience framework regarding disaster stages. Habitat International, 136, 102783. https://doi.org/10.1016/j.scs.2023.104958
10. Bentivoglio R., Isufi E., Jonkman S.N., Taormina R. 2022. Deep learning methods for flood mapping: a review of existing applications and future research directions. Hydrology and earth system sciences, 26(16), 4345–4378, https://doi.org/10.5194/hess-26-4345-2022
11. Braveen M., Anusha K., Jerlin M.A., Seetha R., Sampath N. 2023. IDISense: IoT-based dam water disaster sensing and prevention system. IEEE Sensors Journal, 23(23), 29451–29457. https://doi.org/10.1109/JSEN.2023.3322290
12. Chen J., Li Y., Zhang C., Tian Y., Guo Z. 2023. Urban flooding prediction method based on the combination of LSTM neural network and numerical model. International Journal of Environmental Research and Public Health, 20(2), 1043. https://doi.org/10.3390/ijerph20021043
13. Chen Z., Duan J., Kang L., Qiu G. 2021. Class-imbalanced deep learning via a class-balanced ensemble. IEEE transactions on neural networks and learning systems, 33(10), 5626–5640. https://doi.org/10.1109/TNNLS.2021.3071122
14. Deccan Herland. 2023. FIR filed after Bengaluru techie drowns in flooded underpass. (Accessed on 15 January, 2024) (https://www.deccanherald.com/india/karnataka/bengaluru/fir-filed-after-bengaluru-techie-drowns-in-flooded-underpass-1220935.html).
15. Esfandiari M., Jabari S., McGrath H., Coleman D. 2020. Flood mapping using random forest and identifying the essential conditioning factors; a case study in Fredericton New Brunswick Canada, ISPRS Annals of Photogrammetry Remote Sensing Spatial Information Sciences, 3(3), 609–615. https://doi.org/10.5194/isprs-annals-V-3-2020-609-2020
16. Faruq A., Arsa H.P., Hussein S.F., Razali C.M., Marto A., Abdullah S.S. 2020. Deep learning-based forecast and warning of floods in Klang River, Malaysia. Ingénierie des Systems d Inf., 25(3), 365–370, https://doi.org/10.18280/isi.250311
17. Fazel-Rastgar F., Sivakumar V. 2023. A case study of an extreme flooding episode in Charikar, Eastern Afghanistan, Journal of Water and Climate Change, 14(12), 4689–4707, https://doi.org/10.2166/wcc.2023.462
18. Gaffinet B., Hagensieker R., Loi L. and Schumann G. 2023. Supervised machine learning for flood extent detection with optical satellite data, IGARSS 2023 - 2023 IEEE International Geoscience and Remote Sensing Symposium, Pasadena, CA, USA, 2084–2087, https://doi.org/10.1109/IGARSS52108.2023.10282274
19. Ghasemi P., Karimian N. 2020. A qualitative study of various aspects of the application of IoT in disaster management. 6th International Conference on Web Research (ICWR), 77–83. IEEE. https://doi.org/10.1109/ICWR49608.2020.9122323
20. Goyal H.R., Sharma S. 2023. Flood management system using cloud computing and internet-of-things, 2023 5th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), Navi Mumbai, India. 1–6, https://doi.org/10.1109/ICNTE56631.2023.10146661
21. Gude V., Corns S., Long S. 2020. Flood prediction and uncertainty estimation using deep learning. Water, 12(3), 884. https://doi.org/10.3390/w12030884
22. Gupta L., Dixit J. 2023. Assessment of urban flood susceptibility and role of urban green space (UGS) on flooding susceptibility using GIS-based probabilistic models. Environmental Monitoring and Assessment, 195(12), 1518, https://doi.org/10.1007/s10661-023-12061-4
23. Harshasimha A.C., Bhatt C.M. 2023. Flood vulnerability mapping using maxent machine learning and Analytical Hierarchy Process (AHP) of Kamrup Metropolitan District, Assam. Environ., Sci. Proc. 2023, 25(1), 73. https://doi.org/10.3390/ECWS-7-14301
24. Hashi A.O., Abdirahman A.A., Elmi M.A., Hashi S.Z., Rodriguez O.E. 2021. A real-time flood detection system based on machine learning algorithms with emphasis on deep learning. International Journal of Engineering Trends and Technology, 69(5), 249–256. https://doi.org/10.14445/22315381/IJETT-V69I5P232
25. Ignatowicz J., Kutt K., Nalepa G.J. 2024. Evaluation and Comparison of Emotionally Evocative Image Augmentation Methods. 28th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2024), Procedia Computer Science, Elsevier. https://arxiv.org/pdf/2406.16187
26. Kalantar B., Ueda N., Saeidi V., Janizadeh S., Shabani F., Ahmadi K., Shabani F. 2021. Deep neural network utilizing remote sensing datasets for flood hazard susceptibility mapping in Brisbane, Australia. Remote Sensing, 13(13), 2638. https://doi.org/10.3390/rs13132638
27. Kumar V., Azamathulla H.M., Sharma K.V., Mehta D.J., Maharaj K.T. 2023. The state of the art in deep learning applications, challenges, and future prospects: A comprehensive review of flood forecasting and management, Sustainability, 15(13), 10543. https://doi.org/10.3390/su151310543
28. Lei X., Chen W., Panahi M., Falah F., Rahmati O., Uuemaa E., Kalantari Z., Ferreira C.S., Rezaie F., Tiefenbacher J.P., Lee S. 2021. Urban flood modeling using deep-learning approaches in Seoul, South Korea. Journal of Hydrology, 601, 126684, https://doi.org/10.1016/j.jhydrol.2021.126684
29. Putranto M.F. and Munir R. 2023. Deep

Learning Approach for Heavy Rainfall Prediction Using Himawari-8 And RDCA Data, International Conference on Computer, Control, Informatics and its Applications (IC3INA), Bandung, Indonesia, 424–429. https://doi.org/10.1109/IC3INA60834.2023.10285744

30. Maurya J., Pant H., Dwivedi S., Jaiswal M. 2021, Flood avoidance using IOT. International Journal of Engineering Applied Sciences and Technology, 6(1), 155–158. https://ijeast.com/papers/155-158,Tesma601,IJEAST.pdf
31. Miau S., Hung W.H. 2020. River flooding forecasting and anomaly detection based on deep learning. IEEE Access, 8, 198384–198402. https://doi: 10.1109/ACCESS.2020.3034875
32. Munawar H.S., Ullah F., Qayyum S., Heravi A. 2021. Application of deep learning on uav-based aerial images for flood detection. Smart Cities, 4(3), 1220–1242, https://doi.org/10.3390/smartcities4030065
33. Patil S., Sawant S., Joshi A. 2023. Flood detection using remote sensing and deep learning approaches, 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Delhi, India, 1–6, https://doi.org/10.1109/ICCCNT56998.2023.10306978
34. Prathyusha K., Raju A.J., Rao P.J. 2024. Novel Malaria Risk Prediction and Mapping of Integrated Tribal Development Agency, Paderu Region, India, Using SAMRR. Journal of the Indian Society of Remote Sensing, 52(1),167–187. https://doi.org/10.1007/s12524-023-01796-9
35. Ramachandraiah C. 2011. Coping with urban flooding: a study of the 2009 Kurnool floods, India. Environment and Urbanization, 23(2), 431–46. https://doi/10.1177/0956247811418733
36. Rao G.N., Rao P.J., Duvvuru R., Beulah K., Lydia E.L., Rathnala P., Balakrishna B., Motru V.R. 2024. Neural fuzzy system design in forest fire detection. Microsystem Technologies, 30(4), 455–467. https://doi.org/10.1007/s00542-023-05496-9
37. Rauschmayr N., Kama S., Kim M., Choi M., Kenthapadi K. 2022. Profiling deep learning workloads at scale using amazon sagemaker. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 3801–3809. https://doi.org/10.1145/3534678.3539036
38. Roy P., Pal S.C., Chakrabortty R., Chowdhuri I., Malik S., Das B. 2020. Threats of climate and land use change on future flood susceptibility. Journal of Cleaner Production, 272, 122757, https://doi.org/10.1016/j.jclepro.2020.122757
39. Samikwa E., Voigt T., Eriksson J. 2020. Flood Prediction Using IoT and Artificial Neural Networks with Edge Computing, 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom), IEEE Smart Data(SmartData), 234–240, https://doi.org/10.1109/iThings-GreenCom-CPSComSmartData-Cybermatics50389.2020.00053
40. Sankaranarayanan S., Prabhakar M., Satish S., Jain P., Ramprasad A., Krishnan A. 2020. Flood prediction based on weather parameters using deep learning. Journal of Water and Climate Change, 11(4), 1766–1783. https://doi.org/10.2166/wcc.2019.321
41. Seidel D., Wurster S., Jenks J.D., Sati H., Gangneux J.P., Egger M., Alastruey-Izquierdo A., Ford N.P., Chowdhary A., Sprute R., Cornely O. 2024. Impact of climate change and natural disasters on fungal infections. The Lancet Microbe. 5(6), e594-e605. https://doi.org/10.1016/S2666-5247(24)00039-9
42. Sharma V.K., Azad R.K., Chowdary V.M., Jha C.S. 2022. Delineation of Frequently Flooded Areas Using Remote Sensing: A Case Study in Part of Indo-Gangetic Basin, Geospatial Technologies for Land and Water Resources Management, Springer, Cham, 103, 505–530. https://doi.org/10.1007/978-3-030-90479-1_27
43. Shreekumar S., Madhu D., Akella A.K. 2021. Urban Flood Susceptibility Mapping of Kochi Taluk Using Remote Sensing and GIS. Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT), Erode, India, 1–6. https://doi.org/10.1109/ICECCT52121.2021.9616790
44. The News Minute. 2015. Body of six-year-old washed away in Vizag drain, found 50 km off coast (Accessed on 21September, 2024) (https://www.thenewsminute.com/andhra-pradesh/body-six-year-old-washed-away-vizag-drain-found-50-km-coast-34811)
45. The News Minute. 2021. Andhra flash floods: 30 passengers travelling in RTC buses washed away, 12 dead. (Accessed on 24 December, 2023) (https://www.thenewsminute.com/andhra-pradesh/andhra-flash-floods-30-passengers-travelling-rtc-buses-washed-away-12-dead-157835).
46. Vanama V.S., Shitole S., Rao Y.S. 2020. Urban Flood Mapping with C-band RISAT-1 SAR Images: 2016 Flood Event of Bangalore City, India, 2020 International Conference on Convergence to Digital World - Quo Vadis (ICCDW), Mumbai, India, 1–4, https://doi.org/10.1109/ICCDW45521.2020.9318710
47. Qi W., Ma C., Xu H., Chen, Z., Zhao K., Han H. 2021. A review on applications of urban flood models in flood mitigation strategies, Natural Hazards, 108, 31–62. https://doi.org/10.1007/s11069-021-04715-8
48. Yin Y. 2023. Research on natural disaster target change detection method based on deep learning, IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA), Shenyang, China, 946–950. https://doi.org/10.1109/ICPECA56706.2023.10076044